System Management Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 7.0.x

First Published: 2020-03-01

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

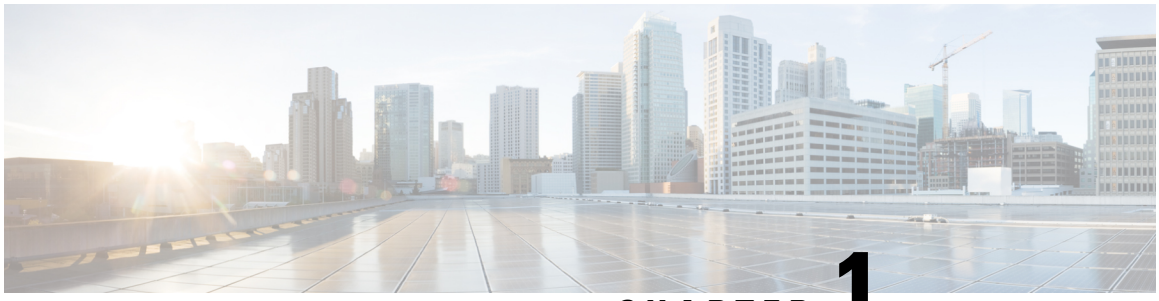
CONTENTS

Preface

Note This product has reached end-of-life status. For more information, see the End-of-Life and End-of-Sale Notices.

This guide describes the System Management configuration details for Cisco IOS XR software. This chapter contains details on the changes made to this document.

Preface



CHAPTER 1

Configuring Physical and Virtual Terminals

Line templates define standard attribute settings for incoming and outgoing transport over physical and virtual terminal lines (vtys). Vty pools are used to apply template settings to ranges of vtys.

This module describes the tasks you need to implement physical and virtual terminals on your Cisco IOS XR network.

Prerequisites for Implementing Physical and Virtual Terminals

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Implementing Physical and Virtual Terminals

To implement physical and virtual terminals, you need to understand the concepts in this section.

Tip You can programmatically manage the physical and virtual terminals using `openconfig-system-terminal.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

Line Templates

The following line templates are available in the Cisco IOS XR software.

- Default line template—The default line template that applies to a physical and virtual terminal lines.
- Console line template—The line template that applies to the console line.

- User-defined line templates—User-defined line templates that can be applied to a range of virtual terminal lines.

Line Template Configuration Mode

Changes to line template attributes are made in line template configuration mode. To enter line template configuration mode, issue the **line** command from XR Config mode, specifying the template to be modified. These line templates can be configured with the **line** command:

- console—console template
- default—default template
- template—user-defined template

After you specify a template with the **line** command, the router enters line template configuration mode where you can set the terminal attributes for the specified line. This example shows how to specify the attributes for the console:

```
RP/0/RP0/CPU0:router(config)# line console
RP/0/RP0/CPU0:router(config-line)#
```

From line template configuration mode, use the online help feature (**?**) to view all available options. Some useful options include:

- absolute-timeout—Specifies a timeout value for line disconnection.
- escape-character—Changes the line escape character.
- exec-timeout—Specifies the EXEC timeout.
- length—Sets the number of lines displayed on the screen.
- session-limit—Specifies the allowable number of outgoing connections.
- session-timeout—Specifies an interval for closing the connection if there is no input traffic.
- timestamp—Displays the timestamp before each command.
- width—Specifies the width of the display terminal.

Note The *default* session-limit for line template is applicable to Telnet sessions only. It is not applicable for SSH sessions.

Line Template Guidelines

The following guidelines apply to modifying the console template and to configuring a user-defined template:

- Modify the templates for the physical terminal lines on the router (the console port) from line template configuration mode. Use the **line console** command from XR Config mode to enter line template configuration mode for the console template.

- Modify the template for virtual lines by configuring a user-defined template with the **line** *template-name* command, configuring the terminal attributes for the user-defined template from line template configuration, and applying the template to a range of virtual terminal lines using the **vty pool** command.

Attributes not defined in the console template, or any virtual template, are taken from the default template.

The default settings for the default template are described for all commands in line template configuration mode in the *Terminal Services Commands on* module in *System Management Command Reference for Cisco 8000 Series Routers*.

Note Before creating or modifying the vty pools, enable the telnet server using the **telnet server** command in XR Config mode. See *IP Addresses and Services Configuration Guide for Cisco 8000 Series Routers* and *IP Addresses and Services Command Reference for Cisco 8000 Series Routers* for more information.

Terminal Identification

The physical terminal lines for the console port is identified by its location, expressed in the format of *rack*/*slot*/*module* , on the active or standby route processor (RP) where the respective console port resides. For virtual terminals, physical location is not applicable; the Cisco IOS XR software assigns a vty identifier to vtys according to the order in which the vty connection has been established.

vty Pools

Each virtual line is a member of a pool of connections using a common line template configuration. Multiple vty pools may exist, each containing a defined number of vtys as configured in the vty pool. The Cisco IOS XR software supports the following vty pools by default:

- Default vty pool—The default vty pool consists of five vtys (vtys 0 through 4) that each reference the default line template.
- Default fault manager pool—The default fault manager pool consists of six vtys (vtys 100 through 105) that each reference the default line template.

In addition to the default vty pool and default fault manager pool, you can also configure a user-defined vty pool that can reference the default template or a user-defined template.

When configuring vty pools, follow these guidelines:

- The vty range for the default vty pool must start at vty 0 and must contain a minimum of five vtys.
- The vty range from 0 through 99 can reference the default vty pool.
- The vty range from 5 through 99 can reference a user-defined vty pool.
- The vty range from 100 is reserved for the fault manager vty pool.
- The vty range for fault manager vty pools must start at vty 100 and must contain a minimum of six vtys.
- A vty can be a member of only one vty pool. A vty pool configuration will fail if the vty pool includes a vty that is already in another pool.
- If you attempt to remove an active vty from the active vty pool when configuring a vty pool, the configuration for that vty pool will fail.

How to Implement Physical and Virtual Terminals on Cisco IOS XR Software

Modifying Templates

This task explains how to modify the terminal attributes for the console and default line templates. The terminal attributes that you set will modify the template settings for the specified template.

SUMMARY STEPS

1. **configure**
2. **line** {**console** | **default**}
3. Configure the terminal attribute settings for the specified template using the commands in line template configuration mode.
4. Use one of the following commands:
   - **end**
   - **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.

Step 2	**line** {**console** | **default**}
	Example:
	`RP/0/RP0/CPU0:router(config)# line console`
	or
	`RP/0/RP0/CPU0:router(config)# line default`
		Enters line template configuration mode for the specified line template.
		• **console** —Enters line template configuration mode for the console template.
		• **default** —Enters line template configuration mode for the default line template.

Step 3	Configure the terminal attribute settings for the specified template using the commands in line template configuration mode.	—
Step 4	Use one of the following commands: • **end** • **commit** **Example:** `RP/0/RP0/CPU0:router(config-line)# end`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`

	Command or Action	Purpose
	or `RP/0/RP0/CPU0:router(config-line)# commit`	• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Creating and Modifying vty Pools

This task explains how to create and modify vty pools.

SUMMARY STEPS

1. **configure**
2. **telnet** {**ipv4** | **ipv6**} **server max-servers** *limit*
3. **line template** *template-name*
4. Configure the terminal attribute settings for the specified line template using the commands in line template configuration mode.
5. **exit**
6. **vty-pool** {**default** | *pool-name* | **eem**} *first-vty last-vty* [**line-template** {**default** | *template-name*}]
7. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**telnet** {**ipv4**	**ipv6**} **server max-servers** *limit* **Example:** `RP/0/RP0/CPU0:router(config)# telnet ipv4 server max-servers 10`	Specifies the number of allowable Telnet servers. Up to 100 Telnet servers are allowed. **Note** By default no Telnet servers are allowed. You must configure this command in order to enable the use of Telnet servers.

	Command or Action	Purpose
Step 3	**line template** *template-name* **Example:** `RP/0/RP0/CPU0:router(config)# line template 1`	Enters line template configuration mode for a user-defined template.
Step 4	Configure the terminal attribute settings for the specified line template using the commands in line template configuration mode.	—
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-line)# exit`	Exits line template configuration mode and returns the router to global configuration mode.

Step 6	**vty-pool** {**default** | *pool-name* | **eem**} *first-vty last-vty* [**line-template** {**default** | *template-name*}]
	Example:
	`RP/0/RP0/CPU0:router(config)# vty-pool default 0 5 line-template default`
	or
	`RP/0/RP0/CPU0:router(config)# vty-pool pool1 5 50 line-template template1`
	or
	`RP/0/RP0/CPU0:router(config)# vty-pool eem 100 105 line-template template1`
		Creates or modifies vty pools.
		• If you do not specify a line template with the **line-template** keyword, a vty pool defaults to the default line template.
		• **default** —Configures the default vty pool.
		  • The default vty pool must start at vty 0 and must contain a minimum of five vtys (vtys 0 through 4).
		  • You can resize the default vty pool by increasing the range of vtys that compose the default vty pool.
		• *pool-name* —Creates a user-defined vty pool.
		  • A user-defined pool must start at least at vty 5, depending on whether the default vty pool has been resized.
		  • If the range of vtys for the default vty pool has been resized, use the first range value free from the default line template. For example, if the range of vtys for the default vty pool has been configured to include 10 vtys (vty 0 through 9), the range value for the user-defined vty pool must start with vty 10.
		• **eem** —Configures the embedded event manager pool.
		  • The default embedded event manager vty pool must start at vty 100 and must contain a minimum of six vtys (vtys 100 through 105).

	Command or Action	Purpose
		• **line-template** *template-name* —Configures the vty pool to reference a user-defined template.
Step 7	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Monitoring Terminals and Terminal Sessions

This task explains how to monitor terminals and terminal sessions using the **show** EXEC commands available for physical and terminal lines.

Note The commands can be entered in any order.

SUMMARY STEPS

1. (Optional) **show line** [**aux location** *node-id* | **console location** *node-id* | **vty** *number*]
2. (Optional) **show terminal**
3. (Optional) **show users**

DETAILED STEPS

	Command or Action	Purpose
Step 1	(Optional) **show line** [**aux location** *node-id* | **console location** *node-id* | **vty** *number*] **Example:** `RP/0/RP0/CPU0:router# show line`	Displays the terminal parameters of terminal lines. • Specifying the **show line aux location** *node-id* EXEC command displays the terminal parameters of the auxiliary line. • Specifying the **show line console location** *node-id* EXEC command displays the terminal parameters of the console. • For the **location** *node-id* keyword and argument, enter the location of the Route Processor (RP) on which the respective auxiliary or console port resides.

	Command or Action	Purpose
		• The *node-id* argument is expressed in the format of *rack*/*slot*/*module* . • Specifying the **show line vty** *number* EXEC command displays the terminal parameters for the specified vty.
Step 2	(Optional) **show terminal** **Example:** `RP/0/RP0/CPU0:router# show terminal`	Displays the terminal attribute settings for the current terminal line.
Step 3	(Optional) **show users** **Example:** `RP/0/RP0/CPU0:router# show users`	Displays information about the active lines on the router.

Configuration Examples for Implementing Physical and Virtual Terminals

Modifying the Console Template: Example

This configuration example shows how to modify the terminal attribute settings for the console line template:

```
line console
  exec-timeout 0 0
  escape-character 0x5a
  session-limit 10
  disconnect-character 0x59
  session-timeout 100
  transport input telnet
  transport output telnet
```

In this configuration example, the following terminal attributes are applied to the console line template:

- The EXEC time out for terminal sessions is set to 0 minutes, 0 seconds. Setting the EXEC timeout to 0 minutes and 0 seconds disables the EXEC timeout function; thus, the EXEC session for the terminal session will never time out.
- The escape character is set to the 0x5a hexadecimal value (the 0x5a hexadecimal value translates into the “Z” character).
- The session limit for outgoing terminal sessions is set to 10 connections.
- The disconnect character is set to 0x59 hexadecimal value (the 0x59 hexadecimal character translates into the “Y” character).

- The session time out for outgoing terminal sessions is set to 100 minutes (1 hour and 40 minutes).
- The allowed transport protocol for incoming terminal sessions is Telnet.
- The allowed transport protocol for outgoing terminal sessions is Telnet.

To verify that the terminal attributes for the console line template have been applied to the console, use the **show line** command:

```
RP/0/RP0/CPU0:router# show line console location 0/0/CPU0

Tty             Speed     Modem  Uses   Noise Overruns           Acc I/O
*  con0/0/CPU0      9600        -     -       -      0/0               -/-

Line con0_0_CPU0, Location "Unknown", Type "Unknown"
Length: 24 lines, Width: 80 columns
Baud rate (TX/RX) is 9600, 1 parity, 2 stopbits, 8 databits
Template: console
Config:
Allowed transports are telnet.
```

Modifying the Default Template: Example

This configuration example shows how to override the terminal settings for the default line template:

```
line default
  exec-timeout 0 0
  width 512
  length 512
```

In this example, the following terminal attributes override the default line template default terminal attribute settings:

- The EXEC timeout for terminal sessions is set to 0 minutes and 0 seconds. Setting the EXEC timeout to 0 minutes and 0 seconds disables the EXEC timeout function; thus, the EXEC session for the terminal session will never time out (the default EXEC timeout for the default line template is 10 minutes).
- The width of the terminal screen for the terminals referencing the default template is set to 512 characters (the default width for the default line template is 80 characters).
- The length, the number of lines that will display at one time on the terminal referencing the default template, is set to 512 lines (the default length for the default line template is 24 lines).

Configuring a User-Defined Template to Reference the Default vty Pool: Example

This configuration example shows how to configure a user-defined line template (named test in this example) for vtys and to configure the line template test to reference the default vty pool:

```
line template test
  exec-timeout 100 0
  width 100
  length 100
  exit
vty-pool default 0 4 line-template test
```

Configuring a User-Defined Template to Reference a User-Defined vty Pool: Example

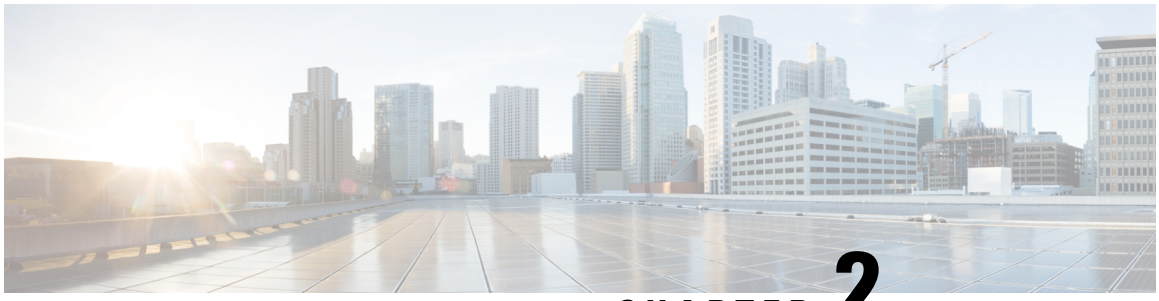
This configuration example shows how to configure a user-defined line template (named test2 in this example) for vtys and to configure the line template test to reference a user-defined vty pool (named pool1 in this example):

```
line template test2
  exec-timeout 0 0
  session-limit 10
  session-timeout 100
  transport input all
  transport output all
  exit
vty-pool pool1 5 50 line-template test2
```

Configuring a User-Defined Template to Reference the Fault Manager vty Pool: Example

This configuration example shows how to configure a user-defined line template (named test3 in this example) for vtys and to configure the line template test to reference the fault manager vty pool:

```
line template test3
  width 110
  length 100
  session-timeout 100
  exit
  vty-pool eem 100 106 line-template test3
```



CHAPTER 2

Configuring Simple Network Management Protocol

Simple Network Management Protocol (SNMP) is an application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language used for the monitoring and management of devices in a network.

This module describes the tasks you need to implement SNMP on your Cisco IOS XR network.

Prerequisites for Implementing SNMP

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Restrictions for SNMP use on Cisco IOS XR Software

SNMP outputs are only 32-bits wide and therefore cannot display any information greater than 2^{32}. 2^{32} is equal to 4.29 Gigabits.

Note A10 Gigabit interface is greater than 2^{32}, so if you are trying to display speed information regarding the interface, you might see concatenated results.

To display correct speed of an interface greater than 10 Gigabit, ifHighSpeed can be used.

The recommended maximum number of object identifiers (OIDs) that can be accommodated in a single SNMP request is 75. A request with more than 75 OIDs can result in SNMP requests being dropped with SNMP polling timeout.

Information about Implementing SNMP

To implement SNMP, you need to understand the concepts described in this section.

SNMP Functional Overview

The SNMP framework consists of three parts:

- SNMP manager
- SNMP agent
- Management Information Base (MIB)

SNMP Manager

The SNMP manager is the system used to control and monitor the activities of network hosts using SNMP. The most common managing system is called a *network management system* (NMS). The term NMS can be applied to either a dedicated device used for network management, or the applications used on such a device. A variety of network management applications are available for use with SNMP. These features range from simple command-line applications to feature-rich graphical user interfaces (such as the CiscoWorks 2000 line of products).

SNMP Agent

The SNMP agent is the software component within the managed device that maintains the data for the device and reports these data, as needed, to managing systems. The agent and MIB reside on the router. To enable the SNMP agent, you must define the relationship between the manager and the agent.

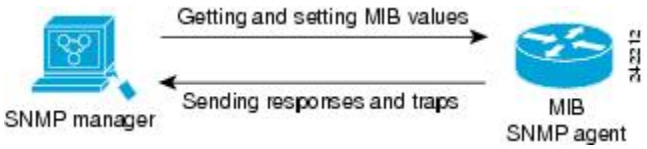
MIB

The *Management Information Base* (MIB) is a virtual information storage area for network management information, which consists of collections of managed objects. Within the MIB there are collections of related objects, defined in MIB modules. MIB modules are written in the SNMP MIB module language, as defined in STD 58, RFC 2578, RFC 2579, and RFC 2580. Note that individual MIB modules are also referred to as MIBs; for example, the Interfaces Group MIB (IF-MIB) is a MIB module within the MIB on your system.

The SNMP agent contains MIB variables whose values the SNMP manager can request or change through Get or Set operations. A manager can get a value from an agent or store a value into that agent. The agent gathers data from the MIB, the repository for information about device parameters and network data. The agent can also respond to manager requests to get or set data.

This figure illustrates the communications relationship between the SNMP manager and agent. A manager can send the agent requests to get and set MIB values. The agent can respond to these requests. Independent of this interaction, the agent can send unsolicited notifications (traps) to the manager to notify the manager of network conditions.

Figure 1: Communication Between an SNMP Agent and Manager



IP-MIB Support

RFC4293 IP-MIB was specifically designed to provide IPv4 and IPv6 statistics individually. The **ipIfStatsTable** defined in RFC 4293, lists the interface specific statistics. IPv6 statistics support in ipIfStatsTable was added earlier but, IOS-XR implementation of IP-MIB did not support IPv4 statistics as per RFC4293 in earlier releases.

IOS-XR implementation of IP-MIB supports IPv4 statistics as per RFC4293. This will enable you to collect the IPV4 and IPv6 statistics separately for each interface. The **ipIfStatsTable** is indexed by two **sub-ids address type (IPv4 or IPv6)** and the **interface ifindex[1]**. The implementation of IP-MIB support for IPv4 and IPv6 is separated for better readability and maintainability.

The list of OIDs added to the **ipIfStatsTable** for IPv4 statistics are:

- ipIfStatsInReceives
- ipIfStatsHCInReceives
- ipIfStatsInOctets
- ipIfStatsHCInOctets
- ipIfStatsOutTransmits
- ipIfStatsHCOutTransmits
- ipIfStatsOutOctets
- ipIfStatsHCOutOctets
- ipIfStatsDiscontinuityTime

For more information on the list of new OIDs added for iPv4 statistics, see SNMP OID Navigator.

SNMP Versions

Cisco IOS XR software supports the following versions of SNMP:

- Simple Network Management Protocol Version 1 (SNMPv1)
- Simple Network Management Protocol Version 2c (SNMPv2c)
- Simple Network Management Protocol Version 3 (SNMPv3)

Both SNMPv1 and SNMPv2c use a community-based form of security. The community of managers able to access the agent MIB is defined by an IP address access control list and password.

SNMPv2c support includes a bulk retrieval mechanism and more detailed error message reporting to management stations. The bulk retrieval mechanism supports the retrieval of tables and large quantities of information, minimizing the number of round-trips required. The SNMPv2c improved error handling support

includes expanded error codes that distinguish different kinds of error conditions; these conditions are reported through a single error code in SNMPv1. Error return codes now report the error type. Three kinds of exceptions are also reported: no such object exceptions, no such instance exceptions, and end of MIB view exceptions.

SNMPv3 is a security model. A *security model* is an authentication strategy that is set up for a user and the group in which the user resides. A *security level* is the permitted level of security within a security model. A combination of a security model and a security level will determine which security mechanism is employed when an SNMP packet is handled. See Security Models and Levels for SNMPv1, v2, v3, on page 15 for a list of security levels available in SNMPv3. The SNMPv3 feature supports RFCs 3411 to 3418.

You must configure the SNMP agent to use the version of SNMP supported by the management station. An agent can communicate with multiple managers; for this reason, you can configure the Cisco IOS-XR software to support communications with one management station using the SNMPv1 protocol, one using the SNMPv2c protocol, and another using SMNPv3.

Comparison of SNMPv1, v2c, and v3

SNMP v1, v2c, and v3 all support the following operations:

- get-request—Retrieves a value from a specific variable.
- get-next-request—Retrieves the value following the named variable; this operation is often used to retrieve variables from within a table. With this operation, an SNMP manager does not need to know the exact variable name. The SNMP manager searches sequentially to find the needed variable from within the MIB.
- get-response—Operation that replies to a get-request, get-next-request, and set-request sent by an NMS.
- set-request—Operation that stores a value in a specific variable.
- trap—Unsolicited message sent by an SNMP agent to an SNMP manager when some event has occurred.

This table identifies other key SNMP features supported by the SNMP v1, v2c, and v3.

Table 1: SNMPv1, v2c, and v3 Feature Support

Feature	SNMP v1	SNMP v2c	SNMP v3
Get-Bulk Operation	No	Yes	Yes
Inform Operation	No	Yes	Yes
64 Bit Counter	No	Yes	Yes
Textual Conventions	No	Yes	Yes
Authentication	No	No	Yes
Privacy (Encryption)	No	No	Yes
Authorization and Access Controls (Views)	No	No	Yes

Security Models and Levels for SNMPv1, v2, v3

The security level determines if an SNMP message needs to be protected from disclosure and if the message needs to be authenticated. The various security levels that exist within a security model are as follows:

- noAuthNoPriv—Security level that does not provide authentication or encryption.
- authNoPriv—Security level that provides authentication but does not provide encryption.
- authPriv—Security level that provides both authentication and encryption.

Three security models are available: SNMPv1, SNMPv2c, and SNMPv3. The security model combined with the security level determine the security mechanism applied when the SNMP message is processed.

The below table identifies what the combinations of security models and levels mean.

Table 2: SNMP Security Models and Levels

Model	Level	Authentication	Encryption	What Happens
v1	noAuthNoPriv	Community string	No	Uses a community string match for authentication.
v2c	noAuthNoPriv	Community string	No	Uses a community string match for authentication.
v3	noAuthNoPriv	Username	No	Uses a username match for authentication.
v3	authNoPriv	HMAC-MD5 or HMAC-SHA	No	Provides authentication based on the HMAC[1]-MD5[2] algorithm or the HMAC-SHA[3].
v3	authPriv	HMAC-MD5 or HMAC-SHA	DES	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Provides DES[4] 56-bit encryption in addition to authentication based on the CBC[5] DES (DES-56) standard.
v3	authPriv	HMAC-MD5 or HMAC-SHA	3DES	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Provides 168-bit 3DES[6] level of encryption.
v3	authPriv	HMAC-MD5 or HMAC-SHA	AES	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Provides 128-bit AES[7] level of encryption.

1 Hash-Based Message Authentication Code
2 Message Digest 5
3 Secure Hash Algorithm
4 Data Encryption Standard
5 Cipher Block Chaining
6 Triple Data Encryption Standard
7 Advanced Encryption Standard

Use of 3DES and AES encryption standards requires that the security package be installed. For information on installing software packages, see *Upgrading and Managing Cisco IOS XR Software*.

SNMPv3 Benefits

SNMPv3 provides secure access to devices by providing authentication, encryption and access control. These added security benefits secure SNMP against the following security threats:

- Masquerade—The threat that an SNMP user may assume the identity of another SNMP user to perform management operations for which that SNMP user does not have authorization.
- Message stream modification—The threat that messages may be maliciously reordered, delayed, or replayed (to an extent that is greater than can occur through the natural operation of a subnetwork service) to cause SNMP to perform unauthorized management operations.
- Disclosure—The threat that exchanges between SNMP engines could be eavesdropped. Protecting against this threat may be required as a matter of local policy.

In addition, SNMPv3 provides access control over protocol operations on SNMP managed objects.

SNMPv3 Costs

SNMPv3 authentication and encryption contribute to a slight increase in the response time when SNMP operations on MIB objects are performed. This cost is far outweighed by the security advantages provided by SNMPv3.

This table shows the order of response time (from least to greatest) for the various security model and security level combinations.

Table 3: Order of Response Times from Least to Greatest

Security Model	Security Level
SNMPv2c	noAuthNoPriv
SNMPv3	noAuthNoPriv
SNMPv3	authNoPriv
SNMPv3	authPriv

User-Based Security Model

SNMPv3 User-Based Security Model (USM) refers to SNMP message-level security and offers the following services:

- Message integrity—Ensures that messages have not been altered or destroyed in an unauthorized manner and that data sequences have not been altered to an extent greater than can occur nonmaliciously.
- Message origin authentication—Ensures that the claimed identity of the user on whose behalf received data was originated is confirmed.
- Message confidentiality—Ensures that information is not made available or disclosed to unauthorized individuals, entities, or processes.

SNMPv3 authorizes management operations only by configured users and encrypts SNMP messages.

USM uses two authentication protocols:

- HMAC-MD5-96 authentication protocol
- HMAC-SHA-96 authentication protocol

USM uses Cipher Block Chaining (CBC)-DES (DES-56) as the privacy protocol for message encryption.

View-Based Access Control Model

The View-Based Access Control Model (VACM) enables SNMP users to control access to SNMP managed objects by supplying read, write, or notify access to SNMP objects. It prevents access to objects restricted by views. These access policies can be set when user groups are configured with the **snmp-server group** command.

MIB Views

For security reasons, it is often valuable to be able to restrict the access rights of some groups to only a subset of the management information within the management domain. To provide this capability, access to a management object is controlled through MIB views, which contain the set of managed object types (and, optionally, the specific instances of object types) that can be viewed.

Access Policy

Access policy determines the access rights of a group. The three types of access rights are as follows:

- read-view access—The set of object instances authorized for the group when objects are read.
- write-view access—The set of object instances authorized for the group when objects are written.
- notify-view access—The set of object instances authorized for the group when objects are sent in a notification.

IP Precedence and DSCP Support for SNMP

SNMP IP Precedence and differentiated services code point (DSCP) support delivers QoS specifically for SNMP traffic. You can change the priority setting so that SNMP traffic generated in a router is assigned a specific QoS class. The IP Precedence or IP DSCP code point value is used to determine how packets are handled in weighted random early detection (WRED).

After the IP Precedence or DSCP is set for the SNMP traffic generated in a router, different QoS classes cannot be assigned to different types of SNMP traffic in that router.

The IP Precedence value is the first three bits in the type of service (ToS) byte of an IP header. The IP DSCP code point value is the first six bits of the differentiate services (DiffServ Field) byte. You can configure up to eight different IP Precedence markings or 64 different IP DSCP markings.

Session MIB support on subscriber sessions

SNMP monitoring requires information about subscribers of all types. The CISCO-SUBSCRIBER-SESSION-MIB is defined to model per-subscriber data as well as aggregate subscriber (PPPoE) data. It is required to support notifications (traps) for aggregate session counts crossing configured

thresholds. Generic MIB Data Collector Manager (DCM) support for CISCO-SUBSCRIBER-SESSION-MIB, helps faster data collection and also better handling of parallel data.

SNMP Notifications

A key feature of SNMP is the ability to generate notifications from an SNMP agent. These notifications do not require that requests be sent from the SNMP manager. On Cisco IOS XR software, unsolicited (asynchronous) notifications can be generated only as *traps*. Traps are messages alerting the SNMP manager to a condition on the network. Notifications can indicate improper user authentication, restarts, the closing of a connection, loss of connection to a neighbor router, or other significant events.

Note Inform requests (inform operations) are supported in Cisco IOS XR software.

Traps are less reliable than informs because the receiver does not send any acknowledgment when it receives a trap. The sender cannot determine if the trap was received. An SNMP manager that receives an inform request acknowledges the message with an SNMP response protocol data unit (PDU). If the manager does not receive an inform request, it does not send a response. If the sender never receives a response, the inform request can be sent again. Thus, informs are more likely to reach their intended destination.

However, traps are often preferred because informs consume more resources in the router and in the network. Unlike a trap, which is discarded as soon as it is sent, an inform request must be held in memory until a response is received or the request times out. Also, traps are sent only once, and an inform may be retried several times. The retries increase traffic and contribute to a higher overhead on the network. Thus, traps and inform requests provide a trade-off between reliability and resources.

Figure 2: Trap Received by the SNMP Manager

In this illustration, the agent router sends a trap to the SNMP manager. Although the manager receives the trap, it does not send any acknowledgment to the agent. The agent has no way of knowing that the trap reached its destination.

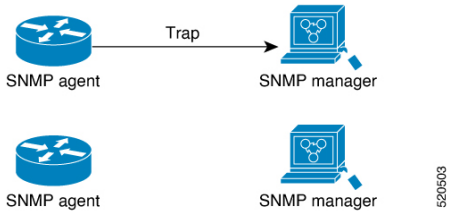
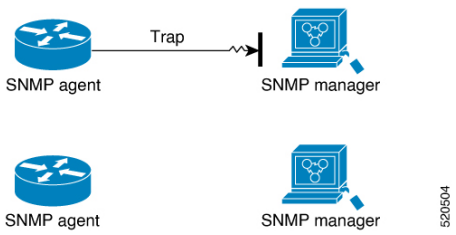


Figure 3: Trap Not Received by the SNMP Manager

In this illustration, the agent sends a trap to the manager, but the trap does not reach the manager. Because the agent has no way of knowing that the trap did not reach its destination, the trap is not sent again. The manager never receives the trap.



Session Types

The supported session types are:

- PPPoE
- IP SUB PKT
- IP SUB DHCP

How to Implement SNMP on Cisco IOS XR Software

This section describes how to implement SNMP.

The **snmp-server** commands enable SNMP on Management Ethernet interfaces by default. For information on how to enable SNMP server support on other inband interfaces, see the *Implementing Management Plane Protection on Cisco IOS XR Software* module in *System Security Configuration Guide for Cisco 8000 Series Routers*.

Configuring SNMPv3

This task explains how to configure SNMPv3 for network management and monitoring.

Note No specific command enables SNMPv3; the first **snmp-server** global configuration command (config), that you issue enables SNMPv3. Therefore, the sequence in which you issue the **snmp-server** commands for this task does not matter.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 (Optional) **snmp-server engineid local** *engine-id*

Example:

```
RP/0/RP0/CPU0:router# snmp-server engineID
local 00:00:00:09:00:00:00:a1:61:6c:20:61
```

Specifies the identification number of the local SNMP engine.

Step 3 **snmp-server view** *view-name oid-tree* {**included** | **excluded**}

Example:

```
RP/0/RP0/CPU0:router# snmp-server view
view_name 1.3.6.1.2.1.1.5 included
```

Creates or modifies a view record.

Step 4 **snmp-server group** *name* {**v1** | **v2c** | **v3** {**auth** | **noauth** | **priv**}} [**read** *view*] [**write** *view*] [**notify** *view*] [*access-list-name*]

Example:

```
RP/0/RP0/CPU0:router# snmp-server group
group_name v3 noauth read view_name1 write view_name2
```

Configures a new SNMP group or a table that maps SNMP users to SNMP views.

Step 5 **snmp-server user** *username groupname* {**v1** | **v2c** | **v3** [**auth** {**md5** | **sha**} {**clear** | **encrypted**} *auth-password* [**priv des56** {**clear** | **encrypted**} *priv-password*]]} [*access-list-name*]

Example:

```
RP/0/RP0/CPU0:router# snmp-server user
noauthuser group_name v3
```

Configures a new user to an SNMP group.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 7 (Optional) **show snmp**

Example:

```
RP/0/RP0/CPU0:router# show snmp
```

Displays information about the status of SNMP.

Step 8 (Optional) **show snmp engineid**

Example:

```
RP/0/RP0/CPU0:router# show snmp engineid
```

Displays information about the local SNMP engine.

Step 9 (Optional) **show snmp group**

Example:

```
RP/0/RP0/CPU0:router# show snmp group
```

Displays information about each SNMP group on the network.

Step 10 (Optional) **show snmp users**

Example:

```
RP/0/RP0/CPU0:router# show snmp users
```

Displays information about each SNMP username in the SNMP users table.

Step 11 (Optional) **show snmp view**

Example:

```
RP/0/RP0/CPU0:router# show snmp view
```

Displays information about the configured views, including the associated MIB view family name, storage type, and status.

Configure to Drop Error PDUs

Perform this configuration to avoid error PDUs being sent out of router when polled with incorrect SNMPv3 user name. If the configuration is not set, it will respond with error PDUs by default. After applying this configuration, when router is polled with unknown SNMPv3 user name, the NMS will get time out instead of getting unknown user name error code.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **snmp-server drop unknown-user**

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server drop unknown-user
```

Drop the error PDUs when the router is polled with incorrect SNMPv3 user name.

Step 3 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring SNMPv3: Examples

Setting an Engine ID

This example shows how to set the identification of the local SNMP engine:

```
snmp-server engineID local 00:00:00:09:00:00:00:a1:61:6c:20:61
```

Note After the engine ID has been configured, the SNMP agent restarts.

Verifying the Identification of the Local SNMP Engines

This example shows how to verify the identification of the local SNMP engine:

```
config
  show snmp engineid

  SNMP engineID 00000009000000a1ffffffff
```

Creating a View

There are two ways to create a view:

- You can include the object identifier (OID) of an ASN.1 subtree of a MIB family from a view by using the **included** keyword of the **snmp-server view** command.
- You can exclude the OID subtree of the ASN.1 subtree of a MIB family from a view by using the **excluded** keyword of the **snmp-server view** command.

This example shows how to create a view that includes the sysName (1.3.6.1.2.1.1.5) object:

```
config
  snmp-server view SNMP_VIEW1 1.3.6.1.2.1.1.5 included
```

This example shows how to create a view that includes all the OIDs of a system group:

```
config
  snmp-server view SNMP_VIEW1 1.3.6.1.2.1.1 included
```

This example shows how to create a view that includes all the OIDs under the system group except the sysName object (1.3.6.1.2.1.1.5), which has been excluded:

```
config
  snmp-server view SNMP_VIEW1 1.3.6.1.2.1.1 included
  snmp-server view SNMP_VIEW1 1.3.6.1.2.1.1.5 excluded
```

Verifying Configured Views

This example shows how to display information about the configured views:

```
RP/0/RP0/CPU0:router# show snmp view

  v1default 1.3.6.1 - included nonVolatile active
  SNMP_VIEW1 1.3.6.1.2.1.1 - included nonVolatile active
  SNMP_VIEW1 1.3.6.1.2.1.1.5 - excluded nonVolatile active
```

Creating Groups

If you do not explicitly specify a notify, read, or write view, the Cisco IOS XR software uses the v1 default (1.3.6.1). This example shows how to create a group that utilizes the default view:

```
RP/0/RP0/CPU0:router# snmp-server group group-name v3 auth
```

The following configuration example shows how to create a group that has read access to all the OIDs in the system except the sysUpTime object (1.3.6.1.2.1.1.3), which has been excluded from the view applied to the group, but write access only to the sysName object (1.3.6.1.2.1.1.5):

```
!
  snmp-server view view_name1 1.3.6.1.2.1.1 included
  snmp-server view view_name1 1.3.6.1.2.1.1.3 excluded
  snmp-server view view_name2 1.3.6.1.2.1.1.5 included
  snmp-server group group_name1 v3 auth read view_name1 write view_name2
  !
```

Verifying Groups

This example shows how to verify the attributes of configured groups:

```
RP/0/RP0/CPU0:router# show snmp group

  groupname: group_name1                  security model:usm
  readview : view_name1                   writeview: view_name2
  notifyview: v1default
  row status: nonVolatile
```

Creating and Verifying Users

Given the following SNMPv3 view and SNMPv3 group configuration:

```
!
  snmp-server view view_name 1.3.6.1.2.1.1 included
  snmp-server group group_name v3 noauth read view_name write view-name
  !
```

This example shows how to create a noAuthNoPriv user with read and write view access to a system group:

```
config
  snmp-server user noauthuser group_name v3
```

Note The user must belong to a noauth group before a noAuthNoPriv user can be created.

This example shows how to verify the attributes that apply to the SNMP user:

```
RP/0/RP0/CPU0:router# show snmp user

  User name: noauthuser
  Engine ID: localSnmpID
  storage-type: nonvolatile active
```

Given the following SNMPv3 view and SNMPv3 group configuration:

```
!
  snmp-server view SNMP_VIEW1 1.3.6.1.2.1.1 included
  snmp-server group SNMP_GROUP1 v3 auth notify SNMP_VIEW1 read SNMP_VIEW1 write SNMP_VIEW1
 !
```

This example shows how to create a user with authentication (including encryption), read, and write view access to a system group:

```
config
  snmp-server user userv3authpriv SNMP_GROUP1 v3 auth md5 password123 priv aes 128 password123
```

Given the following SNMPv3 view and SNMPv3 group configuration:

```
!
  snmp-server view view_name 1.3.6.1.2.1.1 included
  snmp group group_name v3 priv read view_name write view_name
  !
```

This example shows how to create authNoPriv user with read and write view access to a system group:

```
RP/0/RP0/CPU0:router# snmp-server user authuser group_name v3 auth md5 clear auth_passwd
```

Note As the group is configured at a security level of Auth, the user must be configured as "auth" at a minimum to access this group ("priv" users could also access this group). The authNoPriv user configured in this group, authuser, must supply an authentication password to access the view. In the example, auth_passwd is set as the authentication password string. Note that **clear** keyword is specified before the auth_passwd password string. The **clear** keyword indicates that the password string being supplied is unencrypted.

This example shows how to verify the attributes that apply to SNMP user:

```
RP/0/RP0/CPU0:router# show snmp user

  User name: authuser
  Engine ID: localSnmpID
  storage-type: nonvolatile active
```

Given the following SNMPv3 view and SNMPv3 group configuration:

```
!
  snmp view view_name 1.3.6.1.2.1.1 included
  snmp group group_name v3 priv read view_name write view_name
  !
```

This example shows how to create an authPriv user with read and write view access to a system group:

```
config
  snmp-server user privuser group_name v3 auth md5 clear auth_passwd priv des56 clear
priv_passwd
```

Note As the group has a security level of Priv, the user must be configured as a "priv" user to access this group. In this example, the user, privuser, must supply both an authentication password and privacy password to access the OIDs in the view.

This example shows how to verify the attributes that apply to the SNMP user:

```
RP/0/RP0/CPU0:router# show snmp user

  User name: privuser
  Engine ID: localSnmpID
  storage-type: nonvolatile active
```

Configuring SNMP Trap Notifications

The following example shows how to configure the router to send SNMP trap notifications.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **snmp-servergroup***name*{**v1v2v3**{**auth** | **noauth** | **priv**}}[**read***view*]**write***view*] [**notify***view*] [*access-list-name*]

Example:

```
RP/0/RP0/CPU0:router# snmp-server group group_name v3 noauth read view_name1 writer view_name2
```

Configures a new SNMP group or a table that maps SNMP users to SNMP views.

Step 3 **snmp-serveruser***groupname*{**v1v2cv3**{**auth** | **md5** | **sha**}{**clear** | **encrypted**}*auth-password*] [**priv des56** {**clear** | *access-list-name*]

Example:

```
RP/0/RP0/CPU0:router# snmp-server group group_name v3 noauth read view_name1 writer view_name2
```

Configures a new SNMP group or a table that maps SNMP users to SNMP views.

Step 4 **snmp-serveruser***usernamegroupname*{**v1v2cv3**{**auth** | **md5** | **sha**}{**clear** | **encrypted**}*auth-password*] [**priv des56** {**clear** | *access-list-name*]

Example:

```
RP/0/RP0/CPU0:routerconfig# snmp-server user noauthuser group_name v3
```

Configures a new SNMP group or a table that maps SNMP users to SNMP views.

Step 5 [**snmp-server host** *address* [**traps**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [*notification-type*]

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server host 12.26.25.61 traps version 3
noauth userV3noauth
```

Specifies SNMP trap notifications, the version of SNMP to use, the security level of the notifications, and the recipient (host) of the notifications.

Step 6 **snmp-server traps** [*notification-type*]

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server traps bgp
```

Enables the sending of trap notifications and specifies the type of trap notifications to be sent.

- If a trap is not specified with the *notification-type* argument, all supported trap notifications are enabled on the router. To display which trap notifications are available on your router, enter the **snmp-server traps ?** command.

Step 7 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.

- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 8 (Optional) **show snmp host**

Example:

```
RP/0/RP0/CPU0:router# show snmp host
```

Displays information about the configured SNMP notification recipient (host), port number, and security model.

Configure to Drop Error PDUs

Perform this configuration to avoid error PDUs being sent out of router when polled with incorrect SNMPv3 user name. If the configuration is not set, it will respond with error PDUs by default. After applying this configuration, when router is polled with unknown SNMPv3 user name, the NMS will get time out instead of getting unknown user name error code.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **snmp-server drop unknown-user**

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server drop unknown-user
```

Drop the error PDUs when the router is polled with incorrect SNMPv3 user name.

Step 3 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Trap Notifications: Example

The following example configures an SNMP agent to send out different types of traps. The configuration includes a v2c user, a noAuthNoPriv user, anauthNoPriv user, and an AuthPriv user.

Note The default User Datagram Protocol (UDP) port is 161. If you do not a specify a UDP port with the **udp-port** keyword and *port* argument, then the configured SNMP trap notifications are sent to port 161.

```
!
  snmp-server host 10.50.32.170 version 2c public udp-port 2345
  snmp-server host 10.50.32.170 version 3 auth userV3auth udp-port 2345
  snmp-server host 10.50.32.170 version 3 priv userV3priv udp-port 2345
  snmp-server host 10.50.32.170 version 3 noauth userV3noauth udp-port 2345
  snmp-server user userv2c groupv2c v2c
  snmp-server user userV3auth groupV3auth v3 auth md5 encrypted 140F0A13
 snmp-server user userV3priv groupV3priv v3 auth md5 encrypted 021E1C43 priv des56 encrypted
1110001C
  snmp-server user userV3noauth groupV3noauth v3 LROwner
  snmp-server view view_name 1.3 included
  snmp-server community public RW
  snmp-server group groupv2c v2c read view_name
  snmp-server group groupV3auth v3 auth read view_name
  snmp-server group groupV3priv v3 priv read view_name
  snmp-server group groupV3noauth v3 noauth read view_name
  !
```

This example shows how to verify the configuration SNMP trap notification recipients host, the recipients of SNMP trap notifications. The output displays the following information:

- IP address of the configured notification host
- UDP port where SNMP notification messages are sent
- Type of trap configured
- Security level of the configured user
- Security model configured

```
config
  show snmp host

  Notification host: 10.50.32.170 udp-port: 2345 type: trap
  user: userV3auth security model: v3 auth

  Notification host: 10.50.32.170 udp-port: 2345 type: trap
  user: userV3noauth security model: v3 noauth

  Notification host: 10.50.32.170 udp-port: 2345 type: trap
  user: userV3priv security model: v3 priv

  Notification host: 10.50.32.170 udp-port: 2345 type: trap
  user: userv2c security model: v2c
```

Setting the Contact, Location, and Serial Number of the SNMP Agent

This task explains how to set the system contact string, system location string, and system serial number of the SNMP agent.

Note The sequence in which you issue the **snmp-server** commands for this task does not matter.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 (Optional) **snmp-server contact** *system-contact-string*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server contact
Dial System Operator at beeper # 27345
```

Sets the system contact string.

Step 3 (Optional) **snmp-server location** *system-location*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server location
Building 3/Room 214
```

Sets the system location string.

Step 4 (Optional) **snmp-server chassis-id** *serial-number*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server chassis-id 1234456
```

Sets the system serial number.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Defining the Maximum SNMP Agent Packet Size

This task shows how to configure the largest SNMP packet size permitted when the SNMP server is receiving a request or generating a reply.

Note The sequence in which you issue the **snmp-server** commands for this task does not matter.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 (Optional) **snmp-server packetsize** *byte-count*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server packetsize 1024
```

Sets the maximum packet size.

Step 3 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Changing Notification Operation Values

After SNMP notifications have been enabled, you can specify a value other than the default for the source interface, message queue length, or retransmission interval.

This task explains how to specify a source interface for trap notifications, the message queue length for each host, and the retransmission interval.

Note The sequence in which you issue the **snmp-server** commands for this task does not matter.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 (Optional) **snmp-server trap-source** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server trap-source POS 0/0/1/0
```

Specifies a source interface for trap notifications.

Step 3 (Optional) **snmp-server queue-length** *length*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server queue-length 20
```

Establishes the message queue length for each notification.

Step 4 (Optional) **snmp-server trap-timeout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server trap-timeout 20
```

Defines how often to resend notifications on the retransmission queue.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Setting IP Precedence and DSCP Values

This task describes how to configure IPv4 Precedence or IPv4 DSCP for SNMP traffic.

Before you begin

SNMP must be configured.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 Use one of the following commands:

- **snmp-server ipv4 precedence** *value*
- **snmp-server ipv4 dscp** *value*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server dscp 24
```

Configures an IPv4 precedence or IPv4 DSCP value for SNMP traffic.

Step 3 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Setting IPv6 Precedence and DSCP Values

This task describes how to configure IPv6 Precedence or IPv6 DSCP for SNMP traffic.

Before you begin

SNMP must be configured.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 Use one of the following commands:

- **snmp-server ipv6 precedence** *value*
- **snmp-server ipv6 dscp** *value*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server dscp 24
```

Configures an IPv6 precedence or IPv6 DSCP value for SNMP traffic.

Step 3 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Setting an IP Precedence Value for SNMP Traffic: Example

The following example shows how to set the SNMP IPv4 Precedence value to 7:

```
configure
  snmp-server ipv4 precedence 7
  exit

  Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]: y
```

The following example shows how to set the SNMP IPv6 Precedence value to 7:

```
configure
  snmp-server ipv6 precedence 7
  exit

  Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]: y
```

Setting an IP DSCP Value for SNMP Traffic: Example

The following example shows how to set the IPv4 DSCP value of SNMP traffic to 45:

```
configure
  snmp-server ipv4 dscp 45
  exit

  Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]: y
```

The following example shows how to set the IPv6 DSCP value of SNMP traffic to 45:

```
configure
  snmp-server ipv6 dscp 45
  exit

  Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]: y
```

Displaying SNMP Context Mapping

The SNMP agent serves queries based on SNMP contexts created by the client features. There is a context mapping table. Each entry in the context mapping table includes a context name, the name of the feature that created the context, and the name of the specific instance of the feature.

show snmp context-mapping

Example:

```
RP/0/RP0/CPU0:router# show snmp context-mapping
```

Displays the SNMP context mapping table.

Monitoring Packet Loss

It is possible to monitor packet loss by configuring the generation of SNMP traps when packet loss exceeds a specified threshold. The configuration described in this task enables the creation of entries in the MIB tables of the EVENT-MIB. This can then be monitored for packet loss using SNMP GET operations.

Before you begin

Note Entries created in the EVENT-MIB MIB tables using the configuration described in this task cannot be altered using an SNMP SET.

Entries to the EVENT-MIB MIB tables created using an SNMP SET cannot be altered using the configuration described in this task.

snmp-server mibs eventmib packet-loss *type interface-path-id* **falling** *lower-threshold* **interval** *sampling-interval* **rising** *upper-threshold*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server mibs eventmib packet-loss falling 1 interval 5 rising 2
```

Generates SNMP EVENT-MIB traps for the interface when the packet loss exceeds the specified thresholds. Up to 100 interfaces can be monitored.

falling *lower-threshold* —Specifies the lower threshold. When packet loss between two intervals falls below this threshold and an mteTriggerRising trap was generated previously, a SNMP mteTriggerFalling trap is generated. This trap is not generated until the packet loss exceeds the upper threshold and then falls back below the lower threshold.

interval *sampling-interval* —Specifies how often packet loss statistics are polled. This is a value between 5 and 1440 minutes, in multiples of 5.

rising *upper-threshold* —Specifies the upper threshold. When packet loss between two intervals increases above this threshold, an SNMP mteTriggreRising trap is generated. This trap is not generated until the packet loss drops below the lower threshold and then rises above the upper threshold.

Configuring MIB Data to be Persistent

Many SNMP MIB definitions define arbitrary 32-bit indices for their object tables. MIB implementations often do a mapping from the MIB indices to some internal data structure that is keyed by some other set of data. In these MIB tables the data contained in the table are often other identifiers of the element being modelled. For example, in the ENTITY-MIB, entries in the entPhysicalTable are indexed by the 31-bit value, entPhysicalIndex, but the entities could also be identified by the entPhysicalName or a combination of the other objects in the table.

Because of the size of some MIB tables, significant processing is required to discover all the mappings from the 32-bit MIB indices to the other data which the network management station identifies the entry. For this reason, it may be necessary for some MIB indices to be persistent across process restarts, switchovers, or device reloads. The ENTITY-MIB entPhysicalTable and CISCO-CLASS-BASED-QOS-MIB are two such MIBs that often require index values to be persistent.

Also, because of query response times and CPU utilization during CISCO-CLASS-BASED-QOS-MIB statistics queries, it is desirable to cache service policy statistics.

Step 1 (Optional) **snmp-server mibs cbqosmib persist**

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server mibs cbqosmib persist
```

Enables persistent storage of the CISCO-CLASS-BASED-QOS-MIB data.

Step 2 (Optional) **snmp-server cbqosmib cache refresh time** *time*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server mibs cbqosmib cache
refresh time 45
```

Enables QoS MIB caching with a specified cache refresh time.

Step 3 (Optional) **snmp-server cbqosmib cache service-policy count** *count*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server mibs cbqosmib cache
service-policy count 50
```

Enables QoS MIB caching with a limited number of service policies to cache.

Step 4 **snmp-server ifindex persist**

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server ifindex persist
```

Enables if Index persistence globally on all Simple Network Management Protocol (SNMP) interfaces.

Configuring LinkUp and LinkDown Traps for a Subset of Interfaces

By specifying a regular expression to represent the interfaces for which you are interested in setting traps, you can enable or disable linkUp and linkDown traps for a large number of interfaces simultaneously.

Before you begin

SNMP must be configured.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **snmp-server interface subset** *subset-number* **regular-expression** *expression*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server interface subset 10
    regular-expression "^Gig[a-zA-Z]+[0-9/]+\."
RP/0/RP0/CPU0:router(config-snmp-if-subset)#
```

Enters snmp-server interface mode for the interfaces identified by the regular expression.

The subset-number argument identifies the set of interfaces, and also assigns a priority to the subset in the event that an interface is included in more than one subset. Lower numbers have higher priority and their configuration takes precedent over interface subsets with higher numbers.

The *expression* argument must be entered surrounded by double quotes.

Step 3 **notification linkupdown disable**

Example:

```
RP/0/RP0/CPU0:router(config-snmp-if-subset)# notification linkupdown disable
```

Disables linkUp and linkDown traps for all interfaces being configured. To enable previously disabled interfaces, use the **no** form of this command.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes, and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration mode, without committing the configuration changes.

Step 5 (Optional) **show snmp interface notification subset** *subset-number*

Example:

```
RP/0/RP0/CPU0:router# show snmp interface notification subset 10
```

Displays the linkUp and linkDown notification status for all interfaces identified by the subset priority.

Step 6 (Optional) **show snmp interface notification regular-expression** *expression*

Example:

```
RP/0/RP0/CPU0:router# show snmp interface notification
    regular-expression "^Gig[a-zA-Z]+[0-9/]+\."
```

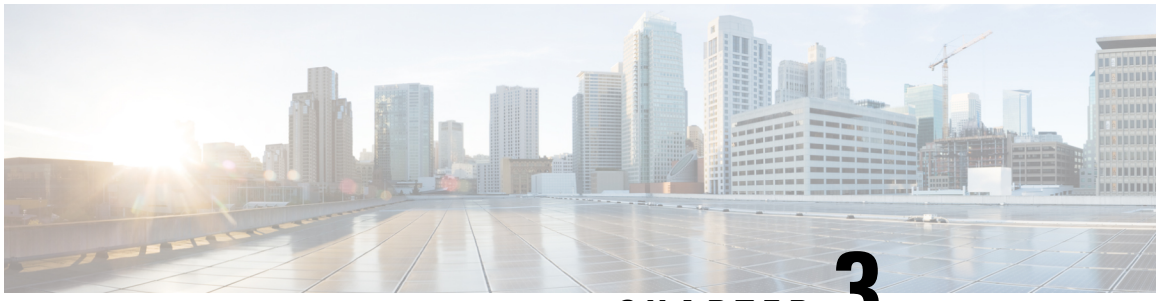
Displays the linkUp and linkDown notification status for all interfaces identified by the regular expression.

Step 7 (Optional) **show snmp interface notification** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router# show snmp interface notification
    tengige 0/4/0/3.10
```

Displays the linkUp and linkDown notification status for the specified interface.



CHAPTER 3

Configuring Periodic MIB Data Collection and Transfer

This document describes how to periodically transfer selected MIB data from your router to a specified Network Management System (NMS). The periodic MIB data collection and transfer feature is also known as bulk statistics.

Prerequisites for Periodic MIB Data Collection and Transfer

To use periodic MIB data collection and transfer, you should be familiar with the Simple Network Management Protocol (SNMP) model of management information. You should also know what MIB information you want to monitor on your network devices, and the OIDs or object names for the MIB objects to be monitored.

Information About Periodic MIB Data Collection and Transfer

SNMP Objects and Instances

A type (or class) of SNMP management information is called an object. A specific instance from a type of management information is called an object instance (or SNMP variable). To configure a bulk statistics collection, you must specify the object types to be monitored using a bulk statistics object list and the specific instances of those objects to be collected using a bulk statistics schema.

MIBs, MIB tables, MIB objects, and object indices can all be specified using a series of numbers called an object identifier (OID). OIDs are used in configuring a bulk statistics collection in both the bulk statistics object lists (for general objects) and in the bulk statistics schemas (for specific object instances).

Bulk Statistics Object Lists

To group the MIB objects to be polled, you need to create one or more object lists. A bulk statistics object list is a user-specified set of MIB objects that share the same MIB index. Object lists are identified using a name that you specify. Named bulk statistics object lists allow the same configuration to be reused in different bulk statistics schemas.

All the objects in an object list must share the same MIB index. However, the objects do not need to be in the same MIB and do not need to belong to the same MIB table. For example, it is possible to group ifInOctets and a CISCO-IF-EXTENSION-MIB object in the same schema, because the containing tables for both objects are indexed by the ifIndex.

Bulk Statistics Schemas

Data selection for the Periodic MIB Data Collection and Transfer Mechanism requires the definition of a schema with the following information:

- Name of an object list.
- Instance (specific instance or series of instances defined using a wild card) that needs to be retrieved for objects in the specified object list.
- How often the specified instances need to be sampled (polling interval). The default polling interval is 5 minutes.

A bulk statistics schema is also identified using a name that you specify. This name is used when configuring the transfer options.

Bulk Statistics Transfer Options

After configuring the data to be collected, a single virtual file (VFile or *bulk statistics file*) with all collected data is created. This file can be transferred to a network management station using FTP or TFTP. You can specify how often this file should be transferred. The default transfer interval is once every 30 minutes. You can also configure a secondary destination for the file to be used if, for whatever reason, the file cannot be transferred to the primary network management station.

The value of the transfer interval is also the collection period (collection interval) for the local bulk statistics file. After the collection period ends, the bulk statistics file is frozen, and a new local bulk statistics file is created for storing data. The frozen bulk statistics file is then transferred to the specified destination.

By default, the local bulk statistics file is deleted after successful transfer to an network management station.

Benefits of Periodic MIB Data Collection and Transfer

Periodic MIB data collection and transfer (bulk statistics feature) allows many of the same functions as the bulk file MIB (CISCO-BULK-FILE-MIB.my), but offers some key advantages. The main advantage is that this feature can be configured through the CLI and does not require an external monitoring application.

Periodic MIB data collection and transfer is mainly targeted for medium to high-end platforms that have sufficient local storage (volatile or permanent) to store bulk statistics files. Locally storing bulk statistics files helps minimize loss of data during temporary network outages.

This feature also has more powerful data selection features than the bulk file MIB; it allows grouping of MIB objects from different tables into data groups (object lists). It also incorporates a more flexible instance selection mechanism, where the application is not restricted to fetching an entire MIB table.

How to Configure Periodic MIB Data Collection and Transfer

Configuring a Bulk Statistics Object List

The first step in configuring the Periodic MIB Data Collection and Transfer Mechanism is to configure one or more object lists.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **snmp-server mib bulkstat object-list** *list-name*

Example:

```
snmp-server mib bulkstat object-list ifMib
```

Defines an SNMP bulk statistics object list and enters bulk statistics object list configuration mode.

Step 3 **add** {**oid** | *object-name*}

Example:

```
RP/0/RP0/CPU0:router(config-bulk-objects)# add 1.3.6.1.2.1.2.2.1.11
RP/0/RP0/CPU0:router(config-bulk-objects)# add ifAdminStatus
RP/0/RP0/CPU0:router(config-bulk-objects)# add ifDescr
```

Adds a MIB object to the bulk statistics object list. Repeat as desired until all objects to be monitored in this list are added.

Note All the objects in a bulk statistics object list have to be indexed by the same MIB index. However, the objects in the object list do not need to belong to the same MIB or MIB table.

When specifying an object name instead of an OID (using the add command), only object names with mappings shown in the **show snmp mib object** command output can be used.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.

- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring a Bulk Statistics Schema

The second step in configuring periodic MIB data collection and transfer is to configure one or more schemas.

Before you begin

The bulk statistics object list to be used in the schema must be defined.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **snmp-server mib bulkstat schema** *schema-name*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server mib
bulkstat schema intE0
RP/0/RP0/CPU0:router(config-bulk-sc)#
```

Names the bulk statistics schema and enters bulk statistics schema mode.

Step 3 **object-list** *list-name*

Example:

```
RP/0/RP0/CPU0:router(config-bulk-sc)# object-list
ifMib
```

Specifies the bulk statistics object list to be included in this schema. Specify only one object list per schema. If multiple object-list commands are executed, the earlier ones are overwritten by newer commands.

Step 4 Do one of the following:

- **instance exact** {**interface** *interface-id* [**sub-if**] | **oid** *oid*}
- **instance wild** {**interface** *interface-id* [**sub-if**] | **oid** *oid*}
- **instance range start** *oid* **end** *oid*
- **instance repetition** *oid* **max** *repeat-number*

Example:

```
RP/0/RP0/CPU0:router(config-bulk-sc)# instance
wild oid 1
```

or

```
RP/0/RP0/CPU0:router(config-bulk-sc)# instance
exact interface TenGigE 0/1.25
```

or

```
RP/0/RP0/CPU0:router(config-bulk-sc)# instance
range start 1 end 2
```

or

```
RP/0/RP0/CPU0:router(config-bulk-sc)# instance
repetition 1 max 4
```

Specifies the instance information for objects in this schema:

- The **instance exact** command indicates that the specified instance, when appended to the object list, represents the complete OID.
- The **instance wild** command indicates that all subindices of the specified OID belong to this schema. The wild keyword allows you to specify a partial, “wild carded” instance.
- The **instance range** command indicates a range of instances on which to collect data.
- The **instance repetition** command indicates data collection to repeat for a certain number of instances of a MIB object.

Note Only one **instance** command can be configured per schema. If multiple **instance** commands are executed, the earlier ones are overwritten by new commands.

Step 5 **poll-interval** *minutes*

Example:

```
RP/0/RP0/CPU0:router(config-bulk-sc)# poll-interval 10
```

Sets how often data should be collected from the object instances specified in this schema, in minutes. The default is once every 5 minutes. The valid range is from 1 to 20000.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Bulk Statistics Transfer Options

The final step in configuring periodic MIB data collection and transfer is to configure the transfer options. The collected MIB data are kept in a local file-like entity called a VFile (virtual file, referred to as a bulk statistics file in this document). This file can be transferred to a remote network management station at intervals you specify.

Before you begin

The bulk statistics object lists and bulk statistics schemas must be defined before configuring the bulk statistics transfer options.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **snmp-server mib bulkstat transfer-id** *transfer-id*

Example:

```
RP/0/RP0/CPU0:router(config)# snmp-server mib
bulkstat transfer bulkstat1
```

Identifies the transfer configuration with a name (*transfer-id* argument) and enters bulk statistics transfer configuration mode.

Step 3 **buffer-size** *bytes*

Example:

```
RP/0/RP0/CPU0:router(config-bulk-tr)# buffersize 3072
```

(Optional) Specifies the maximum size for the bulk statistics data file, in bytes. The valid range is from 1024 to 2147483647 bytes. The default buffer size is 2048 bytes.

Note If the maximum buffer size for a bulk statistics file is reached before the transfer interval time expires, all additional data received is deleted. To correct this behavior, you can decrease the polling frequency, or increase the size of the bulk statistics buffer.

Step 4 **Example:**

(Optional) Specifies the format of the bulk statistics data file (VFile). The default is schemaASCII.

Note Transfers can only be performed using schemaASCII (cdcSchemaASCII) format. SchemaASCII is a human-readable format that contains parser-friendly hints for parsing data values.

Step 5 **schema** *schema-name*

Example:

```
RP/0/RP0/CPU0:router(config-bulk-tr)# schema TenGigE 0/5/0/11/1
RP/0/RP0/CPU0:router(config-bulk-tr)# schema TenGigE/0-CAR
RP/0/RP0/CPU0:router(config-bulk-tr)# schema TenGigE 0/5/0/11/1
```

Specifies the bulk statistics schema to be transferred. Repeat this command as desired. Multiple schemas can be associated with a single transfer configuration; all collected data are placed in a single bulk data file (VFile).

Step 6 **transfer-interval** *minutes*

Example:

```
RP/0/RP0/CPU0:router(config-bulk-tr)# transfer-interval 20
```

(Optional) Specifies how often the bulk statistics file are transferred, in minutes. The default value is once every 30 minutes. The transfer interval is the same as the collection interval.

Step 7 **url primary** *url*

Example:

```
RP/0/RP0/CPU0:router(config-bulk-tr)# url primary
ftp://user:password@host/folder/bulkstat1
```

Specifies the network management system (host) that the bulk statistics data file is transferred to, and the protocol to use for transfer. The destination is specified as a Uniform Resource Locator (URL). FTP or TFTP can be used for the bulk statistics file transfer.

Step 8 **url secondary** *url*

Example:

```
RP/0/RP0/CPU0:router(config-bulk-tr)# url secondary
tftp://10.1.0.1/tftpboot/user/bulkstat1
```

(Optional) Specifies a backup transfer destination and protocol for use in the event that transfer to the primary location fails. FTP or TFTP can be used for the bulk statistics file transfer.

Step 9 **retry** *number*

Example:

```
RP/0/RP0/CPU0:router(config-bulk-tr)# retry 1
```

(Optional) Specifies the number of transmission retries. The default value is 0 (in other words, no retries). If an attempt to send the bulk statistics file fails, the system can be configured to attempt to send the file again using this command.

One retry includes an attempt first to the primary destination then, if the transmission fails, to the secondary location. For example, if the retry value is 1, an attempt is made first to the primary URL, then to the secondary URL, then to the primary URL again, then to the secondary URL again. The valid range is from 0 to 100.

If all retries fail, the next normal transfer occurs after the configured transfer-interval time.

Step 10 **retain** *minutes*

Example:

```
RP/0/RP0/CPU0:router(config-bulk-tr)# retain 60
```

(Optional) Specifies how long the bulk statistics file should be kept in system memory, in minutes, after the completion of the collection interval and a transmission attempt is made. The default value is 0. Zero (0) indicates that the file is deleted immediately after the transfer is attempted. The valid range is from 0 to 20000.

Note If the retry command is used, you should configure a retain interval larger than 0. The interval between retries is the retain interval divided by the retry number. For example, if **retain 10** and **retry 2** are configured, two retries are attempted once every 5 minutes. Therefore, if retain 0 is configured, no retries are attempted.

Step 11 **enable**

Example:

```
RP/0/RP0/CPU0:router(config-bulk-tr)# enable
```

Begins the bulk statistics data collection and transfer process for this configuration.

- For successful execution of this action, at least one schema with non-zero number of objects must be configured.
- Periodic collection and file transfer begins only if this command is configured. Conversely, the **no enable** command stops the collection process. A subsequent **enable** starts the operations again.
- Each time the collection process is started using the **enable** command, data is collected into a new bulk statistics file. When the **no enable** command is used, the transfer process for any collected data immediately begins (in other words, the existing bulk statistics file is transferred to the specified management station).

Step 12 **commit** *minutes*

Example:

```
RP/0/RP0/CPU0:router(config-bulk-tr)# retain 60
```

If the maximum buffer size for a bulk statistics file is reached before the transfer interval time expires, the transfer operation is still initiated, but any bulk statistics data received after the file was full, and before it was transferred, are deleted. To correct this behavior, you can decrease the polling frequency, or increase the size of the bulk statistics buffer.

If **retain 0** is configured, no retries are attempted. This is because the interval between retries is the retain value divided by the retry value. For example, if **retain 10** and **retry 2** are configured, retries are attempted once every 5 minutes. Therefore, if you configure the retry command, you should also configure an appropriate value for the retain command.

Periodic MIB Data Collection and Transfer: Example

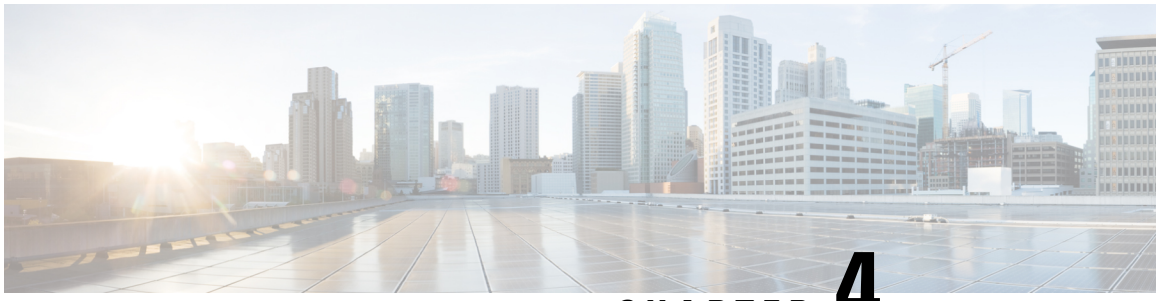
This example shows how to configure periodic MIB data collection and transfer:

```
snmp-server mib bulkstat object-list cempo
add cempMemPoolName
add cempMemPoolType
!
snmp-server mib bulkstat schema cempWild
object-list cempo
instance wild oid 8695772
poll-interval 1
!
snmp-server mib bulkstat schema cempRepeat
object-list cempo
instance repetition 8695772.1 max 4294967295
poll-interval 1
!
snmp-server mib bulkstat transfer-id cempt1
enable
url primary tftp://223.255.254.254/auto/tftp-sjc-users3/username/dumpdcm
schema cempWild
schema cempRepeat
transfer-interval 2
!
```

This example shows sample bulk statistics file content:

```
Schema-def cempt1.cempWild "%u, %s, %s, %d" Epochtime instanceoid
            1.3.6.1.4.1.9.9.221.1.1.1.1.3 1.3.6.1.4.1.9.9.221.1.1.1.1.2
cempt1.cempWild: 1339491515, 8695772.1, processor, 2
cempt1.cempWild: 1339491515, 8695772.2, reserved, 11
cempt1.cempWild: 1339491515, 8695772.3, image, 12
cempt1.cempWild: 1339491575, 8695772.1, processor, 2
cempt1.cempWild: 1339491575, 8695772.2, reserved, 11
cempt1.cempWild: 1339491575, 8695772.3, image, 12
Schema-def cempt1.cempRepeat "%u, %s, %s, %d" Epochtime instanceoid
            1.3.6.1.4.1.9.9.221.1.1.1.1.3 1.3.6.1.4.1.9.9.221.1.1.1.1.2
cempt1.cempRepeat: 1339491515, 8695772.1, processor, 2
cempt1.cempRepeat: 1339491515, 8695772.2, reserved, 11
cempt1.cempRepeat: 1339491515, 8695772.3, image, 12
```

```
cempt1.cempRepeat: 1339491515, 26932192.1, processor, 2
cempt1.cempRepeat: 1339491515, 26932192.2, reserved, 11
cempt1.cempRepeat: 1339491515, 26932192.3, image, 12
cempt1.cempRepeat: 1339491515, 35271015.1, processor, 2
cempt1.cempRepeat: 1339491515, 35271015.2, reserved, 11
cempt1.cempRepeat: 1339491515, 35271015.3, image, 12
cempt1.cempRepeat: 1339491515, 36631989.1, processor, 2
cempt1.cempRepeat: 1339491515, 36631989.2, reserved, 11
cempt1.cempRepeat: 1339491515, 36631989.3, image, 12
cempt1.cempRepeat: 1339491515, 52690955.1, processor, 2
cempt1.cempRepeat: 1339491515, 52690955.2, reserved, 11
cempt1.cempRepeat: 1339491515, 52690955.3, image, 12
```



CHAPTER 4

Configuring Cisco Discovery Protocol

Cisco Discovery Protocol (CDP) is a media- and protocol-independent protocol that runs on all Cisco-manufactured equipment including routers, bridges, access and communication servers, and switches. Using CDP, you can view information about all the Cisco devices that are directly attached to the device.

Prerequisites for Implementing CDP

To enable CDP, you must install the CDP package on your router.

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Implementing CDP

CDP is primarily used to obtain protocol addresses of neighboring devices and discover the platform of those devices. CDP can also be used to display information about the interfaces your router uses. CDP is media- and protocol-independent, and runs on all equipment manufactured by Cisco, including routers, bridges, access servers, and switches.

Use of SNMP with the CDP MIB allows network management applications to learn the device type and the SNMP agent address of neighboring devices and to send SNMP queries to those devices. CDP uses the CISCO-CDP-MIB.

CDP runs on all media that support Subnetwork Access Protocol (SNAP), including LAN, Frame Relay, and ATM physical media. CDP runs over the data link layer only. Therefore, two systems that support different network-layer protocols can learn about each other.

Each device configured for CDP sends periodic messages, known as *advertisements*, to a multicast address. Each device advertises at least one address at which it can receive SNMP messages. The advertisements also contain time-to-live, or hold-time, information, which indicates the length of time a receiving device holds

CDP information before discarding it. Each device also listens to the periodic CDP messages sent by others to learn about neighboring devices and determine when their interfaces to the media go up or down.

CDP Version-2 (CDPv2) is the most recent release of the protocol and provides more intelligent device tracking features. These features include a reporting mechanism that allows for more rapid error tracking, thereby reducing costly downtime. Reported error messages can be sent to the console or to a logging server, and can cover instances of unmatching native VLAN IDs (IEEE 802.1Q) on connecting ports, and unmatching port duplex states between connecting devices.

CDPv2 **show** commands can provide detailed output on VLAN Trunking Protocol (VTP) management domain and duplex modes of neighbor devices, CDP-related counters, and VLAN IDs of connecting ports.

Type-length-value fields (TLVs) are blocks of information embedded in CDP advertisements. This table summarizes the TLV definitions for CDP advertisements.

Table 4: Type-Length-Value Definitions for CDPv2

TLV	Definition
Device-ID TLV	Identifies the device name in the form of a character string.
Address TLV	Contains a list of network addresses of both receiving and sending devices.
Port-ID TLV	Identifies the port on which the CDP packet is sent.
Capabilities TLV	Describes the functional capability for the device in the form of a device type; for example, a switch.
Version TLV	Contains information about the software release version on which the device is running.
Platform TLV	Describes the hardware platform name of the device, for example, Cisco 4500.
VTP Management Domain TLV	Advertises the system's configured VTP management domain name-string. Used by network operators to verify VTP domain configuration in adjacent network nodes.
Native VLAN TLV	Indicates, per interface, the assumed VLAN for untagged packets on the interface. CDP learns the native VLAN for an interface. This feature is implemented only for interfaces that support the IEEE 802.1Q protocol.
Full/Half Duplex TLV	Indicates status (duplex configuration) of CDP broadcast interface. Used by network operators to diagnose connectivity problems between adjacent network elements.

How to Implement CDP on Cisco IOS XR Software

Enabling CDP

To enable CDP, you must first enable CDP globally on the router and then enable CDP on a per-interface basis. This example explains how to enable CDP globally on the router and then enable CDP on an interface.

```
Router:# configure
Router(config):# cdp
Router(config):# commit
```

```
Router:# configure
Router(config):# interface hundredGigE 0/0/0/4
Router(config-if):# cdp
Router(config-if):# commit
```

Modifying CDP Default Settings

This task explains how to modify the default version, hold-time setting, and timer settings.

Note The commands can be entered in any order.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **cdp advertise v1**

Example:

```
RP/0/RP0/CPU0:router(config)# cdp advertise v1
```

Configures CDP to use only version 1 (CDPv1) in communicating with neighboring devices.

- By default, when CDP is enabled, the router sends CDPv2 packets. CDP also sends and receives CDPv1 packets if the device with which CDP is interacting does not process CDPv2 packets.
- In this example, the router is configured to send and receive only CDPv1 packets.

Step 3 **cdp holdtime** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config)# cdp holdtime 30
```

Specifies the amount of time that the receiving networking device will hold a CDP packet sent from the router before discarding it.

- By default, when CDP is enabled, the receiving networking device holds a CDP packet for 180 seconds before discarding it.

  Note The CDP hold time must be set to a higher number of seconds than the time between CDP transmissions, which is set with the **cdp timer** command.

- In this example, the value of hold-time for the *seconds* argument is set to 30.

Step 4 **cdp timer** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config)# cdp timer 20
```

Specifies the frequency at which CDP update packets are sent.

- By default, when CDP is enabled, CDP update packets are sent at a frequency of once every 60 seconds.

  Note A lower timer setting causes CDP updates to be sent more frequently.

- In this example, CDP update packets are configured to be sent at a frequency of once every 20 seconds.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 6 (Optional) **show cdp**

Example:

```
RP/0/RP0/CPU0:router# show cdp
```

Displays global CDP information.

The output displays the CDP version running on the router, the hold time setting, and the timer setting.

Monitoring CDP

This task shows how to monitor CDP.

Note The commands can be entered in any order.

Step 1 **show cdp entry** {***** | *entry-name*} [**protocol** | **version**]

Example:

```
RP/0/RP0/CPU0:router# show cdp entry *
```

Displays information about a specific neighboring device or all neighboring devices discovered using CDP.

Step 2 **show cdp interface** [*type interface-path-id* | **location** *node-id*]

Example:

```
RP/0/RP0/CPU0:router# show cdp interface pos 0/0/0/1
```

Displays information about the interfaces on which CDP is enabled.

Step 3 **show cdp neighbors** [*type interface-path-id* | **location** *node-id*] [**detail**]

Example:

```
RP/0/RP0/CPU0:router# show cdp neighbors
```

Displays detailed information about neighboring devices discovered using CDP.

Step 4 **show cdp traffic** [**location** *node-id*]

Example:

```
RP/0/RP0/CPU0:router# show cdp traffic
```

Displays information about the traffic gathered between devices using CDP.

Examples

The following is sample output for the **show cdp neighbors** command:

```
RP/0/RP0/CPU0:router# show cdp neighbors

  Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                    S - Switch, H - Host, I - IGMP, r - Repeater

Device ID       Local Intrfce    Holdtme Capability Platform  Port ID
NCS5500         Hu0/0/0/4        15      R          NCS-5500  Hu0/0/0/4
```

The following is sample output for the **show cdp neighbors** command. In this example, the optional *type instance* arguments are used in conjunction with the **detail** optional keyword to display detailed information about a CDP neighbor. The output includes information on both IPv4 and IPv6 addresses.

```
RP/0/RP0/CPU0:router# show cdp neighbors hundredGigE 0/0/0/4 detail

  -------------------------
Device ID: NCS5500
SysName : NCS5500
Entry address(es):
```

```
  IPv4 address: 40.0.0.2
  IPv6 address: 10:10:10:10::1
Platform: cisco NCS-5500,  Capabilities: Router
Interface: HundredGigE0/0/0/4
Port ID (outgoing port): HundredGigE0/0/0/4
Holdtime : 13 sec

Version :
7.1.1.112I

advertisement version: 2
Duplex: full
```

The following is sample output for the **show cdp entry** command. In this example, the optional *entry* argument is used to display entry information related to a specific CDP neighbor.

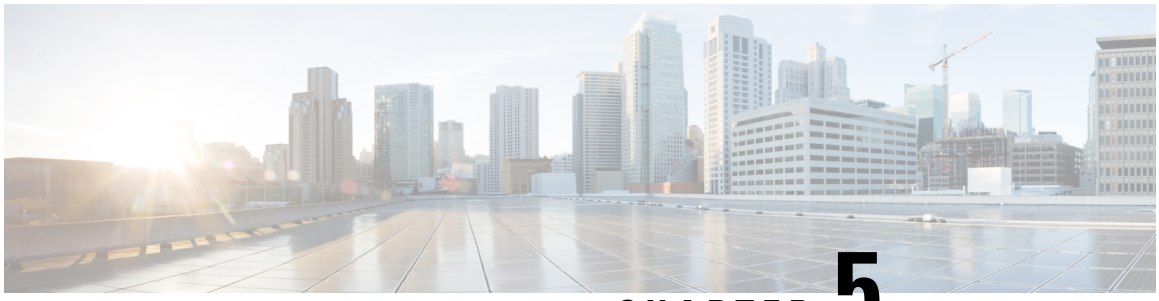
```
RP/0/RP0/CPU0:router# show cdp entry NCS5500
 -------------------------
Device ID: NCS5500
SysName : NCS5500
Entry address(es):
  IPv4 address: 40.0.0.2
  IPv6 address: 10:10:10:10::1
Platform: cisco NCS-5500,  Capabilities: Router
Interface: HundredGigE0/0/0/4
Port ID (outgoing port): HundredGigE0/0/0/4
Holdtime : 11 sec

Version :
7.1.1.112I

advertisement version: 2
Duplex: full
```

The following is sample output for the **show cdp interface** command. In this example, CDP information related to interface 0/0/0/4 is displayed.

```
RP/0/RP0/CPU0:router# show cdp interface hundredGigE 0/0/0/4


  HundredGigE0/0/0/4 is Up
  Encapsulation ether
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
```

The following is sample output for the **show cdp traffic** command:

```
RP/0/RP0/CPU0:router# show cdp traffic

 CDP counters :
        Packets output: 10, Input: 39
        Hdr syntax: 0, Chksum error: 0, Encaps failed: 0
        No memory: 0, Invalid packet: 0, Truncated: 0
        CDP version 1 advertisements output: 0, Input: 0
        CDP version 2 advertisements output: 10, Input: 39
        Unrecognize Hdr version: 0, File open failed: 0
```



CHAPTER 5

Configuring Smart Licensing

This module describes the configuration related to the Smart Licensing.

Table 5: Feature History for Smart License

Release	Modification
Release 7.0.11	Smart Licensing was introduced

This model contains the following topics:

What is Smart Licensing?

Smart Licensing is a cloud-based, flexible software licensing model that enables you to activate and manage Cisco software licenses across their organization. Smart Licensing solution allows you to easily track the status of your license and software usage trends. Cisco Smart Licensing establishes a pool of licenses or entitlements that can be used across the entire organization in a flexible and automated manner. Smart Licensing helps simplify four core functions:

- **Purchase**—Creates a Smart Account (and optionally, your Virtual Account). Licenses are added to your Smart Account and are immediately available for use.
- **Install**—Register your product with your Smart Account using an account-based Registration Token. Thereafter, the entire process is automatic. Product Activation Keys (PAKs) and license files are no longer needed.
- **Management**—Make changes to license consumption by updating your configuration; any license change is automatically reflected in your Smart Account. You can share licenses in your Virtual Account through the license pooling option. License pools (logical grouping of licenses) can reflect your organization structure. Smart Licensing solution also offers Cisco Smart Software Manager, a centralized portal that enables you to manage all your Cisco software licenses from one centralized website.

- **Visibility and Asset Management**—Cisco Smart Software Manager (CSSM) portal offers an integrated view of the licenses you own and have deployed. You can use this data to make better purchase decisions, based on your consumption.

Note

- Smart Licensing is enabled by default.
- Only Flexible Consumption Model smart licenses are supported.

For more information on Smart Licensing and related documentation, see https://www.cisco.com/c/en_in/products/software/smart-accounts/software-licensing.html.

What is Flexible Consumption Model?

The Flexible Consumption Model (FCM) provides the capability and flexibility to purchase software capacity as needed. FCM delivers the following:

- Pay-as-you-grow—Enables you to lower initial costs and add more capacity over time.
- Simplify operations—FCM delivers the carrier-class IOS-XR software feature set with two software suites, Essentials and Advanced, that simplify license management.
- Utilize capital efficiently—License pooling enables an efficient way to share licenses across the network.

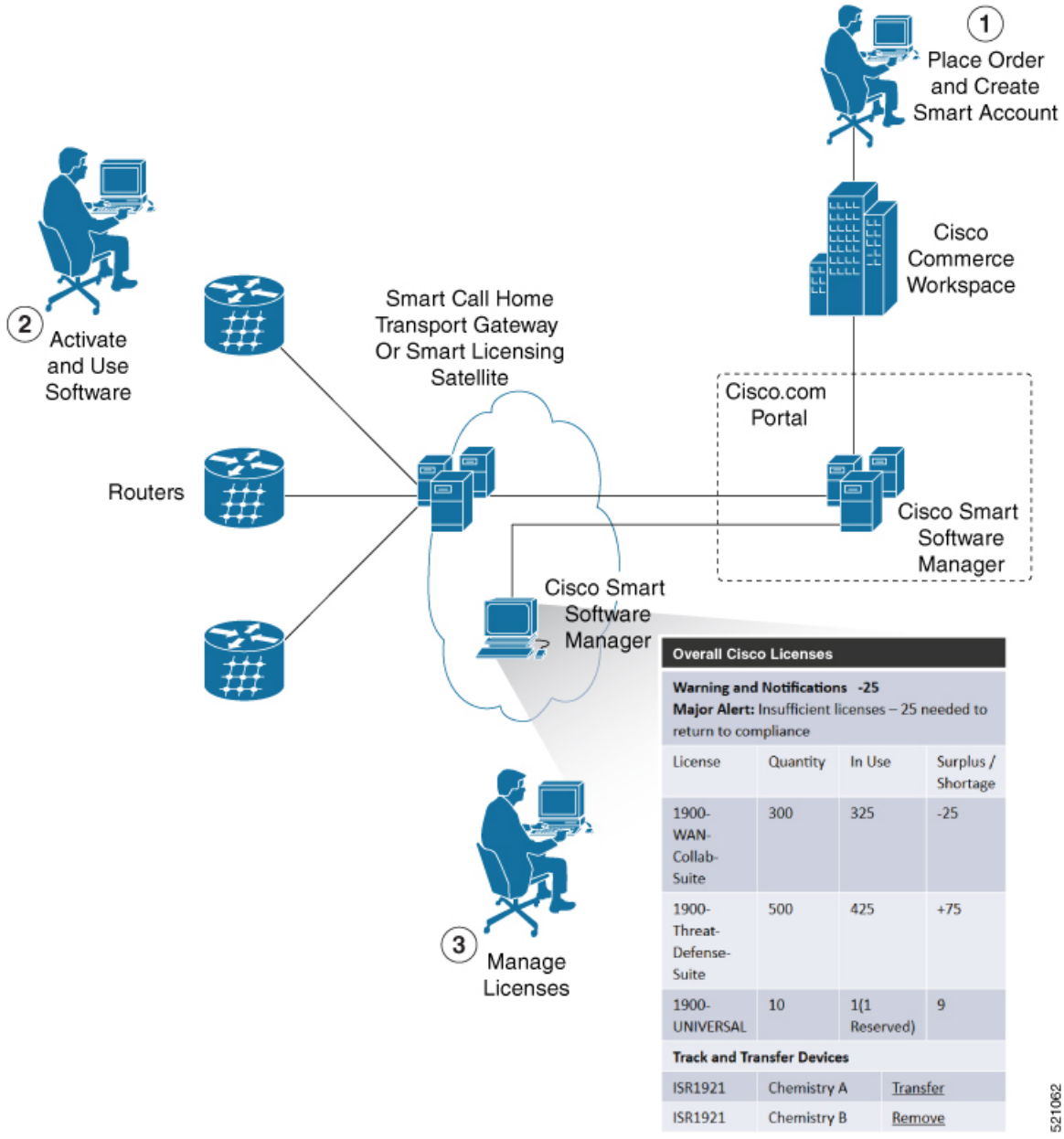
The following table provides information about FCM licenses for Cisco 8000 Series routers:

License Name	Hardware Supported	Consumption Pattern
Essential and Advanced Licenses: • ESS-CA-400G-RTU-2 • ESS-CA-100G-RTU-2 • ADV-CA-400G-RTU-2 • ADV-CA-100G-RTU-2	Fixed port router: Cisco 8201 Router Modular port router: Cisco 8812 Router	The number of essential or advanced licenses consumed depends on the number of active ports and is reported on per chassis basis.

License Name	Hardware Supported	Consumption Pattern
Hardware Tracking Licenses that support chassis • 8201-TRK • 8812-TRK • 8808-TRK • 8818-TRK • 8202-TRK • 8804-TRK • 8800-LC-48H-TRK • 8800-LC-36FH-TRK	These Tracking licenses are named on the basis of the hardware supported. For example, 8201-TRK licenses support Cisco 8201 Router.	The number of licenses consumed depends on the number of line cards in use.

How Does Smart Licensing Work?

Figure 4: Smart Licensing - Workflow



1. Place Order and Create Smart Account—You must have a Smart Account to set up Smart Licensing.
   a. Go to https://software.cisco.com/.
   b. Under the **Administration** section, click **Get a Smart Account or Request Access to an Existing Smart Account**.
   c. Verify or enter your Cisco.com profile details to complete creating a Smart Account.

2. Activate and Use Software—Register your product. For more information, see the *Registering your Router* section. After you enable Smart Licensing, you can use either of the following options to communicate with the CSSM:
   - Smart Call Home—The Smart Call Home feature is automatically configured after the Smart Licensing is enabled. Smart Call Home is used by Smart Licensing as a medium for communication with the CSSM. You can use this feature to page a network support engineer, email a Network Operations Center, or use Cisco Smart Call Home services to generate a case with the Technical Assistance Center. The Call Home feature can deliver alert messages containing information about diagnostics and environmental faults and events. For more information on Smart Call Home feature, see the Smart Call Home Deployment Guide.
   - Smart Licensing CSSM On-Prem—The Smart licensing on-premise option provides an on-premises collector that can be used to consolidate and manage Smart license usage, as well as facilitate communications back to the CSSM at Cisco.com.
3. Manage Licenses—You can manage and view reports about your overall license usage in the Smart Software Manager portal.

What is Cisco Smart Software Manager?

Cisco Smart Software Manager enables you to manage all of your Cisco Smart software licenses from one centralized website. With Cisco Smart Software Manager, you organize and view your licenses in groups called virtual accounts (collections of licenses and product instances). The Cisco Smart Software Manager allows you to:

- Create, manage, or view virtual accounts
- Create and manage Product Instance Registration Tokens
- Transfer licenses between virtual accounts or view licenses
- Transfer, remove, or view product instances
- Run reports against your virtual accounts
- Modify your email notification settings
- View overall account information

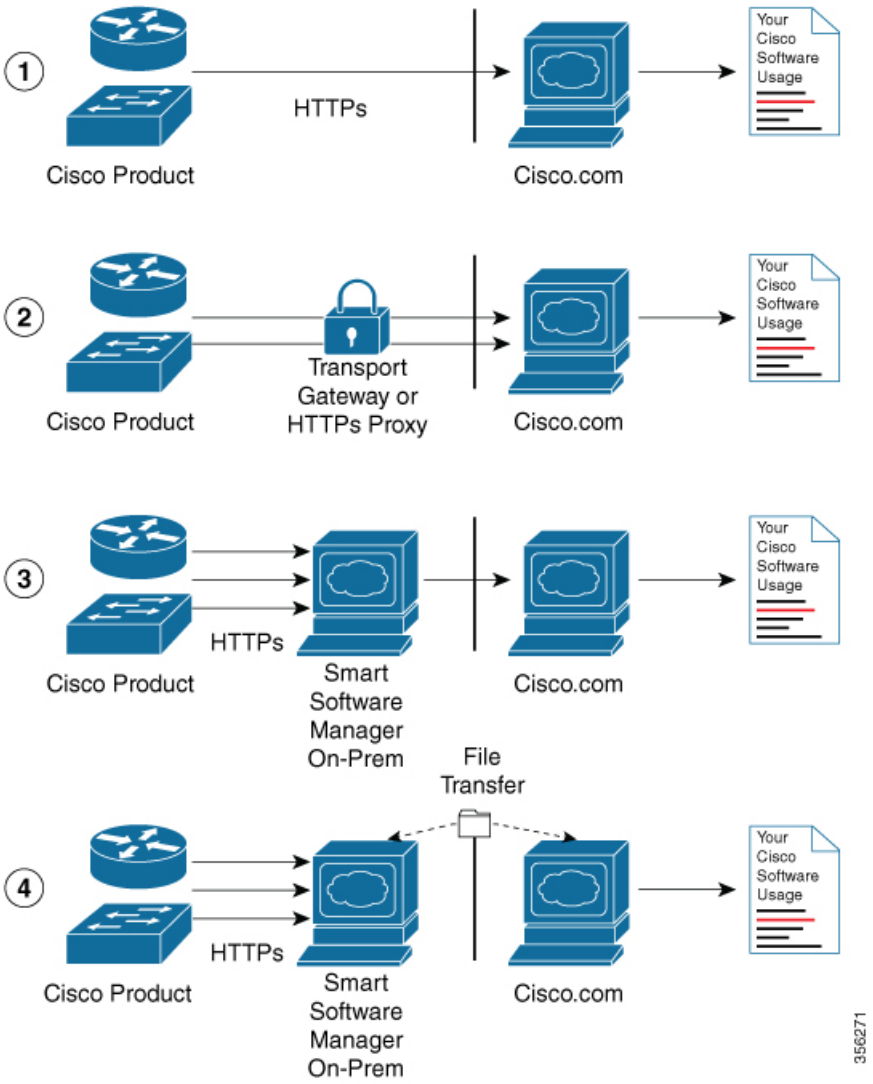
To access the Cisco Smart Software Manager:

- Go to https://software.cisco.com.
- Under the **License** section, click **Smart Software Licensing**.

Smart Licensing Deployment Options

The following illustration shows the various options available for deploying Smart Licensing:

Figure 5: Smart Licensing Deployment Options



1. Direct cloud access—In this method, Cisco products send usage information directly over the internet to CSSM on http://www.cisco.com; no additional components are needed for deployment.
2. Direct cloud access through an HTTPs proxy—In direct cloud access through an HTTPs proxy deployment method, Cisco products send usage information over the internet through a proxy server—either a Smart Call Home Transport Gateway or off-the-shelf Proxy (such as Apache) to CSSM on http://www.cisco.com.
3. Mediated access through an on-premises collector-connected—In mediated access through an on-premises collector-connected deployment method, Cisco products send usage information to a locally connected collector, which acts as a local license authority. Periodically, the information is exchanged to keep the databases in synchronization.
4. Mediated access through an on-premises collector-disconnected—In the mediated access through an on-premises collector-disconnected deployment method, Cisco products send usage information to a local disconnected collector, which acts as a local license authority. Exchange of human-readable information is performed occasionally (once a month) to keep the databases in synchronization.

Options 1 and 2 provide easy deployment options, whereas options 3 and 4 provide secure environment deployment options.

Note Smart Software On-Premise provides support for options 3 and 4.

The communication between Cisco devices and CSSM is facilitated by the Smart Call Home software.

Configuring Smart Licensing

Prerequisites for Configuring Smart Licensing

Ensure that you have completed the following activities on Cisco Smart Software Manager:

- Set up a Cisco Smart Account. For more information, see the *How Smart Licensing Works* section in this document.
- Set up Virtual Account or accounts. For more information, see the *Virtual Accounts* section in the Smart Software Manager Help.
- Create user roles in the **Users** tab in the **Manage Smart Account** page. Provide the appropriate user access rights.
- Accept the Smart Software Licensing Agreement on Cisco Smart Software Manager to register your router.
- Have a layer 3 connection set up on your router.
- Configure a valid DNS and proper time on the router to connect CSSM or CSSM On-Prem.

Setting up the Router for Smart Licensing

Table 6: Three-step Roadmap to Set up the Router for Smart Licensing

Activity	Communication Connection Options		
Step 1—Configure Communications	See the *Configuring a Direct Cloud Connection* section.	See the *Configuring a Connection through a HTTP Proxy* section.	See the *Connecting to CSSM On-Premise* section.
Step 2—Register and Activate	See the *Registering and Activating your Router* section.		
Step 3—Verify the Configuration	See the *Verifying your Smart Licensing Configuration* section.		

Configuring a Communications Connection Between the Router and Cisco Smart Software Manager

Configuring a Direct Cloud Connection

In this deployment option, the **configure call-home profile** is configured by default. Use the **show call-home profile all** command to check the profile status.

Call Home service provides email-based and web-based notification of critical system events to Cisco Smart Software Manager.

To configure and enable Call Home service:

SUMMARY STEPS

1. **configure terminal**
2. **call-home**
3. **service active**
4. **contact-email-addr** *email-address*
5. **profile CiscoTAC-1**
6. **destination transport-method http**
7. **destination address http** *url*
8. **active**
9. **no destination transport-method email**
10. **commit**
11. **exit**
12. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 2	**call-home** **Example:** `Router(config)# call-home`	Enters Call Home configuration mode.
Step 3	**service active** **Example:** `Router(config-call-home)# service active`	Activates Call Home service.
Step 4	**contact-email-addr** *email-address* **Example:** `Router(config-call-home)# contact-email-addr username@example.com`	Assigns the provided email address. You can enter up to 200 characters in email address format. **Note** Spaces are not allowed in the email address.

	Command or Action	Purpose
Step 5	**profile CiscoTAC-1** **Example:** `Router(config-call-home)# profile CiscoTAC-1`	Enables the CiscoTAC-1 profile to be used with the Call Home service. By default, the CiscoTAC-1 profile is disabled.
Step 6	**destination transport-method http** **Example:** `Router(config-call-home-profile)# destination transport-method http`	Enables the Call Home service through an HTTP connection.
Step 7	**destination address http** *url* **Example:** `Router(config-call-home-profile)# destination address http https://tools.cisco.com/its/service/oddce/services/DDCEService`	Connects the router to the Cisco Smart Software Manager.
Step 8	**active** **Example:** `Router(config-call-home-profile)# active`	Enables the destination profile.
Step 9	**no destination transport-method email** **Example:** `Router(config-call-home-profile)# no destination transport-method email`	Disables the email option for the Call Home service.
Step 10	**commit** **Example:** `Router(config-call-home-profile)# commit`	Commits the configuration.
Step 11	**exit** **Example:** `Router(config-call-home-profile)# exit`	Exits the Call Home destination profile configuration mode and returns to the Call Home configuration mode.

Step 12	**exit**
	Example:
	`Router(config-call-home)# exit`
	`Router(config)#`
		Exits the Call Home configuration mode and returns to the global configuration mode.

Configuring a Connection Through an HTTP Proxy

The Call Home service can be configured through an HTTPs proxy server.

SUMMARY STEPS

1. **configure terminal**
2. **call-home**
3. **service active**

4. **contact-email-address** *email-address*
5. **http-proxy** *proxy-address* **port** *port-number*
6. **profile CiscoTAC-1**
7. **no destination transport-method email**
8. **exit**
9. **profile** *profile-name*
10. **reporting smart-licensing-data**
11. **destination transport-method http**
12. **destination address http** *url*
13. **active**
14. **exit**
15. **exit**
16. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 2	**call-home** **Example:** `Router(config)# call-home`	Enters Call Home configuration mode.
Step 3	**service active** **Example:** `Router(config-call-home)# service active`	Enables the Call Home feature.
Step 4	**contact-email-address** *email-address* **Example:** `Router(config-call-home)# contact-email-addr sch-smart-licensing@cisco.com`	Configures the default email address.
Step 5	**http-proxy** *proxy-address* **port** *port-number* **Example:** `Router(config-call-home)# http-proxy 198.51.100.10 port 3128`	Provides the proxy server information to the Call Home service.
Step 6	**profile CiscoTAC-1** **Example:** `Router(config-call-home)# profile CiscoTAC-1`	Enables the CiscoTAC-1 profile to be used with the Call Home service. By default, the CiscoTAC-1 profile is disabled.
Step 7	**no destination transport-method email** **Example:**	Disables the email option for the Call Home service.

	Command or Action	Purpose
	`Router(config-call-home-profile)# no destination transport-method email`	

Step 8	**exit**
	Example:
	`Router(config-call-home-profile)# exit`
	`Router(config-call-home)#`
		Exits the Call Home destination profile configuration mode and returns to the Call Home configuration mode.

Step 9	**profile** *profile-name* **Example:** `Router(config-call-home)# profile test1`	Enters the Call Home destination profile configuration mode for the specified destination profile name. If the specified destination profile does not exist, it is created.
Step 10	**reporting smart-licensing-data** **Example:** `Router(config-call-home-profile)# reporting smart-licensing-data`	Enables data sharing with the Call Home service through the configured transport method, in this case, HTTP.
Step 11	**destination transport-method http** **Example:** `Router(config-call-home-profile)# destination transport-method http`	Enables the HTTP message transport method.
Step 12	**destination address http** *url* **Example:** `Router(config-call-home-profile)# destination address http https://tools.cisco.com/its/service/oddce/services/DDCEService`	Connects the router to the Cisco Smart Software Manager.
Step 13	**active** **Example:** `Router(config-call-home-profile)# active`	Enables the destination profile.
Step 14	**exit** **Example:** `Router(config-call-home-profile)# exit`	Exits the Call Home destination profile configuration mode and returns to the Call Home configuration mode.

Step 15	**exit**
	Example:
	`Router(config-call-home)# exit`
	`Router(config)#`
		Exits the Call Home configuration mode and returns to the global configuration mode.

Step 16	**commit** **Example:** `Router(config)# commit`	Commits the configuration.

Connecting to CSSM On-Premise

This section describes how to configure the Call Home service for on-premise smart software through connected or disconnected mode.

SUMMARY STEPS

1. **configure terminal**
2. **call-home**
3. **profile** *profile-name*
4. **reporting smart-licensing-data**
5. **destination transport-method http**
6. **destination address http** *url*
7. **no destination address http** *url*
8. **destination preferred-msg-format** {**long-text** | **short-text** | **xml**}
9. **active**
10. **exit**
11. **exit**
12. **http client source-interface** *ip-version interface-type interface-number*
13. **crypto ca trustpoint** *name*
14. **commit**
15. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 2	**call-home** **Example:** `Router(config)# call-home`	Enters Call Home configuration mode.
Step 3	**profile** *profile-name* **Example:** `Router(config-call-home)# profile test1`	Enters the Call Home destination profile configuration mode for the specified destination profile name. If the specified destination profile does not exist, it is created.
Step 4	**reporting smart-licensing-data** **Example:** `Router(config-call-home-profile)# reporting smart-licensing-data`	Enables data sharing with the Call Home service through the configured transport method, in this case, HTTP.
Step 5	**destination transport-method http** **Example:** `Router(config-call-home-profile)# destination transport-method http`	Enables the HTTP message transport method.

	Command or Action	Purpose

Step 6	**destination address http** *url*
	Example:
	`Router(config-call-home-profile)# destination address http http://209.165.201.15/Transportgateway/services/RouterRequestHandler`
	
	`Or`
	`Router(config-call-home-profile)# destination address http https://209.165.201.15/Transportgateway/services/RouterRequestHandler`
		Configures the destination URL (CSSM) to which Call Home messages are sent.
		Note Ensure the IP address or the fully qualified domain name (FQDN) in the destination URL matches the IP address or the FQDN as configured for the **Host Name** on the CSSM On-Prem.

Step 7	**no destination address http** *url* **Example:** `Router(config-call-home-profile)# no destination address http https://tools.cisco.com/its/service/oddce/services/DDCEService`	Removes the default destination address.
Step 8	**destination preferred-msg-format** {**long-text** | **short-text** | **xml**} **Example:** `Router(config-call-home-profile)# destination preferred-msg-format xml`	(Optional) Configures a preferred message format. The default message format is XML.
Step 9	**active** **Example:** `Router(config-call-home-profile)# active`	Enables the destination profile.
Step 10	**exit** **Example:** `Router(config-call-home-profile)# exit`	Exits the Call Home destination profile configuration mode and returns to the Call Home configuration mode.

Step 11	**exit**
	Example:
	`Router(config-call-home)# exit`
	`Router(config)#`
		Exits the Call Home configuration mode and returns to the global configuration mode.

Step 12	**http client source-interface** *ip-version interface-type interface-number* **Example:** `Router(config)# http client source-interface ipv4 Vlan100`	Configures a source interface for the HTTP client. **Note** This command is mandatory for a VRF interface.
Step 13	**crypto ca trustpoint** *name* **Example:**	(Optional) Declares the trustpoint and its name.

	Command or Action	Purpose
	`Router(config)# crypto ca trustpoint SLA-TrustPoint` `Router(config-trustp)#`	
Step 14	**commit** **Example:** `Router(config-trustp)# commit`	Commits the configuration.

Step 15	**end**
	Example:
	`Router(config-trustp)# end`
	`Router(config)#`
		Returns to the global configuration mode.

Installing CSSM On-Premise

For information on installation instructions, see the Smart Software Manager On-Prem Installation Guide.

Registering and Activating Your Router

Product registration securely associates a device with the Smart Account and the Virtual Account of your choice. It also establishes trust between the end product and the CSSM. Tokens are used to register a product with the appropriate Virtual Account on CSSM Cloud (on Cisco.com) or CSSM On-Premise.

A Registration Token:

- Can be either used once or reused multiple times. You can set a limit to the number of times a token can be reused when you create the token.
- Can be created and revoked at any time.
- Expires after a period of time (default is 30 days; minimum is one day; maximum is 365 days)

A Registration Token is not:

- Product specific: The same Registration Token can be used on different product types.
- A license, key, or PAK.
- Stored on the Cisco device and they are not persistent.
- Required after the product is registered. Token expiration has no effect on previously registered products; it simply means that that token can no longer be used to register a new product.

Generating a New Token from CSSM

Step 1 If you choose the direct cloud access deployment option, log in to CSSM from https://software.cisco.com/#.

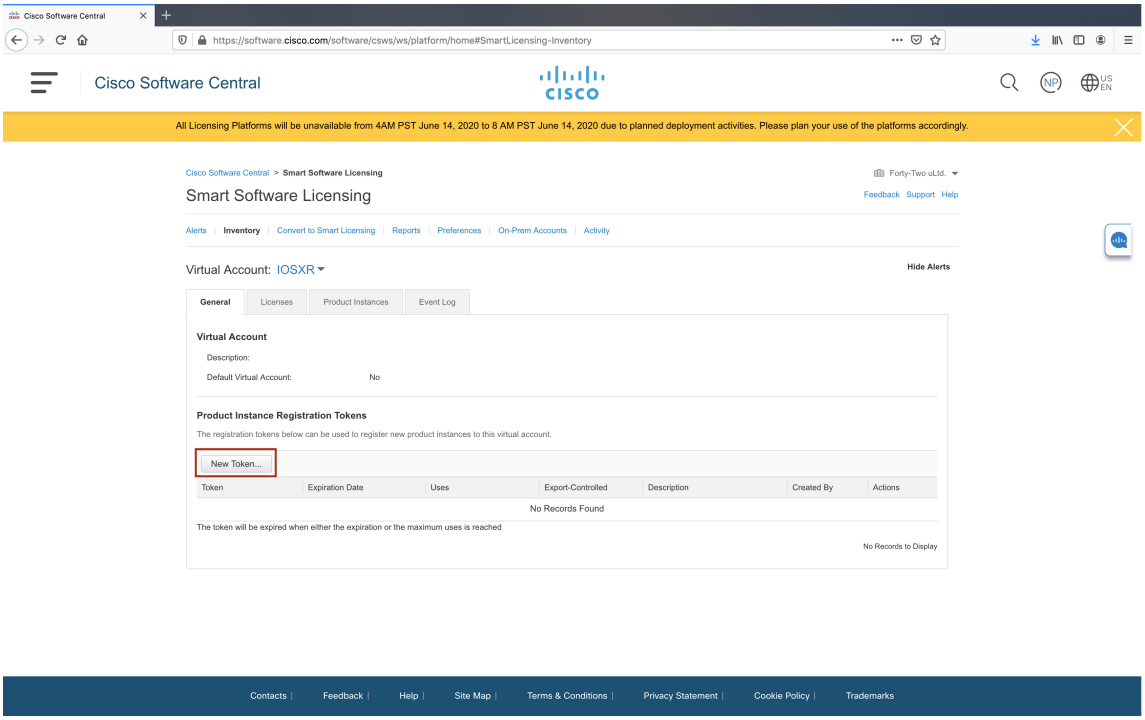
If you chose the mediated access deployment option, log in to CSSM On-Prem from *https://<on-prem-ip-address>:8443*.

Step 2 Select the **Inventory** tab.

Step 3 From the Virtual Account drop-down list, choose the virtual account to which you want to register your product.

Step 4 Select the **General** tab.

Step 5 Click **New Token**.



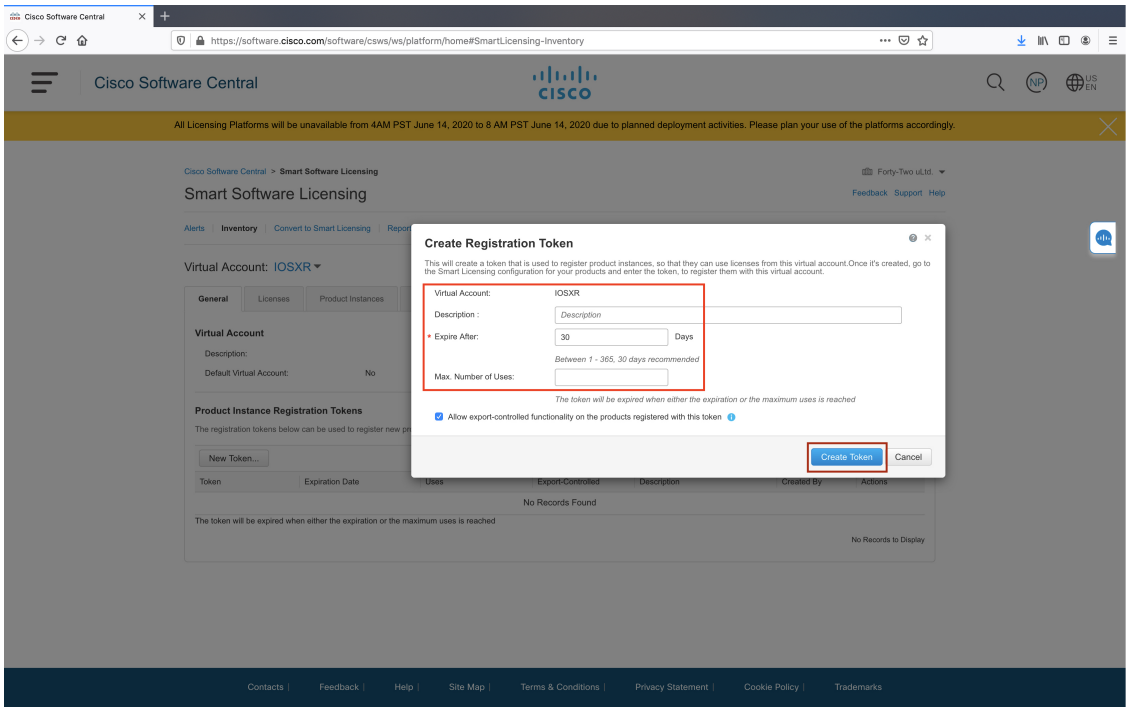
The **Create Registration Token** window is displayed.

Step 6 In the **Description** field, enter the token description.

In the **Expire After** field, enter the number of days the token must be active. The default value is 30 days.

In the **Max. Number of Uses** field, enter the maximum number of uses allowed after which the token expires.

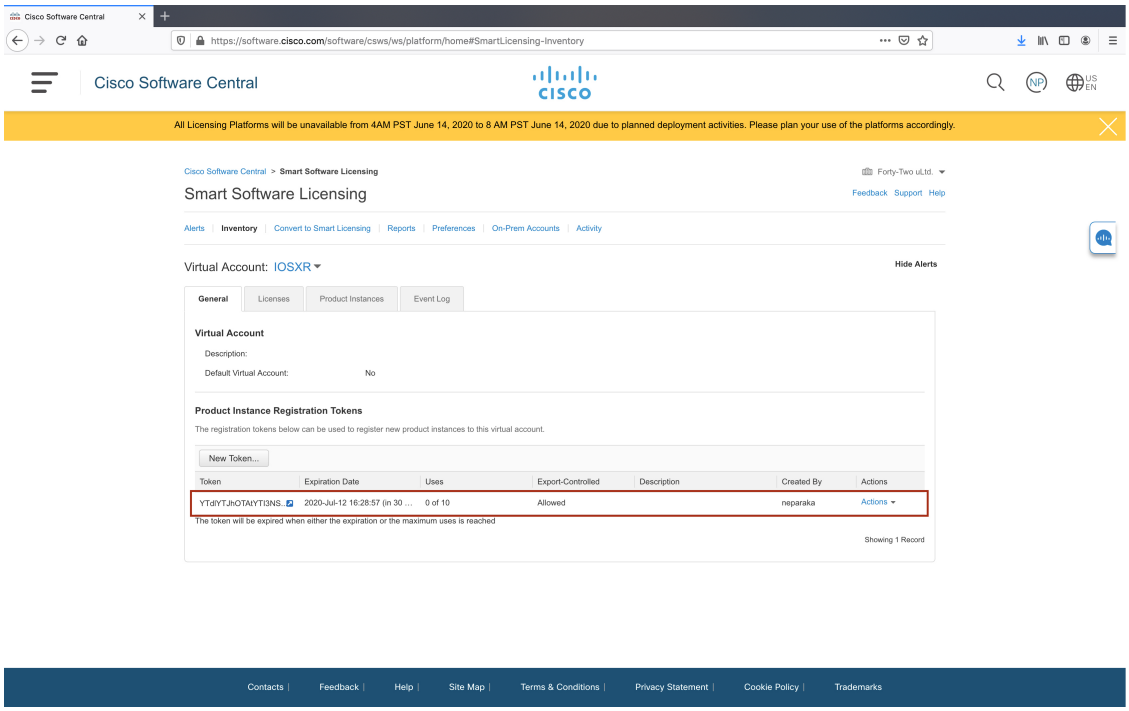
Select the **Allow export-controlled functionality on the products registered with this token** checkbox to ensure Cisco compliance with US and country-specific export policies and guidelines. For more information, see https://www.cisco.com/c/en/us/about/legal/global-export-trade.html.



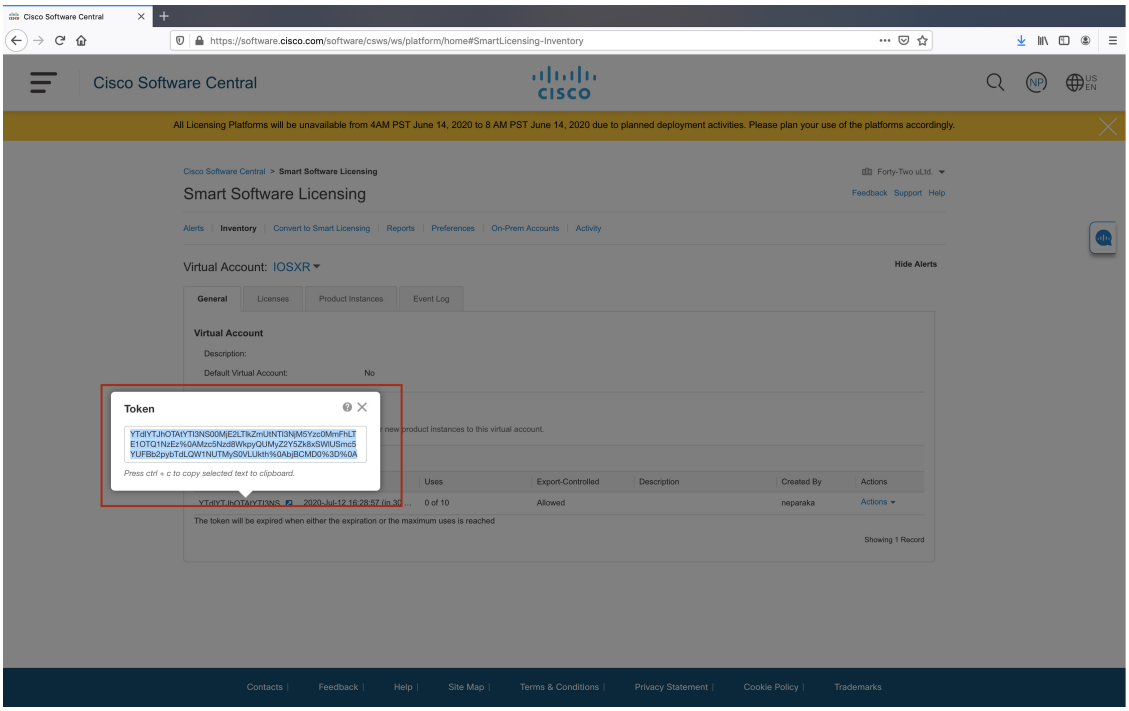
521051

Click **Create Token**.

Step 7 After the token is created, select and copy the token to a text file.



521052



521053

You need this token to register your router.

What to do next

See the *Registering Your Device With the Token* section.

Registering Your Device With the Token

SUMMARY STEPS

1. **license smart register idtoken** *token-ID*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**license smart register idtoken** *token-ID* **Example:** `license smart register idtoken $Tl4UytrNXBzbEs1ck8veUtWaG5abnZJOFdDa1FwbVRa%0AblRMoz0%3D%0A`	Registers Smart Licensing on the router using the registration token created in the CSSM. On successful registration, the product instance is created in the CSSM virtual account and its license usage is displayed on the CSSM.

Renewing Your Smart Licensing Registration

Your registration is automatically renewed every six months. To find the status of the license, use the **license smart renew auth** command.

As long as the license is in an 'Authorized' or 'Out-of-compliance' (OOC) state, the authorization period is renewed. Grace period starts when an authorization period expires. During the grace period or when the grace period is in the 'Expired' state, the system continues to try to renew the authorization period. If a retry is successful, a new authorization period starts.

Note If the smart license renewal fails, then the product instance goes to an unidentified state and starts consuming the evaluation period.

Before you begin

Ensure that the following conditions are met to renew your smart license:

- Smart licensing is enabled.
- The router is registered.

SUMMARY STEPS

1. **license smart renew** {**auth** | **id**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**license smart renew** {**auth** | **id**} **Example:** `Router# license smart renew auth`	Renews your token ID or authorization with Cisco smart licensing.

Deregistering Your Router from CSSM

When a router is taken off the inventory, shipped elsewhere for redeployment, or returned to Cisco for replacement, you can deregister that router.

Before you begin

Ensure that a Layer 3 connection to CSSM is available to successfully deregister the device.

SUMMARY STEPS

1. **license smart deregister**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**license smart deregister** **Example:** `Router# license smart deregister`	Cancels the registration of the router and sends the router into evaluation mode. All smart licensing entitlements and certificates on the corresponding platform are removed. The product instance of the router stored on CSSM is also removed.

Verifying the Smart Licensing Configuration

Use the following **show** commands to verify the default Smart Licensing configuration. If any issue is detected, take corrective action before making further configurations.

SUMMARY STEPS

1. **show license status**
2. **show license all**
3. **show license status**
4. **show license udi**
5. **show license summary**
6. **show license platform summary**
7. **show license platform detail**
8. **show call-home smart-licensing statistics**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show license status** **Example:** `Router# show license status`	Displays the compliance status of Smart Licensing. Following are the possible status: • **Waiting**—Indicates that the initial state after your device has made a license entitlement request. The device establishes communication with Cisco and successfully registers itself with the Cisco license manager. • **Authorized**—Indicates that your device is able to communicate with the Cisco license manager, and is authorized to initiate requests for license entitlements. • **Out-Of-Compliance**—Indicates that one or more of your licenses are out-of-compliance. Buy more licenses, or renew the existing licenses. • **Eval Period**—Indicates that Smart Licensing is consuming the evaluation period. Register the device with the Cisco Licensing manager, else your license expires.

	Command or Action	Purpose
		Note Repetitive 'Smart Licensing evaluation expired' warning messages are displayed on the console every hour, but there is no functionality impact on the device. To stop these repetitive messages, register the device again with new a registration token. • **Disabled**—Indicates that Smart Licensing is disabled. • **Invalid**—Indicates that Cisco does not recognize the entitlement tag as the tag is not in the database.
Step 2	**show license all** **Example:** `Router# show license all`	Displays all entitlements in use. The output also displays the associated licensing certificates, compliance status, Unique Device Identifier (UDI), and other details.
Step 3	**show license status** **Example:** `Router# show license status`	Displays the status of all entitlements in use.
Step 4	**show license udi** **Example:** `Router# show license udi`	Displays the Unique Device Identifier (UDI) information.
Step 5	**show license summary** **Example:** `Router# show license summary`	Displays a summary of all entitlements in use.
Step 6	**show license platform summary** **Example:** `Router# show license platform summary`	Displays the registration status and provides detailed information about the essential, advanced, and tracking license consumption in generic or vortex license model.
Step 7	**show license platform detail** **Example:** `Router# show license platform detail`	Displays detailed information about: • Licenses that can be consumed on a platform in both, generic and vortex models • The active model, whether generic or vortex model • The current count and the next consumption count of a license
Step 8	**show call-home smart-licensing statistics** **Example:** `Router# show call-home smart-licensing statistics`	Displays statistics of the communication between the Smart Licensing manager and the Cisco back-end using Smart Call Home. **Note** If the communication fails or drops, check your call home configuration for any errors.

Smart Licensing Configuration Examples

Example: Viewing the Call Home Profile

To display the **http Call Home profile** or the **On-Prem Call Home profile**, use the **show call-home profile all** command.

```
Router# show call-home profile all
Fri Sep 11 23:26:22.571 UTC
Profile Name: CiscoTAC-1
    Profile status: ACTIVE
    Profile mode: Full Reporting
    Reporting Data: Smart Call Home, Smart Licensing
    Preferred Message Format: xml
    Message Size Limit: 3145728 Bytes
    Transport Method: http
    HTTP  address(es): https://tools.cisco.com/its/service/oddce/services/DDCEService
    Other address(es): default

    Periodic configuration info message is scheduled every 3 day of the month at 11:25

    Periodic inventory info message is scheduled every 3 day of the month at 11:10

    Alert-group               Severity
    ------------------------  ------------
    inventory                 normal

    Syslog-Pattern            Severity
    ------------------------  ------------
    .*                        critical


Router# show call-home profile all
Sat Sep 12 00:04:12.514 UTC
Profile Name: CiscoTAC-1
    Profile status: ACTIVE
    Profile mode: Full Reporting
    Reporting Data: Smart Call Home, Smart Licensing
    Preferred Message Format: xml
    Message Size Limit: 3145728 Bytes
    Transport Method: http
    HTTP  address(es): http://10.30.110.38/Transportgateway/services/DeviceRequestHandler
    Other address(es): default
    Periodic configuration info message is scheduled every 3 day of the month at 11:25
    Periodic inventory info message is scheduled every 3 day of the month at 11:10
    Alert-group               Severity
    ------------------------  ------------
    inventory                 normal

    Syslog-Pattern            Severity
    ------------------------  ------------
    .*                        critical
```

Example: Viewing License Information Before Registration

To display the license entitlements, use the **show license all** command:

```
Router# show license all
Smart Licensing Status
======================
Smart Licensing is ENABLED
```

```
Registration:
  Status: UNREGISTERED
  Export-Controlled Functionality: NOT ALLOWED

License Authorization:
  Status: EVAL MODE
  Evaluation Period Remaining: 89 days, 2 hours, 53 minutes, 27 seconds

Export Authorization Key:
  Features Authorized:
    <none>

Utility:
  Status: DISABLED

Data Privacy:
  Sending Hostname: yes
    Callhome hostname privacy: DISABLED
    Smart Licensing hostname privacy: DISABLED
  Version privacy: DISABLED

Transport:
  Type: Callhome

License Usage
==============
(8808-TRK):
  Description:
  Count: 1
  Version: 1.0
  Status: EVAL MODE
  Export status: NOT RESTRICTED
(XR-8K-7.0-TRK):
  Description:
  Count: 1
  Version: 1.0
  Status: EVAL MODE
  Export status: NOT RESTRICTED
(8800-LC-48H-TRK):
  Description:
  Count: 1
  Version: 1.0
  Status: EVAL MODE
  Export status: NOT RESTRICTED

Product Information
===================
UDI: PID:8808,SN:FOX224PJKHQ
Agent Version
Smart Agent for Licensing: 4.9.6_rel/41

Reservation Info
================
License reservation: DISABLED
```

To display the license usage information, use the **show license usage** command:

```
Router# show license usage

License Authorization:
  Status: EVAL MODE
  Evaluation Period Remaining: 89 days, 2 hours, 49 minutes, 40 seconds

(8808-TRK):
```

```
  Description:
  Count: 1
  Version: 1.0
  Status: EVAL MODE
  Export status: NOT RESTRICTED

(XR-8K-7.0-TRK):
  Description:
  Count: 1
  Version: 1.0
  Status: EVAL MODE
  Export status: NOT RESTRICTED

(8800-LC-48H-TRK):
  Description:
  Count: 1
  Version: 1.0
  Status: EVAL MODE
  Export status: NOT RESTRICTED
```

To display all the license summaries, use the **show license summary** command:

```
Router# show license summary
Smart Licensing is ENABLED

Registration:
  Status: UNREGISTERED
  Export-Controlled Functionality: NOT ALLOWED

License Authorization:
  Status: EVAL MODE
  Evaluation Period Remaining: 89 days, 2 hours, 46 minutes, 55 seconds

License Usage:
  License                 Entitlement tag               Count Status
  -----------------------------------------------------------------------------
                          (8808-TRK)                        1 EVAL MODE
                          (XR-8K-7.0-TRK)                   1 EVAL MODE
                          (8800-LC-48H-TRK)                 1 EVAL MODE
```

To display the license status information, use the **show license status** command:

```
Router# show license status

Smart Licensing is ENABLED

Utility:
  Status: DISABLED

Data Privacy:
  Sending Hostname: yes
    Callhome hostname privacy: DISABLED
    Smart Licensing hostname privacy: DISABLED
  Version privacy: DISABLED

Transport:
  Type: Callhome

Registration:
  Status: UNREGISTERED
  Export-Controlled Functionality: NOT ALLOWED
```

```
License Authorization:
  Status: EVAL MODE
  Evaluation Period Remaining: 89 days, 2 hours, 46 minutes, 4 seconds

Export Authorization Key:
  Features Authorized:
    <none>
```

Example: Registering the Router

To register a device, use the **license smart register idtoken** command:

```
Router# license smart register idtoken
Tl4UytrNXBzbEs1ck8veUtWaG5abnZJOFdDa1FwbVRa%0AblRMbz0%3D%0A
```

Example: Viewing License Information After Registration

To display the license entitlements, use the **show license all** command

```
Router# show license all
Smart Licensing Status
======================
Smart Licensing is ENABLED

Registration:
  Status: REGISTERED
  Smart Account: Forty-Two uLtd.
  Export-Controlled Functionality: ALLOWED
  Initial Registration: SUCCEEDED on Sep 11 2020 23:38:45 UTC
  Last Renewal Attempt: None
  Next Renewal Attempt: Mar 10 2021 23:38:45 UTC
  Registration Expires: Sep 11 2021 23:33:42 UTC

License Authorization:
  Status: EVAL MODE
  Evaluation Period Remaining: 89 days, 2 hours, 44 minutes, 21 seconds
  Last Communication Attempt: NOT STARTED
    Failure reason: Communication not started.
  Next Communication Attempt: None
  Communication Deadline: None

Export Authorization Key:
  Features Authorized:
    <none>

Utility:
  Status: DISABLED

Data Privacy:
  Sending Hostname: yes
    Callhome hostname privacy: DISABLED
    Smart Licensing hostname privacy: DISABLED
  Version privacy: DISABLED

Transport:
  Type: Callhome

License Usage
==============

(8808-TRK):
  Description:
```

```
  Count: 1
  Version: 1.0
  Status: EVAL MODE
  Export status: NOT RESTRICTED

(XR-8K-7.0-TRK):
  Description:
  Count: 1
  Version: 1.0
  Status: EVAL MODE
  Export status: NOT RESTRICTED

(8800-LC-48H-TRK):
  Description:
  Count: 1
  Version: 1.0
  Status: EVAL MODE
  Export status: NOT RESTRICTED

Product Information
===================
UDI: PID:8808,SN:FOX224PJKHQ

Agent Version
=============
Smart Agent for Licensing: 4.9.6_rel/41

Reservation Info
================
License reservation: DISABLED
```

To display the license usage information, use the **show license usage** command:

```
Router# show license usage
License Authorization:
  Status: OUT OF COMPLIANCE on Sep 11 2020 23:39:08 UTC

8808 Base HW Tracking PID (8808-TRK):
  Description: 8808 Base HW Tracking PID
  Count: 1
  Version: 1.0
  Status: OUT OF COMPLIANCE
  Export status: NOT RESTRICTED

8000 Software Tracking PID 7.0 (XR-8K-7.0-TRK):
  Description: 8000 Software Tracking PID 7.0
  Count: 1
  Version: 1.0
  Status: OUT OF COMPLIANCE
  Export status: NOT RESTRICTED

8800-LC-48H Linecard Tracking PID (8800-LC-48H-TRK):
  Description: 8800-LC-48H Linecard Tracking PID
  Count: 1
  Version: 1.0
  Status: OUT OF COMPLIANCE
  Export status: NOT RESTRICTED
```

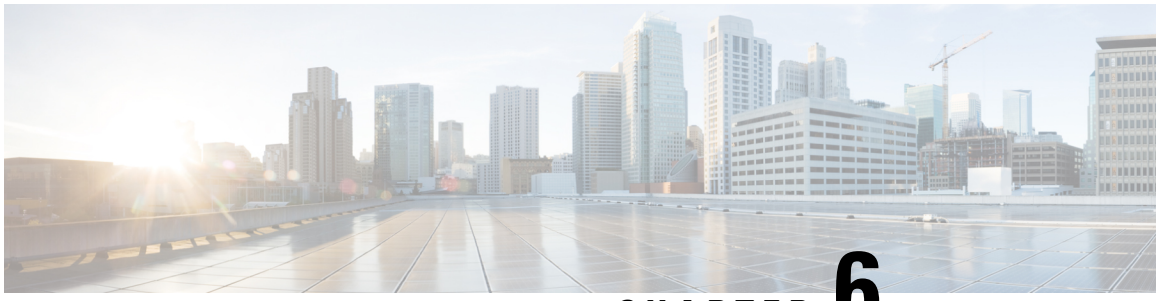
To display all the license summaries, use the **show license summary** command:

```
Router# show license summary
Smart Licensing is ENABLED

Registration:
  Status: REGISTERED
```

```
  Smart Account: Forty-Two uLtd.
  Virtual Account: IOSXR
  Export-Controlled Functionality: ALLOWED
  Last Renewal Attempt: None
  Next Renewal Attempt: Mar 10 2021 23:38:45 UTC

License Authorization:
  Status: OUT OF COMPLIANCE
  Last Communication Attempt: SUCCEEDED
  Next Communication Attempt: Sep 12 2020 11:39:07 UTC

License Usage:
  License                 Entitlement tag               Count Status
  -----------------------------------------------------------------------------
  8808 Base HW Trackin... (8808-TRK)                        1 OUT OF COMPLIANCE
  8000 Software Tracki... (XR-8K-7.0-TRK)                   1 OUT OF COMPLIANCE
  8800-LC-48H Linecard... (8800-LC-48H-TRK)                 1 OUT OF COMPLIANCE
```

To display the license status information, use the **show license status** command:

```
Router# show license status

Smart Licensing is ENABLED

Utility:
  Status: DISABLED

Data Privacy:
  Sending Hostname: yes
    Callhome hostname privacy: DISABLED
    Smart Licensing hostname privacy: DISABLED
  Version privacy: DISABLED

Transport:
  Type: Callhome

Registration:
  Status: REGISTERED
  Smart Account: Forty-Two uLtd.
  Virtual Account: IOSXR
  Export-Controlled Functionality: ALLOWED
  Initial Registration: SUCCEEDED on Sep 11 2020 23:38:45 UTC
  Last Renewal Attempt: None
  Next Renewal Attempt: Mar 10 2021 23:38:45 UTC
  Registration Expires: Sep 11 2021 23:33:42 UTC

License Authorization:
  Status: OUT OF COMPLIANCE on Sep 11 2020 23:39:08 UTC
  Last Communication Attempt: SUCCEEDED on Sep 11 2020 23:39:08 UTC
  Next Communication Attempt: Sep 12 2020 11:39:07 UTC
  Communication Deadline: Dec 10 2020 23:34:07 UTC

Export Authorization Key:
  Features Authorized:
    <none>
```



CHAPTER 6

Configuring Call Home

This module describes the configuring of the Call Home feature.

Table 7: Feature History for Configuring Call Home

Release	Modification
Release 7.0.11	Call Home was introduced

This model contains the following topics:

About Call Home

Call Home provides an email and HTTPS based notification for critical system policies. A range of message formats are available for compatibility with pager services or XML-based automated parsing applications. You can use this feature to page a network support engineer, or email a Network Operations Center. You can also use Cisco Smart Call Home services to generate a case with the Technical Assistance Center. The Call Home feature can deliver alert messages containing information about diagnostics and environmental faults and events.

The Call Home feature can deliver alerts to multiple recipients, referred to as Call Home destination profiles. Each profile includes configurable message formats and content categories. A predefined destination is

provided for sending alerts to the Cisco TAC, however you also can define your own destination profiles. When you configure Call Home to send messages, the appropriate CLI show command is executed and the command output is attached to the message. Call Home messages are delivered in the following formats:

- Short text format which provides a one or two line description of the fault that is suitable for pagers or printed reports.
- Full text format which provides fully formatted message with detailed information that is suitable for human reading.
- XML machine-readable format that uses Extensible Markup Language (XML) and Adaptive Messaging Language (AML) XML schema definition (XSD). The AML XSD is published on the Cisco.com website at http://www.cisco.com. The XML format enables communication with the Cisco Systems Technical Assistance Center.

The Call Home feature is enabled by default. The Cisco TAC-1 profile is created after the device starts. The default Call Home settings that includes destination address, transport methods, alert-group subscriptions, and more are saved in the CiscoTAC-1 profile. To check the default settings, use the **show call-home profile CiscoTAC-1** command.

Benefits of Using Call Home

The Call Home feature offers the following benefits:

- Multiple message-format options:
  - Short Text—Suitable for pagers or printed reports.
  - Plain Text—Full formatted message information suitable for human reading.
  - XML—Matching readable format using Extensible Markup Language (XML) and Adaptive Markup Language (AML) document type definitions (DTDs). The XML format enables communication with the Cisco Smart Call Home server.
- Multiple concurrent message destinations.
- Multiple message categories, including configuration, environmental conditions, inventory, syslog, and crash events.
- Filtering of messages by severity and pattern matching.
- Scheduling of periodic message sending.

Prerequisites for Call Home

How you configure Call Home depends on how you intend to use the feature. Consider the following requirements before you configure Call Home:

- Obtain e-mail, phone, and street address information for the Call Home contact to be configured so that the receiver can determine the origin of messages received.
- Identify the name or IPv4 address of a primary Simple Mail Transfer Protocol (SMTP) server and any backup servers, if using e-mail message delivery.

- Verify IP connectivity from the router to the e-mail server(s) or the destination HTTP server.
- If Cisco Smart Call Home is used, an active service contract covering the device is required to provide full SCH service.

How to Configue Call Home

To configure the sending of Call Home messages, do the following:

1. Assign contact information.
2. Configure and enable one or more destination profiles.
3. Associate one or more alert groups to each profile.
4. Configure the email server options, if using e-mail message delivery.
5. Enable Call Home.

The above tasks are described in detail in the below procedures.

Note Before enabling Call-Home, you must configure the source interface for HTTPS over IPv6. However, for HTTPS over IPv4, Call-Home works without the source interface.

In case of a dual-stack call-home configuration on the device, the IPv4 address is preferred over the IPv6 address. This may result in IPv6 resolution failure. Due to this limitation, the IPv6 device registration with the licensing server may only be done with a single mode, that is, IPv6 only configuration.

Use the **http client source-interface ipv6** command to configure the source interface.

Configuring Contact Information

Each router must include a contact e-mail address. You can optionally include other identifying information for your system installation.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **call-home**

Example:

```
RP/0/RP0/CPU0:router(config)# call-home
RP/0/RP0/CPU0:router(config-call-home)#
```

Enters call home configuration mode.

Step 3 **contact-email-addr** *email-address*

Example:

```
RP/0/RP0/CPU0:router(config-call-home)# contact-email-addr
user1@cisco.com
```

Configures the customer email address. Enter up to 200 characters in email address format with no spaces.

Step 4 (Optional) **contract-id** *contract-id-string*

Example:

```
RP/0/RP0/CPU0:router(config-call-home)# contract-id
Contract-identifier
```

Configures the contract ID. Enter up to 64 characters. If you include spaces, you must enclose the entry in quotes ("").

Step 5 (Optional) **customer-id** *customer-id-string*

Example:

```
RP/0/RP0/CPU0:router(config-call-home)# customer-id Customer1
```

Configures the customer ID. Enter up to 64 characters. If you include spaces, you must enclose the entry in quotes ("").

Step 6 (Optional) **phone-number** *phone-number-string*

Example:

```
RP/0/RP0/CPU0:router(config-call-home)# phone-number +405-123-4567
```

Configures the customer phone number. The number must begin with a plus (+) prefix, and may contain only dashes (-) and numbers. Enter up to 16 characters.

Step 7 (Optional) **street-address** *street-address*

Example:

```
RP/0/RP0/CPU0:router(config-call-home)# street-address "300 E. Tasman Dr.
San Jose, CA 95134"
```

Configures the customer street address where RMA equipment can be shipped. Enter up to 200 characters. If you include spaces, you must enclose the entry in quotes ("").

Step 8 (Optional) **site-id** *site-id-string*

Example:

```
RP/0/RP0/CPU0:router(config-call-home)# site-id SJ-RouterRoom1
```

Configures the site ID for the system. Enter up to 200 characters. If you include spaces, you must enclose the entry in quotes ("").

Step 9 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 10 **show call-home**

Example:

```
RP/0/RP0/CPU0:router# show call-home
```

Displays information about the system contacts.

Destination Profiles

A destination profile includes the following information:

- One or more alert groups—The group of alerts that trigger a specific Call Home message if the alert occurs.
- One or more e-mail or HTTPS destinations—The list of recipients for the Call Home messages generated by alert groups assigned to this destination profile.
- Message format—The format for the Call Home message (short text, full text, or XML).
- Message severity level—The Call Home severity level that the alert must meet before a Call Home message is sent to all e-mail and HTTPS URL addresses in the destination profile. An alert is not generated if the Call Home severity level of the alert is lower than the message severity level set for the destination profile. The inventory and configuration alert groups do not have concept of severity level. They are generated directly.

You can also configure a destination profile to allow periodic inventory update messages by using the inventory alert group that will send out periodic messages daily, weekly, or monthly.

The following predefined destination profiles are supported:

- CiscoTAC-1—Supports the Cisco-TAC alert group in XML message format.

Configuring and Activating Destination Profiles

You must have at least one activated destination profile for Call Home messages to be sent. The CiscoTAC-1 profile exists by default but is not active.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **call-home**

Example:

```
RP/0/RP0/CPU0:router(config)# call-home
RP/0/RP0/CPU0:router(config-call-home)#
```

Enters call home configuration mode.

Step 3 **profile** *profile-name*

Example:

```
RP/0/RP0/CPU0:router(config-call-home)# profile my_profile
RP/0/RP0/CPU0:router(config-call-home-profile)#
```

Enters call home profile configuration mode to configure a new or existing profile.

Step 4 **destination address email** *email-address*

Example:

```
RP/0/RP0/CPU0:router(config-call-home-profile)# destination
address email support_me@cisco.com
```

Configures an email address to which Call Home messages are sent for this profile.

Step 5 **destination message-size-limit** *max-size*

Example:

```
RP/0/RP0/CPU0:router(config-call-home-profile)# destination
message-size-limit 1000
```

Configures the maximum size of Call Home messages for this profile. Values can be between 50 and 3145728 characters.

Step 6 **destination preferred-msg-format** {**short-text** | **long-text** | **xml**}

Example:

```
RP/0/RP0/CPU0:router(config-call-home-profile)# destination
preferred-msg-format xml
```

Configures the message format for this profile. The default is xml.

Step 7 **destination transport-method** [**email** | **hhtp**]

Example:

```
RP/0/RP0/CPU0:router(config-call-home-profile)# destination
transport-method email
```

Configures the transport method for this profile.

Step 8 **active**

Example:

```
RP/0/RP0/CPU0:router(config-call-home-profile)# active
```

Activates the destination profile.

Note At least one destination profile must be active for Call Home messages to be sent.

Step 9 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 10 **show call-home profile** {**all** | *profile-name*}

Example:

```
RP/0/RP0/CPU0:router# show call-home profile all
```

Displays information about the destination profile.

Call Home Alert Groups

An alert group is a predefined subset of alerts or events that Call Home detects and reports to one or more destinations. Alert groups allow you to select the set of alerts that you want to send to a predefined or custom destination profile. Alerts are sent to e-mail destinations in a destination profile only if that alert belongs to one of the alert groups associated with that destination profile and if the alert has a Call Home message severity at or above the message severity set in the destination profile.

The following table lists supported alert groups and the default CLI command output included in Call Home messages generated for the alert group.

Table 8: Alert Groups and Executed Commands

Alert Group	Description	Executed Commands
Environmental	Events related to power, fan, and environment-sensing elements such as temperature alarms.	• **show environment** • **show logging** • **show inventory** • **show environment trace** • **show diag**
Inventory	Inventory status that is provided whenever a unit is cold booted, or when FRUs are inserted or removed. This alert is considered a noncritical event, and the information is used for status and entitlement.	• **show platform** • **show version** • **show diag** • **show inventory oid**

Alert Group	Description	Executed Commands
Syslog	Events generated by specific interesting syslog messages	• **show version** • **show logging** • **show inventory**
Configuration	User-generated request for configuration or configuration change event.	• **show version** • **show running config all** • **show inventory** • **show configuration history last 30** • **show configuration commit changes last 1**
Snapshot	This alert group can be configured for periodic notifications	By default, this alert group has no commands to be run. You can add the required commands that need to be run.

Call Home maps the syslog severity level to the corresponding Call Home severity level for syslog port group messages.

Call Home Message Levels

Call Home allows you to filter messages based on their level of urgency. You can associate each destination profile (predefined and user-defined) with a Call Home message level threshold. The Call Home message level ranges from 0 (lowest level of urgency) to 9 (highest level of urgency). Call Home messages are generated if they have a severity level equal to or greater than the Call Home message level threshold for the destination profile.

Call Home messages that are sent for syslog alert groups have the syslog severity level mapped to the Call Home message level.

Note Call Home does not change the syslog message level in the message text.

The following table lists each Call Home message level keyword and the corresponding syslog level for the syslog port alert group.

Table 9: Severity and syslog Level Mapping

Call Home Level	Keyword	syslog Level	Description
9	Catastrophic	Not-Applicable	Network-wide catastrophic failure.
8	Disaster	Not-Applicable	Significant network impact.

Call Home Level	Keyword	syslog Level	Description
7	Fatal	Emergency (0)	System is unusable.
6	Critical	Alert (1)	Critical conditions that indicate that immediate attention is needed.
5	Major	Critical (2)	Major conditions.
4	Minor	Error (3)	Minor conditions.
3	Warning	Warning (4)	Warning conditions.
2	Notification	Notice (5)	Basic notification and informational messages. Possibly independently insignificant.
1	Normal	Information (6)	Normal event signifying return to normal state.
	Debugging	Debug (7)	Debugging messages.

Associating an Alert Group with a Destination Profile

An alert is sent only to destination profiles that have subscribed to the Call Home alert group.

Before you begin

Use the **show call-home alert-group** command to view available alert groups.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **call-home**

Example:

```
RP/0/RP0/CPU0:router(config)# call-home
RP/0/RP0/CPU0:router(config-call-home)#
```

Enters call home configuration mode.

Step 3 **profile** *profile-name*

Example:

```
RP/0/RP0/CPU0:router(config-call-home)# profile my_profile
RP/0/RP0/CPU0:router(config-call-home-profile)#
```

Enters call home profile configuration mode to configure a new or existing profile.

Step 4 **subscribe-to-alert-group inventory** [**periodic** {**daily** | **monthly** *day-of-month* | **weekly** *day-of-week*} *hh:mm*

Example:

```
RP/0/RP0/CPU0:router(config-call-home-profile)# subscribe-to-alert-group
inventory periodic monthly 1 10:00
```

Configures a destination profile to receive messages for the inventory alert group. Either alerts are sent periodically, or any non-normal event triggers an alert.

Step 5 **subscribe-to-alert-group syslog severity** *severity-level* **pattern** *string*

Example:

```
RP/0/RP0/CPU0:router(config-call-home-profile)# subscribe-to-alert-group
syslog severity major pattern
```

Configures a destination profile to receive messages for the syslog alert group. Alerts with a severity the same or greater than the specified severity level are sent.

- **catastrophic**—Includes network-wide catastrophic events in the alert. This is the highest severity.
- **critical**—Includes events requiring immediate attention (system log level 1).
- **disaster**—Includes events with significant network impact.
- **fatal**—Includes events where the system is unusable (system log level 0).
- **major**—Includes events classified as major conditions (system log level 2).
- **minor**—Includes events classified as minor conditions (system log level 3)
- **normal**—Specifies the normal state and includes events classified as informational (system log level 6). This is the default.
- **notification**—Includes events informational message events (system log level 5).
- **warning**—Includes events classified as warning conditions (system log level 4).

You can specify a pattern to be matched in the syslog message. If the pattern contains spaces, you must enclose it in quotes ("").

Step 6 **subscribe-to-alert-group snapshot severity** *severity-level* **pattern** *string*

Example:

```
RP/0/RP0/CPU0:router(config-call-home-profile)# subscribe-to-alert-group
snapshot severity major pattern
```

Configures a destination profile to receive messages for the snapshot alert group. Alerts with a severity the same or greater than the specified severity level are sent.

You can specify a pattern to be matched in the syslog message. If the pattern contains spaces, you must enclose it in quotes ("").

Step 7 **subscribe-to-alert-group configuration severity** *severity-level* **pattern** *string*

Example:

```
RP/0/RP0/CPU0:router(config-call-home-profile)# subscribe-to-alert-group configuration severity major
 pattern
```

Configures a destination profile to receive messages for the configuration alert group. Alerts with a severity the same or greater than the specified severity level are sent.

You can specify a pattern to be matched in the syslog message. If the pattern contains spaces, you must enclose it in quotes ("").

Step 8 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

Use the **show call-home profile** command to view the profile configurations.

Configuring Email

If Call Home messages are sent via email, the you must configure your email server before Call Home messages can be sent.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **call-home**

Example:

```
RP/0/RP0/CPU0:router(config)# call-home
RP/0/RP0/CPU0:router(config-call-home)#
```

Enters call home configuration mode.

Step 3 (Optional) **sender from** *email-address*

Example:

```
RP/0/RP0/CPU0:router(config-call-home)# sender from
my_email@cisco.com
```

Specifies the email message “from” address.

Step 4 (Optional) **sender reply-to** *email-address*

Example:

```
RP/0/RP0/CPU0:router(config-call-home)# sender reply-to
my_email@cisco.com
```

Specifies the email message “reply-to” address.

Step 5 Required: **mail-server** *address* **priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-call-home)# mail-server
198.61.170.16 priority 1
```

Specifies the mail server to use to send Call Home messages. You can specify an IP address or mail server name. You can specify up to five mail servers to use. The server with the lower priority is tried first.

Step 6 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 7 **show call-home mail-server status**

Example:

```
RP/0/RP0/CPU0:router# show call-home mail-server status
```

Displays the status of the specified mail server.

Configuring a HTTPS Proxy Server

This task enables the user to configure a HTTPS Proxy Server.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **call-home**

Example:

```
RP/0/RP0/CPU0:router (config) # call-home
```

Enters Call Home configuration mode.

Step 3 **http-proxy** *proxy-server-name* **port** *port-number*

Example:

```
RP/0/RP0/CPU0:router (config) # http-proxy p1 port 100
```

Configures the port for the specified HTTPS proxy server. Range is 1 to 65535.

Sending Call-home Data through an Email

This task enables the user to configure sending Call-home data using email as the transport method:

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **call-home**

Example:

```
RP/0/RP0/CPU0:router (config) # call-home
```

Enters Call Home configuration mode.

Step 3 **profile** *name*

Example:

```
RP/0/RP0/CPU0:router (config-call-home) # profile user1
```

Enters call home destination profile configuration mode for the specified destination profile name. If the specified destination profile does not exist, it is created.

Step 4 **active**

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # active
```

Enables the destination profile. By default, a user-defined profile is enabled when it is created.

Step 5 **destination transport-method email**

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # destination transport-method email
```

Configures the message transport method for email. This is the default

Step 6 **destination address email** *email-address*

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # destination address email xyz@cisco.com
```

Configures the destination e-mail address to which Call Home messages are sent.

Step 7 **destination preferred-msg-format {long-text |short-text| xml}**

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # destinationpreferred-msg-format xml
```

(Optional) Configures a preferred message format. The default is XML.

Step 8 **subscribe-to-alert-group syslog severity** *severity-level* **pattern** *string*

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # subscribe-to-alert-group syslog severity normal
pattern COUNT
```

Configures a destination profile to receive messages for the syslog alert group. Alerts with a severity the same or greater than the specified severity level are sent.

- **critical**—Includes events requiring immediate attention (system log level 1).
- **disaster**—Includes events with significant network impact.
- **fatal**—Includes events where the system is unusable (system log level 0).
- **major**—Includes events classified as major conditions (system log level 2).
- **minor**—Includes events classified as minor conditions (system log level 3).
- **normal**—Specifies the normal state and includes events classified as informational (system log level 6).

  This is the default.
- **notification**—Includes events informational message events (system log level 5).
- **warning**—Includes events classified as warning conditions (system log level 4).

You can specify a pattern to be matched in the syslog message. If the pattern contains spaces, you must enclose it in quotes ("").

Step 9 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Sending Call-home Data through HTTPS

This task enables the user to configure sending Call-home data using HTTPS as the transport method:

Note For the HTTPS function to work you should use the **crypto ca trustpoint** command to declare a CA, followed by the **crl option** command. This ensures that the certificates of other peers are accepted without trying to obtain the appropriate CRL. For example:

```
RP/0/RP0/CPU0:ios(config)#crypto ca trustpoint Trustpool
RP/0/RP0/CPU0:ios(config-trustp)#crl optional
```

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **call-home**

Example:

```
RP/0/RP0/CPU0:router (config) # call-home
```

Enters Call Home configuration mode.

Step 3 **profile** *name*

Example:

```
RP/0/RP0/CPU0:router (config-call-home) # profile user1
```

Enters call home destination profile configuration mode for the specified destination profile name. If the specified destination profile does not exist, it is created.

Step 4 **active**

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # active
```

Enables the destination profile. By default, a user-defined profile is enabled when it is created.

Step 5 **destination transport-method http**

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # destination transport-method http
```

Configures the message transport method for HTTPS.

Step 6 **destination address http** *url*

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # destination address http https://example.com
```

Configures the destination URL address to which Call Home messages are sent.

Step 7 **destination preferred-msg-format {long-text |short-text| xml}**

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # destinationpreferred-msg-format xml
```

(Optional) Configures a preferred message format. The default is XML.

Step 8 **subscribe-to-alert-group syslog severity** *severity-level* **pattern** *string*

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # subscribe-to-alert-group syslog severity normal
pattern COUNT
```

Configures a destination profile to receive messages for the syslog alert group. Alerts with a severity the same or greater than the specified severity level are sent.

- **critical**—Includes events requiring immediate attention (system log level 1).
- **disaster**—Includes events with significant network impact.
- **fatal**—Includes events where the system is unusable (system log level 0).
- **major**—Includes events classified as major conditions (system log level 2).
- **minor**—Includes events classified as minor conditions (system log level 3).
- **normal**—Specifies the normal state and includes events classified as informational (system log level 6).

  This is the default.
- **notification**—Includes events informational message events (system log level 5).
- **warning**—Includes events classified as warning conditions (system log level 4).

You can specify a pattern to be matched in the syslog message. If the pattern contains spaces, you must enclose it in quotes ("").

Step 9 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Call Home to use VRF

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **call-home**

Example:

```
RP/0/RP0/CPU0:router (config) # call-home
```

Enters Call Home configuration mode.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router (config-call-home) # vrf v1
```

Configures call home for the specified VRF. VRF works only for the http transport method. It does not work for the email transport method.

Configuring Call Home Data Privacy

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **call-home**

Example:

```
RP/0/RP0/CPU0:router(config) #  call-home
```

Enters the call home configuration submode.

Step 3 **data-privacy** { **level** { **normal** | **high** } | **hostname** }

Example:

```
RP/0/RP0/CPU0:router(config-call-home) # data-privacy level high
```

Scrubs data from call-home message to protect the privacy of the user. The default data-privacy level is normal.

- **normal** - scrubs all normal level commands , such as , password/ username/ ip/ destination.
- **high** - scrubs all normal level commands plus the IP domain name and IP address commands.
- **hostname** - scrubs all high-level or normal-level commands plus the hostname command. It may cause Smart Call Home processing failure.

Note Enabling the data-privacy command can affect CPU utilization when scrubbing a large amount of data.

Sending Smart License Data

This task enables the user to configure sending Smart License data through HTTPS transport method in TAC or user-defined profile:

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **call-home**

Example:

```
RP/0/RP0/CPU0:router (config) # call-home
```

Enters Call Home configuration mode.

Step 3 **profile** *name*

Perform either one of the below actions:

- For sending Smart License data in TAC profile:

  ```
  RP/0/RP0/CPU0:router (config-call-home) # profile CiscoTAC-1
  ```

- For sending Smart License data in user-defined profile:

  ```
  RP/0/RP0/CPU0:router (config-call-home) # profile user1
  ```

Step 4 **active**

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # active
```

Enables the destination profile. By default, a user-defined profile is enabled when it is created.

Step 5 **reporting smart-licensing-data**

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # reporting smart-licensing-data
```

Enables sending Smart Licensing data.

Step 6 **destination transport-method http**

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # destination transport-method http
```

Configures the message transport method for HTTPS.

Step 7 **destination address http** *url*

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # destination address http https://example.com
```

Configures the destination HTTPS address to which Smart License data is sent.

Step 8 **destination preferred-msg-format {long-text |short-text| xml}**

Example:

```
RP/0/RP0/CPU0:router (config-call-home-profile) # destinationpreferred-msg-format xml
```

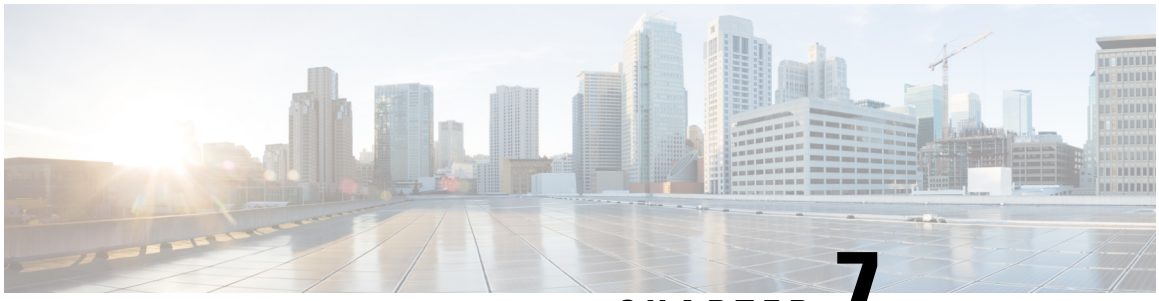
(Optional) Configures a preferred message format. The default is XML.

Step 9 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.



CHAPTER 7

Configuring Network Time Protocol

Prerequisites for Implementing NTP on Cisco IOS XR Software

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Implementing NTP

NTP synchronizes timekeeping among a set of distributed time servers and clients. This synchronization allows events to be correlated when system logs are created and other time-specific events occur.

NTP uses the User Datagram Protocol (UDP) as its transport protocol. All NTP communication uses Coordinated Universal Time (UTC). An NTP network usually receives its time from an authoritative time source, such as a radio clock or an atomic clock attached to a time server. NTP distributes this time across the network. NTP is extremely efficient; no more than one packet per minute is necessary to synchronize two machines to within a millisecond of each other.

NTP uses the concept of a “stratum” to describe how many NTP “hops” away a machine is from an authoritative time source. A “stratum 1” time server typically has an authoritative time source (such as a radio or atomic clock, or a GPS time source) directly attached, a “stratum 2” time server receives its time via NTP from a “stratum 1” time server, and so on.

NTP avoids synchronizing to a machine whose time may not be accurate, in two ways. First, NTP never synchronizes to a machine that is not synchronized itself. Second, NTP compares the time reported by several machines and does not synchronize to a machine whose time is significantly different than the others, even if its stratum is lower. This strategy effectively builds a self-organizing tree of NTP servers.

The Cisco implementation of NTP does not support stratum 1 service; in other words, it is not possible to connect to a radio or atomic clock (for some specific platforms, however, you can connect a GPS time-source device). We recommend that time service for your network be derived from the public NTP servers available in the IP Internet.

If the network is isolated from the Internet, the Cisco implementation of NTP allows a machine to be configured so that it acts as though it is synchronized via NTP, when in fact it has determined the time using other means. Other machines can then synchronize to that machine via NTP.

Several manufacturers include NTP software for their host systems, and a publicly available version for systems running UNIX and its various derivatives is also available. This software also allows UNIX-derivative servers to acquire the time directly from an atomic clock, which would subsequently propagate time information along to Cisco routers.

The communications between machines running NTP (known as *associations*) are usually statically configured; each machine is given the IP address of all machines with which it should form associations. Accurate timekeeping is made possible by exchanging NTP messages between each pair of machines with an association.

In a LAN environment, NTP can be configured to use IP broadcast messages. As compared to polling, IP broadcast messages reduce configuration complexity, because each machine can simply be configured to send or receive broadcast or multicast messages. However, the accuracy of timekeeping is marginally reduced because the information flow is one-way only.

An NTP broadcast client listens for broadcast messages sent by an NTP broadcast server at a designated IPv4 address. The client synchronizes the local clock using the first received broadcast message.

The time kept on a machine is a critical resource, so we strongly recommend that you use the security features of NTP to avoid the accidental or malicious setting of incorrect time. Two mechanisms are available: an access list-based restriction scheme and an encrypted authentication mechanism.

When multiple sources of time (VINES, hardware clock, manual configuration) are available, NTP is always considered to be more authoritative. NTP time overrides the time set by any other method.

Preventing Issues due to GPS Week Number Rollover (WNRO)

- If there are no GPS sources in the NTP source chain or server chain, there is no impact of GPS Week Number Rollover (WNRO).
- GPS WNRO affects only the system clock and not user traffic.
- Contact your GPS manufacturer to fix the GPS source for this condition.

To mitigate impact of GPS sources that are subject to GPS WNRO perform the following optional workarounds:

- If the GPS source has been identified to be a cause of potential disruption on April 6, 2019 (or after), configure ntp master in the Cisco that is device connected to this source, and its clock on the Stratum 1 device to preventively isolate it. This configuration enables the device to present its own clock for synchronization to downstream NTP clients.

Note The usage of ntp master command as mentioned above is only a workaround to this condition. Use this command until the GPS source-related conditions are resolved, and to prevent the distribution of incorrect clock values throughout the network.

- Configure multiple NTP servers (ideally 4, but more than 3) at Stratum 2 level of the network, to enable NTP clients at Stratum 2 level to get clock from more than one Stratum 1 server. This way, WNRO affected Stratum 1 servers are staged to be marked as ‘false ticker’ or ‘outlier’ clock sources as compared to other non-WNRO affected Stratum 1 servers.

How to Implement NTP

Configuring Poll-Based Associations

Note No specific command enables NTP; the first NTP configuration command that you issue enables NTP.

You can configure the following types of poll-based associations between the router and other devices (which may also be routers):

- Client mode
- Symmetric active mode

The client and the symmetric active modes should be used when NTP is required to provide a high level of time accuracy and reliability.

When a networking device is operating in the client mode, it polls its assigned time serving hosts for the current time. The networking device then picks a host from all the polled time servers to synchronize with. Because the relationship that is established in this case is a client-host relationship, the host does not capture or use any time information sent by the local client device. This mode is most suited for file-server and workstation clients that are not required to provide any form of time synchronization to other local clients. Use the **server** command to individually specify the time-serving hosts that you want your networking device to consider synchronizing with and to set your networking device to operate in the client mode.

When a networking device is operating in the symmetric active mode, it polls its assigned time-serving hosts for the current time and it responds to polls by its hosts. Because this is a peer-to-peer relationship, the host also retains time-related information about the local networking device that it is communicating with. This mode should be used when there are several mutually redundant servers that are interconnected via diverse network paths. Most stratum 1 and stratum 2 servers on the Internet today adopt this form of network setup. Use the **peer** command to individually specify the time-serving hosts that you want your networking device to consider synchronizing with and to set your networking device to operate in the symmetric active mode.

When the router polls several other devices for the time, the router selects one device with which to synchronize.

Note To configure a peer-to-peer association between the router and another device, you must also configure the router as a peer on the other device.

You can configure multiple peers and servers, but you cannot configure a single IP address as both a peer and a server at the same time.

To change the configuration of a specific IP address from peer to server or from server to peer, use the **no** form of the **peer** or **server** command to remove the current configuration before you perform the new configuration. If you do not remove the old configuration before performing the new configuration, the new configuration does not overwrite the old configuration.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **ntp**

Example:

```
RP/0/RP0/CPU0:router(config)# ntp
```

Enters NTP configuration mode.

Step 3 **server** *ip-address* [**version** *number*] [**key** *key-id*] [**minpoll** *interval*] [**maxpoll** *interval*] [**source** *type interface-path-id*] [**prefer**] [**burst**] [**iburst**]

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# server 172.16.22.44
    minpoll 8 maxpoll 12
```

Forms a server association with another system. This step can be repeated as necessary to form associations with multiple devices.

Step 4 **peer** *ip-address* [**version** *number*] [**key** *key-id*] [**minpoll** *interval*] [**maxpoll** *interval*] [**source** *type interface-path-id*] [**prefer**]

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# peer 192.168.22.33
    minpoll 8 maxpoll 12 source hundredGigE 0/0/0/1
```

Forms a peer association with another system. This step can be repeated as necessary to form associations with multiple systems.

Note To complete the configuration of a peer-to-peer association between the router and the remote device, the router must also be configured as a peer on the remote device.

Step 5 Use one of the following commands:

- **end**
- **commit**

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# end
```

or

```
RP/0/RP0/CPU0:router(config-ntp)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before
  exiting(yes/no/cancel)?
[cancel]:
```

- Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Broadcast-Based NTP Associates

In a broadcast-based NTP association, an NTP server propagates NTP broadcast packets throughout a network. Broadcast clients listen for the NTP broadcast packets propagated by the NTP server and do not engage in any polling.

Broadcast-based NTP associations should be used when time accuracy and reliability requirements are modest and if your network is localized and has a large number of clients (more than 20). Broadcast-based NTP associations also are recommended for use on networks that have limited bandwidth, system memory, or CPU resources. Time accuracy is marginally reduced in broadcast-based NTP associations because information flows only one way.

Use the **broadcast client** command to set your networking device to listen for NTP broadcast packets propagated through a network. For broadcast client mode to work, the broadcast server and its clients must be located on the same subnet. The time server that is transmitting NTP broadcast packets must be enabled on the interface of the given device using the **broadcast** command.

Use the **broadcast** command to set your networking device to send NTP broadcast packets.

Note No specific command enables NTP; the first NTP configuration command that you issue enables NTP.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **ntp**

Example:

```
RP/0/RP0/CPU0:router(config)# ntp
```

Enters NTP configuration mode.

Step 3 (Optional) **broadcastdelay** *microseconds*

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# broadcastdelay 5000
```

Adjusts the estimated round-trip delay for NTP broadcasts.

Step 4 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# interface POS 0/1/0/0
```

Enters NTP interface configuration mode.

Step 5 **broadcast client**

Example:

```
RP/0/RP0/CPU0:router(config-ntp-int)# broadcast client
```

Configures the specified interface to receive NTP broadcast packets.

Note Go to next step to configure the interface to send NTP broadcast packets.

Step 6 **broadcast** [**destination** *ip-address*] [**key** *key-id*] [**version** *number*]

Example:

```
RP/0/RP0/CPU0:router(config-ntp-int)# broadcast
destination 10.50.32.149
```

Configures the specified interface to send NTP broadcast packets.

Note Go to previous step to configure the interface to receive NTP broadcast packets.

Step 7 Use one of the following commands:

- **end**
- **commit**

Example:

```
RP/0/RP0/CPU0:router(config-ntp-int)# end
```

or

```
RP/0/RP0/CPU0:router(config-ntp-int)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring NTP Access Groups

Note No specific command enables NTP; the first NTP configuration command that you issue enables NTP.

The access list-based restriction scheme allows you to grant or deny certain access privileges to an entire network, a subnet within a network, or a host within a subnet.

The access group options are scanned in the following order, from least restrictive to most restrictive:

1. **peer**—Allows time requests and NTP control queries and allows the system to synchronize itself to a system whose address passes the access list criteria.
2. **serve**—Allows time requests and NTP control queries, but does not allow the system to synchronize itself to a system whose address passes the access list criteria.
3. **serve-only**—Allows only time requests from a system whose address passes the access list criteria.
4. **query-only**—Allows only NTP control queries from a system whose address passes the access list criteria.

If the source IP address matches the access lists for more than one access type, the first type is granted. If no access groups are specified, all access types are granted to all systems. If any access groups are specified, only the specified access types are granted.

For details on NTP control queries, see RFC 1305 (NTP version 3).

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **ntp**

Example:

```
RP/0/RP0/CPU0:router(config)# ntp
```

Enters NTP configuration mode.

Step 3 **access-group**{**peer** | **query-only** | **serve** | **serve-only**} *access-list-name*

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# access-group peer access1
```

Creates an access group and applies a basic IPv4 or IPv6 access list to it.

Step 4 Use one of the following commands:

- **end**
- **commit**

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# end
```

or

```
RP/0/RP0/CPU0:router(config-ntp)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before
  exiting(yes/no/cancel)?
[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring NTP Authentication

This task explains how to configure NTP authentication.

The encrypted NTP authentication scheme should be used when a reliable form of access control is required. Unlike the access-list-based restriction scheme that is based on IP addresses, the encrypted authentication scheme uses authentication keys and an authentication process to determine if NTP synchronization packets sent by designated peers or servers on a local network are deemed as trusted, before the time information that it carries along is accepted.

The authentication process begins from the moment an NTP packet is created. A message authentication code (MAC) is computed using the MD5 Message Digest Algorithm and the MAC is embedded into an NTP synchronization packet. The NTP synchronization packet together with the embedded MAC and key number are transmitted to the receiving client. If authentication is enabled and the key is trusted, the receiving client computes the MAC in the same way. If the computed MAC matches the embedded MAC, the system is allowed to sync to the server that uses this key in its packets.

After NTP authentication is properly configured, your networking device only synchronizes with and provides synchronization to trusted time sources.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **ntp**

Example:

```
RP/0/RP0/CPU0:router(config)# ntp
```

Enters NTP configuration mode.

Step 3 **authenticate**

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# authenticate
```

Enables the NTP authentication feature.

Step 4 **authentication-key** *key-number* **md5** [**clear** | **encrypted**] *key-name*

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# authentication-key 42
md5 clear key1
```

Defines the authentication keys.

- Each key has a key number, a type, a value, and, optionally, a name. Currently the only key type supported is **md5**.

Step 5 **trusted-key** *key-number*

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# trusted-key 42
```

```
Defines trusted authentication keys.
```

- If a key is trusted, this router only synchronizes to a system that uses this key in its NTP packets.

Step 6 Use one of the following commands:

- **end**
- **commit**

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# end
```

or

```
RP/0/RP0/CPU0:router(config-ntp)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before
  exiting(yes/no/cancel)?
[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling NTP Services on a Specific Interface

NTP services are disabled on all interfaces by default.

NTP is enabled globally when any NTP commands are entered. You can selectively prevent NTP packets from being received through a specific interface by turning off NTP on a given interface.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **ntp**

Example:

```
RP/0/RP0/CPU0:router(config)# ntp
```

Enters NTP configuration mode.

Step 3 Use one of the following commands:

- **no interface** *type interface-path-id*
- **interface** *type interface-path-id* **disable**

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# no interface pos 0/0/0/1
```

or

```
RP/0/RP0/CPU0:router(config-ntp)# interface POS 0/0/0/1 disable
```

Disables NTP services on the specified interface.

Step 4 Use one of the following commands:

- **end**
- **commit**

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# end
```

or

```
RP/0/RP0/CPU0:router(config-ntp)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before
  exiting(yes/no/cancel)?
[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Source IP Address for NTP Packets

By default, the source IP address of an NTP packet sent by the router is the address of the interface through which the NTP packet is sent. Use this procedure to set a different source address.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **ntp**

Example:

```
RP/0/RP0/CPU0:router(config)# ntp
```

Enters NTP configuration mode.

Step 3 **source** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# source POS 0/0/0/1
```

Configures an interface from which the IP source address is taken.

Note This interface is used for the source address for all packets sent to all destinations. If a source address is to be used for a specific association, use the **source** keyword in the **peer** or **server** command shown in Configuring Poll-Based Associations, on page 103.

Step 4 Use one of the following commands:

- **end**
- **commit**

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# end
```

or

```
RP/0/RP0/CPU0:router(config-ntp)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before
  exiting(yes/no/cancel)?
[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the System as an Authoritative NTP Server

You can configure the router to act as an authoritative NTP server, even if the system is not synchronized to an outside time source.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **ntp**

Example:

```
RP/0/RP0/CPU0:router(config)# ntp
```

Enters NTP configuration mode.

Step 3 **master** *stratum*

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# master 9
Makes the router an authoritative NTP server.
```

Note Use the **master** command with caution. It is very easy to override valid time sources using this command, especially if a low stratum number is configured. Configuring multiple machines in the same network with the **master** command can cause instability in time keeping if the machines do not agree on the time.

Step 4 Use one of the following commands:

- **end**
- **commit**

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# end
```

or

```
RP/0/RP0/CPU0:router(config-ntp)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before
  exiting(yes/no/cancel)?
[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

- Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
- Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Updating the Hardware Clock

On devices that have hardware clocks (system calendars), you can configure the hardware clock to be periodically updated from the software clock. This is advisable for devices using NTP, because the time and date on the software clock (set using NTP) is more accurate than the hardware clock. The time setting on the hardware clock has the potential to drift slightly over time.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **ntp**

Example:

```
RP/0/RP0/CPU0:router(config)# ntp
```

Enters NTP configuration mode.

Step 3 **update-calendar**

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# update-calendar
```

Configures the router t o update its system calendar from the software clock at periodic intervals.

Step 4 Use one of the following commands:

- **end**
- **commit**

Example:

```
RP/0/RP0/CPU0:router(config-ntp)# end
```

or

```
RP/0/RP0/CPU0:router(config-ntp)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before
  exiting(yes/no/cancel)?
[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Verifying the Status of the External Reference Clock

This task explains how to verify the status of NTP components.

Note The commands can be entered in any order.

Step 1 **show ntp associations** [**detail**] [**location** *node-id*]

Example:

```
RP/0/RP0/CPU0:router# show ntp associations
```

Displays the status of NTP associations.

Step 2 **show ntp status** [**location** *node-id*]

Example:

```
RP/0/RP0/CPU0:router# show ntp status
```

Displays the status of NTP.

FQDN for NTP Server

NTP on Cisco IOS XR Software supports configuration of servers and peers using their Fully Qualified Domain Names (FQDN). While configuring, the FQDN is resolved via DNS into its corresponding IPv4 or

IPv6 address and is stored in the running-configuration of the system. NTP supports FQDN for both IPv4 and IPv6 protocols. You can configure FQDN on default vrf.

Configure FQDN for NTP server

Configuration Example for FQDN on NTP Server on Default VRF

Use the **ntp server** command with the FQDN name to configure FQDN on default VRF. You dont need to specify VRF name. In the following example, *time.cisco.com* is the FQDN.

```
Router#configure
Router(config)#ntp server time.cisco.com
Router(config)#commit
```

Running Configuration

Use the **show running-config ntp** command to see the ntp running configuration.

```
Router#show running-config ntp
ntp
 server 192.0.2.1
!
```

Verification

Use the **show ntp associations** command to verify that an NTP association has come up.

```
Router#show ntp associations

      address       ref clock     st  when  poll reach  delay  offset    disp
 ~192.0.2.1       173.38.201.67    2    42   128    3  196.06  -14.25  3949.4
 * sys_peer, # selected, + candidate, - outlayer, x falseticker, ~ configured
```

Configuration Examples for Implementing NTP

Configuring Poll-Based Associations: Example

The following example shows an NTP configuration in which the router's system clock is configured to form a peer association with the time server host at IP address 192.168.22.33, and to allow the system clock to be synchronized by time server hosts at IP address 10.0.2.1 and 172.19.69.1:

```
ntp
  server 10.0.2.1 minpoll 5 maxpoll 7
  peer 192.168.22.33

  server 172.19.69.1
```

Configuring Broadcast-Based Associations: Example

The following example shows an NTP client configuration in which interface 0/2/0/0 is configured to receive NTP broadcast packets, and the estimated round-trip delay between an NTP client and an NTP broadcast server is set to 2 microseconds:

```
ntp
  interface hundredGigE 0/2/0/0
    broadcast client
    exit
  broadcastdelay 2
```

The following example shows an NTP server configuration where interface 0/2/0/2 is configured to be a broadcast server:

```
ntp
  interface hundredGigE 0/2/0/2
    broadcast
```

Configuring NTP Access Groups: Example

The following example shows a NTP access group configuration where the following access group restrictions are applied:

- Peer restrictions are applied to IP addresses that pass the criteria of the access list named peer-acl.
- Serve restrictions are applied to IP addresses that pass the criteria of access list named serve-acl.
- Serve-only restrictions are applied to IP addresses that pass the criteria of the access list named serve-only-acl.
- Query-only restrictions are applied to IP addresses that pass the criteria of the access list named query-only-acl.

```
ntp
  peer 10.1.1.1
  peer 10.1.1.1
  peer 10.2.2.2
  peer 10.3.3.3
  peer 10.4.4.4
  peer 10.5.5.5
  peer 10.6.6.6
  peer 10.7.7.7
  peer 10.8.8.8
  access-group peer peer-acl
  access-group serve serve-acl
  access-group serve-only serve-only-acl
  access-group query-only query-only-acl
  exit
ipv4 access-list peer-acl
  10 permit ip host 10.1.1.1 any
  20 permit ip host 10.8.8.8 any
  exit
ipv4 access-list serve-acl
  10 permit ip host 10.4.4.4 any
  20 permit ip host 10.5.5.5 any
  exit
ipv4 access-list query-only-acl
  10 permit ip host 10.2.2.2 any
  20 permit ip host 10.3.3.3 any
  exit
ipv4 access-list serve-only-acl
```

```
  10 permit ip host 10.6.6.6 any
  20 permit ip host 10.7.7.7 any
  exit
```

Configuring NTP Authentication: Example

The following example shows an NTP authentication configuration. In this example, the following is configured:

- NTP authentication is enabled.
- Two authentication keys are configured (key 2 and key 3).
- The router is configured to allow its software clock to be synchronized with the clock of the peer (or vice versa) at IP address 10.3.32.154 using authentication key 2.
- The router is configured to allow its software clock to be synchronized with the clock by the device at IP address 10.32.154.145 using authentication key 3.
- The router is configured to synchronize only to systems providing authentication key 3 in their NTP packets.

```
ntp
  authenticate
  authentication-key 2 md5 encrypted 06120A2D40031D1008124
  authentication-key 3 md5 encrypted 1311121E074110232621
  trusted-key 3
  server 10.3.32.154 key 3
  peer 10.32.154.145 key 2
```

Disabling NTP on an Interface: Example

The following example shows an NTP configuration in which 0/2/0/0 interface is disabled:

```
ntp
  interface hundredGigE 0/2/0/0
    disable
    exit
  authentication-key 2 md5 encrypted 06120A2D40031D1008124
  authentication-key 3 md5 encrypted 1311121E074110232621
  authenticate
  trusted-key 3
  server 10.3.32.154 key 3
  peer 10.32.154.145 key 2
```

Configuring the Source IP Address for NTP Packets: Example

The following example shows an NTP configuration in which Ethernet management interface 0/0/CPU0/0 is configured as the source address for NTP packets:

```
ntp
```

```
  authentication-key 2 md5 encrypted 06120A2D40031D1008124
  authentication-key 3 md5 encrypted 1311121E074110232621
  authenticate
  trusted-key 3
  server 10.3.32.154 key 3
  peer 10.32.154.145 key 2
  source MgmtEth0/0/CPU0/0
```

Configuring the System as an Authoritative NTP Server: Example

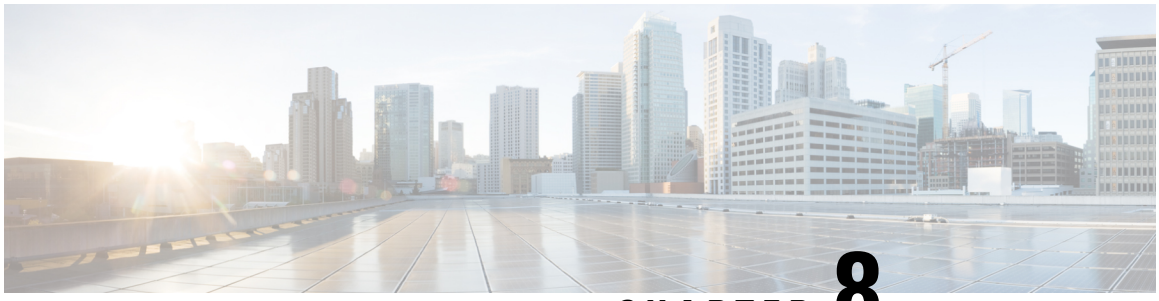
The following example shows a NTP configuration in which the router is configured to use its own NTP master clock to synchronize with peers when an external NTP source becomes unavailable:

```
ntp
  master 6
```

Updating the Hardware Clock: Example

The following example shows an NTP configuration in which the router is configured to update its hardware clock from the software clock at periodic intervals:

```
ntp
  server 10.3.32.154
  update-calendar
```



CHAPTER 8

The Network Configuration Protocol

The Network Configuration Protocol (Netconf) provides mechanisms 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, as defined in RFC6241. Yang is a data modeling language used with Netconf, as defined in RFC6020.

Netconf uses a simple RPC-based (Remote Procedure Call) mechanism to facilitate communication between a client and a server. The client can be a script or application typically running as part of a network manager. The server is typically a network device.

Netconf runs within a Secure Shell (SSH) session as an SSH subsystem, as defined in RFC6242.

The configuration of features need not be done the traditional way (using CLIs), the client application (controller) reads the Yang model and communicates with the Netconf server (IOS XR) accordingly.

Netconf Sessions and Operations

A Netconf session is the logical connection between a network configuration application and a network device. A device should be capable of supporting multiple sessions and atleast one Netconf session.

Characteristics of a netconf session:

- Netconf is connection-oriented - SSH is the underlying transport.
- The netconf client establishes session with the server.
- Netconf sessions are established with the *hello* message. Features and capabilities are announced.
- Sessions can be terminated using the *close* or *kill* messages.

Basic Netconf operations:

- Get configuration <get-config>
- Get all information <get>

- Edit configuration <edit-config>
- Copy configuration <copy-config>

Note <copy-config> does not support source attribute with “data store” at present.

- <lock>, <unlock>
- <kill-session>
- <close-session>
- Commit configuration <commit>

The Yang data model

Each feature has a defined Yang Model which is synthesized from the schemas. A model is published in a tree format and includes:

- Top level nodes and their subtrees
- Subtrees that augment nodes in other yang models

```
Example: The aaa Yang model
(exec-19.42.10) bash-4.2$ pyang -f tree Cisco-IOS-XR-aaa-lib-cfg.yang
module: Cisco-IOS-XR-aaa-lib-cfg
    +--rw aaa
       +--rw accountings
       |  +--rw accounting* [type listname]
       |     +--rw type                  xr:Cisco-ios-xr-string
       |     +--rw listname              xr:Cisco-ios-xr-string
       |     +--rw rp-failover?          dt1:Aaa-accounting-rp-failover
       |     +--rw broadcast?            dt1:Aaa-accounting-broadcast
       |     +--rw type-xr?              dt1:Aaa-accounting
       |     +--rw method1?              dt1:Aaa-method-accounting
       |     +--rw method2?              dt1:Aaa-method-accounting
       |     +--rw method3?              dt1:Aaa-method-accounting
       |     +--rw method4?              dt1:Aaa-method-accounting
       |     +--rw server-group-name1?   string
       |     +--rw server-group-name2?   string
       |     +--rw server-group-name3?   string
       |     +--rw server-group-name4?   string
       +--rw authorizations
       |  +--rw authorization* [type listname]
       |     +--rw type                  xr:Cisco-ios-xr-string
       |     +--rw listname              xr:Cisco-ios-xr-string
       |     +--rw method1?              dt1:Aaa-method
       |     +--rw method2?              dt1:Aaa-method
       |     +--rw method3?              dt1:Aaa-method
       |     +--rw method4?              dt1:Aaa-method
       |     +--rw server-group-name1?   string
       |     +--rw server-group-name2?   string
       |     +--rw server-group-name3?   string
       |     +--rw server-group-name4?   string
       +--rw accounting-update!
       |  +--rw type                 dt1:Aaa-accounting-update
       |  +--rw periodic-interval?   uint32
```

```
+--rw banner
|  +--rw login?   string
+--rw authentications
   +--rw authentication* [type listname]
      +--rw type                  xr:Cisco-ios-xr-string
      +--rw listname              xr:Cisco-ios-xr-string
      +--rw method1?              dt1:Aaa-method
      +--rw method2?              dt1:Aaa-method
      +--rw method3?              dt1:Aaa-method
      +--rw method4?              dt1:Aaa-method
      +--rw server-group-name1?   string
      +--rw server-group-name2?   string
      +--rw server-group-name3?   string
      +--rw server-group-name4?   string
```

Advantages of using the Yang model are:

- Yang supports programmatic interfaces.
- Yang supports simplified network management applications.
- Yang supports interoperability that provides a standard way to model management data.

Netconf and Yang

The workflow displayed here, will help the user to understand how Netconf-Yang can configure and control the network with minimal user intervention. The required components:

- Cisco 8000 Series Router with Netconf capability
- Netconf Client Application with connection to the router

S. No.	Device / component	Action
1	Cisco router	Login/ access the router.
2	Cisco router	Prerequisites for enabling Netconf: • Crypto keys must be generated.
3	Cisco router	Enable Netconf agent. Use the **netconf-yang agent ssh** and **ssh server netconf** command. The port can be selected. By default, it is set as 830.
4	Cisco router	Yang models are a part of the software image. The models can be retrieved from the router , using the <get-schema> operation.

S. No.	Device / component	Action
5	Netconf client (application) The application can be on any standalone application or a SDN controller supporting Netconf	Installs and processes the Yang models. The client can offer a list of supported yang models; else the user will have to browse and locate the required yang file. There is a yang model file for each configuration module; for instance if the user wants to configure CDP , the relevant yang model is Cisco-IOS-XR-cdp-cfg **Note** Refer the table which lists all the supported yang models Supported Yang Models , on page 124
5	Netconf client	Sends Netconf operation request over SSH to the router. A configuration request could include Yang-based XML data to the router. Currently, SSH is the only supported transport method.
6	Cisco router	Understands the Yang-based XML data and the network is configured accordingly (in case of configuration request from the client).
		The interactions between the client and the router happens until the network is configured as desired.

Supported Yang Models

The Yang models can be downloaded from a prescribed location (ftp server) or can also be retrieved directly from the router using the get-schema operation.

For a feature, separate Yang models are available for configuring the feature and to get operational statistics (show commands). The **-cfg.yang** suffix denotes configuration and **-oper*.yang** is for operational data statistics. In some cases, **-oper** is followed by **-sub**, indicating that a submodule(s) is available.

For a list of supported Yang models, see https://github.com/YangModels/yang/tree/master/vendor/cisco/xr

Denial of Services Defense for Netconf-Yang

In case of a DoS (Denial of Service) attack on Netconf, wherein, Netconf receives numerous requests in a short span of time, the router may become irresponsive if Netconf consumes most of the bandwidth or CPU processing time. This can be prevented, by limiting the traffic directed at the Netconf agent. This is achieved using the **netconf-yang agent rate-limit** and **netconf-yang agent session** commands.

If rate-limit is set, the Netconf processor measures the incoming traffic from the SSH server. If the incoming traffic exceeds the set rate-limit, the packets are dropped.

If session-limit is set, the Netconf processor checks for the number of open sessions. If the number of current sessions is greater than or equal to, the set limit, no new sessions are opened.

Session idle- timeout and absolute-timeout also prevent DoS attacks. The Netconf processor closes the sessions, even without user input or intervention, as soon at the time out session is greater than or equal to the set time limit.

The relevant commands are discussed in detail, in the *System Security Command Reference for Cisco 8000 Series Routers*

Enabling NETCONF over SSH

This task enables NETCONF over SSH. SSH is currently the only supported transport method .

If the client supports, Netconf over ssh can utilize the multi-channeling capabilities of IOS XR ssh server.

Prerequisite:

- Crypto keys must be generated prior to this configuration.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters mode.

Step 2 **netconf-yang agent ssh**

Example:

```
RP/0/RP0/CPU0:router (config) # netconf agent ssh
```

Enables NETCONF agent over SSH connection. After NETCONF is enabled, the Yang model in the controllcker, can configure the relevant models.

Note The Yang models can be retrieved from the router via NETCONF <get-schema> operation.

Step 3 **ssh server netconf** [**vrf** *vrf-name* [**ipv4 access-list***ipv4 access list name*] [**ipv6 access-list** *ipv6 access list name*]]

Example:

```
RP/0/RP0/CPU0:router (config) # ssh server netconf vrf netconfvrf ipv4 access-list InternetFilter
```

Brings up the netconf subsytem support with SSH server using a specified VRF of up to 32 characters. If no VRF is specified, the default VRF is used. To stop the SSH server from receiving any further connections for the specified VRF, use the **no** form of this command.

Optionally ACLs for IPv4 and IPv6 can be used to restrict access to the netconf subsystem of the ssh server before the port is opened.

Note The netconf subsystem support with SSH server can be configured for use with multiple VRFs .

Step 4 **ssh server netconf port** *port-number*

Example:

```
RP/0/RP0/CPU0:router (config) # ssh server netconf port 830
```

Configures a port for the netconf ssh server. This command is optional. If no port is specified, port 830 is uses by default.

Note 830 is the IANA-assigned TCP port for NETCONF over SSH, but it can be changed using this command.

What to do next

The **show netconf-yang statistics** command and **show netconf-yang clients** command can be used to verify the configuration details of the netconf agent.

The **clear netconf-yang agent session** command clears the specified Netconf session (on the Netconf server side).

Examples: Netconf over SSH

This section illustrates some examples relevant to Netconf:

Enabling netconf-yang for ssh transport and netconf subsystem for default vrf with default port (830)

```
config
netconf-yang agent ssh
ssh server netconf vrf default
  !
!
```

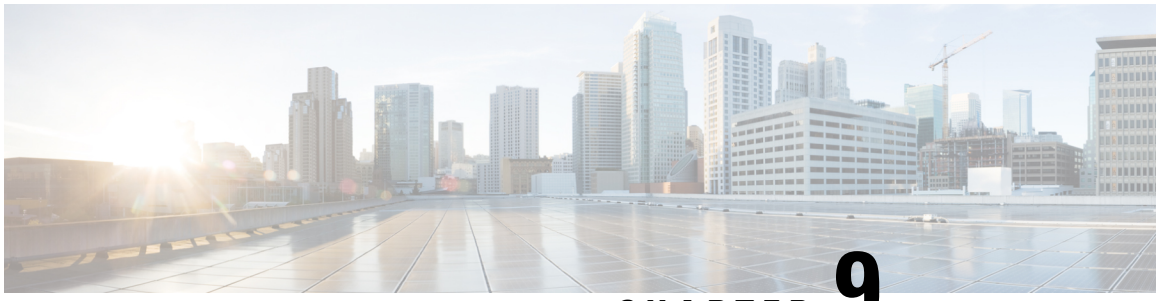
Enabling netconf-yang for ssh transport and netconf subsystem for vrf *green* and vrf *red* with netconf port (831)

```
config
netconf-yang agent ssh
!
ssh server netconf vrf green
ssh server netconf vrf red
ssh server netconf port 831
  !
!
```

Show command outputs

```
show netconf-yang statistics
Summary statistics        requests|             total time|   min time per request|   max
time per request|   avg time per request|
other                              0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
close-session                      4|       0h  0m  0s   3ms|       0h  0m  0s   0ms|
 0h  0m  0s   1ms|       0h  0m  0s   0ms|
kill-session                       0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
get-schema                         0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
get                                0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s
get-config                         1|       0h  0m  0s   1ms|       0h  0m  0s   1ms|
 0h  0m  0s   1ms|       0h  0m  0s   1ms|
edit-config                        3|       0h  0m  0s   2ms|       0h  0m  0s   0ms|
 0h  0m  0s   1ms|       0h  0m  0s   0ms|
commit                             0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
cancel-commit                      0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
```

```
lock                                0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
unlock                              0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
discard-changes                     0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|
validate                            0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
 0h  0m  0s   0ms|       0h  0m  0s   0ms|

show netconf-yang clients
client session ID|   NC version|    client connect time|        last OP time|        last
OP type|    <lock>|
22969|                 1.1|         0d  0h  0m  2s|             11:11:24|
close-session|        No|
15389|                1.1|        0d  0h  0m  1s|            11:11:25|         get-config|
          No|
```



CHAPTER 9

Provision Network Devices using Zero Touch Provisioning

Manually deploying network devices in a large-scale environment requires skilled workers and is time consuming.

With Zero Touch Provisioning (ZTP), you can seamlessly provision thousands of network devices accurately within minutes and without any manual intervention. This can be easily defined using a configuration file or script using shell or phyton.

Learn about Zero Touch Provisioning

ZTP allows you to provision the network device with day 0 configurations and supports both management ports and data ports.

ZTP provides multiple options, such as:

- Automatically apply specific configuration in a large-scale environment.
- Download and install specific IOS XR image.
- Install specific application package or third party applications automatically.
- Deploy containers without manual intervention.
- Upgrade or downgrade software versions effortlessly on thousands of network devices at a time

Benefits of Using ZTP

ZTP helps you manage large-scale service providers infrastructures effortlessly. Following are the added benefits of using ZTP:

- ZTP helps you to remotely provision a router anywhere in the network. Thus eliminates the need to send an expert to deploy network devices and reduces IT cost.
- Automated provisioning using ZTP can remove delay and increase accuracy and thus is cost-effective and provides better customer experience.

  By automating repeated tasks, ZTP allows network administrators to concentrate on more important stuff.
- ZTP process helps you to quickly restore service. Rather than troubleshooting an issue by hand, you can reset a system to well-known working status.

Use Cases

The following are some of the useful use cases for ZTP:

- Using ZTP to install Chef
- Using ZTP to integrate IOS-XR with NSO
- Using ZTP to install Puppet

You can initiate ZTP in one of the following ways:

- **Fresh Boot**: Use this method for devices that has no pre-loaded configuration. See Getting Started with ZTP on a Fresh Boot of a Router. See Zero Touch Provisioning on a Fresh Boot of a Router, on page 130.
- **Manual Invocation:** Use this method when you want to forcefully initiate ZTP on a fully configured device. See Manual ZTP Invocation , on page 142.
- **ZTP Bootscript**: Use this method when you want to hard code a script to be executed on every boot. See Configure ZTP BootScript, on page 144.

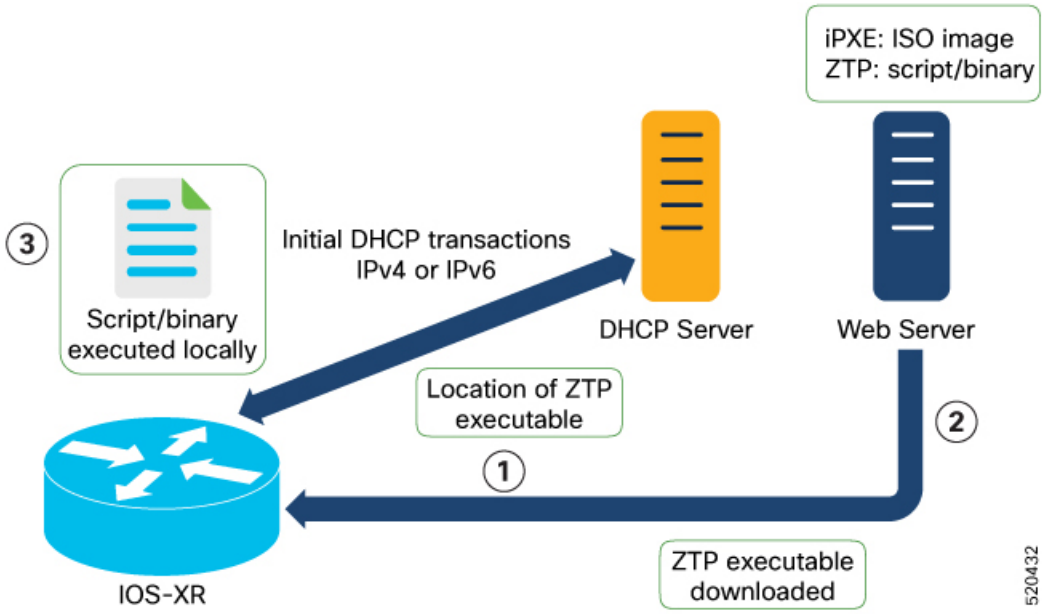
Zero Touch Provisioning on a Fresh Boot of a Router

When you boot the device, the ZTP process initiates automatically if the device does not have a prior configuration.

Fresh Boot Using DHCP

When you boot the device, the ZTP process initiates automatically if the device does not have a prior configuration. During the process, the router receives the details of the configuration file from the DHCP server.

This image depicts the high-level work flow of the ZTP process:



The ZTP process initiates when you boot the network-device with an IOS-XR image. The process starts only on the device that doesn't have a prior configuration.

Here is the high-level work flow of the ZTP process for the Fresh boot:

1. ZTP sends DHCP request to fetch the ZTP configuration file or user script. To help the Bootstrap server uniquely identify the device, ZTP sends below DHCP option
   - DHCP(v4/v6) client-id=Serial Number
   - DHCPv4 option 124: Vendor, Platform, Serial-Number
   - DHCPv6 option 16: Vendor, Platform, Serial-Number

   The following is the default sequential flow of the ZTP process:
   - ZTP sends IPv4 DHCP request first on all the management port. In case there is a failure, then ZTP sends IPv6 DHCP request on all the management port.
   - ZTP sends IPv4 DHCP request first on all the data port. In case there is a failure, then ZTP sends IPv6 DHCP request on all the data port.

     The default sequential flow is defined in configuration file and you can modify the sequence using the configuration file.

2. DHCP server identifies the device and responds with DHCP response using one of the following options:

   DHCP server should be configured to respond with the DHCP options.
   - DHCPv4 using BOOTP filename to supply script/config location.
   - DHCPv4 using Option 67 (bootfile-name) to supply script/config location.
   - DHCPv6 using Option 59 (OPT_BOOTFILE_URL) to supply script/config location

3. The network device downloads the file from the web server using the URI location provided in the DHCP response.
4. The device receives a configuration file or script file from the HTTP server.

Note
- If the downloaded file content starts with !! IOS XR it is considered as a configuration file.
- If the downloaded file content starts with #! /bin/bash, #! /bin/sh or #!/usr/bin/python it is considered as a script file.

5. The device applies the configuration file or executes the script or binary in the default bash shell.
6. The Network device is now up and running.

Configure ZTP Using DHCP

Before you begin to configure ZTP, ensure to set up the DHCP server and HTTP server

Note The HTTP server should be reachable from the management interface or from a data interface if invoked manually and should be readable.

Follow these steps to configure ZTP on a fresh boot:

1. Create the configuration file or user script. See Build your Configuration File, on page 132.
2. Copy the configuration file or user script to the HTTP server.
3. Set up the DHCP for ZTP. See Set Up DHCP Server for ZTP, on page 138.
4. Authenticate the Data Port. See Authentication on Data Ports, on page 141
5. Connect the device to the network.

   ZTP process starts and invokes the DHCP request.

Note When initiated, ZTP checks if the system start-up configuration is applied. If startup configuration is not applied, ZTP waits for 10 minutes before proceeding.

When ZTP process encounters any error, or when ZTP quits or terminates, it revert to the initial configuration that exists before starting of ZTP process.

Build your Configuration File

Based on the business need, you can use a configuration or script file to initiate the ZTP process.

The configuration file content starts with *!! IOS XR* and the script file content starts with *#! /bin/bash, #! /bin/sh or #!/usr/bin/python.*

Once you create the configuration file, apply it to the device using the *ztp_helper* function *xrapply*.

The following is the sample configuration file:

```
!! IOS XR
username root
group root-lr
password 0 lablab
!

hostname ios
alias exec al show alarms brief system active

interface HundredGigE 0/0/0/24
ipv4 address 10.10.10.55 255.255.255.0
no shutdown
!
```

Create User Script

This script or binary is executed in the IOS-XR Bash shell and can be used to interact with IOS-XR CLI to configure, verify the configured state and even run exec commands based on the workflow that the operator chooses.

Build your ZTP script with either shell and python. ZTP includes a set of CLI commands and a set of shell utilities that can be used within the user script.

ZTP Shell Utilities

ZTP includes a set of shell utilities that can be sourced within the user script. **ztp_helper.sh** is a shell script that can be sourced by the user script. **ztp_helper.sh** provides simple utilities to access some XR functionalities. Following are the bash functions that can be invoked:

- **xrcmd**—Used to run a single XR exec command:

  ```
  xrcmd "show running"
  ```

- **xrapply**—Applies the block of configuration, specified in a file:

  ```
  cat >/tmp/config <<%%
  !! XR config example
  hostname node1-mgmt-via-xrapply
  %%
  xrapply /tmp/config
  ```

- **xrapply_with_reason**—Used to apply a block of XR configuration along with a reason for logging purpose:

  ```
  cat >/tmp/config <<%%
  !! XR config example
  hostname node1-mgmt-via-xrapply
  %%
  xrapply_with_reason "this is a system upgrade" /tmp/config
  ```

- **xrapply_string**—Used to apply a block of XR configuration in one line:

  ```
  xrapply_string "hostname foo\ninterface HundredGigE0/0/0/24\nipv4 address 1.2.3.44
  255.255.255.0\n"
  ```

- **xrapply_string_with_reason**—Used to apply a block of XR configuration in one line along with a reason for logging purposes:

```
xrapply_string_with_reason ”system renamed again" "hostname venus\n interface
HundredGigE0/0/0/24\n ipv4 address 172.30.0.144/24\n”
```

- **xrreplace**—Used to apply XR configuration replace in XR namespace via a file.

```
cat rtr.cfg <<%%
!! XR config example
hostname node1-mgmt-via-xrreplace
%%
xrreplace rtr.cfg
```

- **xrapply_with_extra_auth**—Used to apply XR configuration that requires authentication, in XR namespace via a file. The **xrapply_with_extra_auth** API is used when configurations that require additional authentication to be applied such as alias, flex groups.

```
cat >/tmp/config <<%%
!! XR config example
alias exec alarms show alarms brief system active
alias exec version run cat /etc/show_version.txt
%%
xrapply_with_extra_auth >/tmp/config
```

- **xrreplace_with_extra_auth**—Used to apply XR configuration replace in XR namespace via a file The **xrreplace_with_extra_auth** API is used when configurations that require additional authentication to be applied such as alias, flex groups

```
cat >/tmp/config <<%%
!! XR config example
alias exec alarms show alarms brief system active
alias exec version run cat /etc/show_version.txt
%%
xrreplace_with_extra_auth >/tmp/config
```

ZTP Helper Python Library

The ZTP python library defines a single Python class called `ZtpHelpers`. The helper script is located at `/pkg/bin/ztp_helper.sh`

ZtpHelpers Class Methods

Following are utility methods of the `ZtpHelpers` class:

- `init(self, syslog_server=None, syslog_port=None, syslog_file=None):`

```
__init__ constructor
:param syslog_server: IP address of reachable Syslog Server
       :param syslog_port: Port for the reachable syslog server
        :param syslog_file: Alternative or addon file for syslog
        :type syslog_server: str
        :type syslog_port: int
        :type syslog_file:str

         All parameters are optional. When nothing is specified during object creation,
 then all logs are sent to a log rotated file /tmp/ztp_python.log (max size of 1MB).
```

- `setns(cls, fd, nstype):`

```
Class Method for setting the network namespace
           :param cls: Reference to the class ZtpHelpers
           :param fd: incoming file descriptor
           :param nstype: namespace type for the sentns call
           :type nstype: int
                  0      Allow any type of namespace to be joined.
                  CLONE_NEWNET = 0x40000000 (since Linux 3.0)
                         fd must refer to a network namespace
```

- `get_netns_path(cls, nspath=None, nsname=None, nspid=None):`

```
Class Method to fetch the network namespace filepath
           associated with a PID or name
           :param cls: Reference to the class ZtpHelpers
           :param nspath: optional network namespace associated name
           :param nspid: optional network namespace associate PID
           :type nspath: str
           :type nspid: int
           :return: Return the complete file path
           :rtype:  str
```

- `toggle_debug(self, enable):`

```
Enable/disable debug logging
           :param enable: Enable/Disable flag
           :type enable: int
```

- `set_vrf(self, vrfname=None):`

```
Set the VRF (network namespace)
           :param vrfname: Network namespace name
                           corresponding to XR VRF
```

- `download_file(self, file_url, destination_folder):`

```
Download a file from the specified URL
           :param file_url: Complete URL to download file
           :param destination_folder: Folder to store the
                                      downloaded file
           :type file_url: str
           :type destination_folder: str
           :return: Dictionary specifying download success/failure
                    Failure => { 'status' : 'error' }
                    Success => { 'status' : 'success',
                                 'filename' : 'Name of downloaded file',
                                 'folder' : 'Directory location of downloaded file'}
           :rtype: dict
```

- `setup_syslog(self):`

  Method to Correctly set sysloghandler in the correct VRF (network namespace) and point to a remote syslog Server or local file or default log-rotated log file.

- `xrcmd(self, cmd=None):`

```
Issue an IOS-XR exec command and obtain the output
           :param cmd: Dictionary representing the XR exec cmd
                       and response to potential prompts
                       { 'exec_cmd': '', 'prompt_response': '' }
           :type cmd: dict
           :return: Return a dictionary with status and output
                    { 'status': 'error/success', 'output': '' }
           :rtype: dict
```

- xrapply(self, filename=None, reason=None):

```
Apply Configuration to XR using a file
           :param file: Filepath for a config file
                        with the following structure:
                        !
                        XR config command
                        !
                        end

           :param reason: Reason for the config commit.
                          Will show up in the output of:
                          "show configuration commit list detail"
           :type filename: str
           :type reason: str
           :return: Dictionary specifying the effect of the config change
                    { 'status' : 'error/success', 'output': 'exec command based on
status'}
                    In case of Error:  'output' = 'show configuration failed'
                    In case of Success: 'output' = 'show configuration commit changes
 last 1'
           :rtype: dict
```

- xrapply_string(self, cmd=None, reason=None):

```
Apply Configuration to XR using  a single line string
           :param cmd: Single line string representing an XR config command
           :param reason: Reason for the config commit.
                          Will show up in the output of:
                          "show configuration commit list detail"
           :type cmd: str
           :type reason: str
           :return: Dictionary specifying the effect of the config change
                    { 'status' : 'error/success', 'output': 'exec command based on
status'}
                    In case of Error:  'output' = 'show configuration failed'
                    In case of Success: 'output' = 'show configuration commit changes
 last 1'
           :rtype: dict
```

- xrreplace(self, filename=None):

```
Replace XR Configuration using a file

           :param file: Filepath for a config file
                        with the following structure:

                        !
                        XR config commands
                        !
                        end
           :type filename: str
           :return: Dictionary specifying the effect of the config change
                    { 'status' : 'error/success', 'output': 'exec command based on
status'}
                    In case of Error:  'output' = 'show configuration failed'
                    In case of Success: 'output' = 'show configuration commit changes
 last 1'
           :rtype: dict
```

Example

The following shows the sample script in phyton

```
[apple2:~]$ python sample_ztp_script.py

###### Debugs enabled ######


###### Change context to user specified VRF ######


###### Using Child class method, setting the root user ######

2016-12-17 04:23:24,091 - DebugZTPLogger - DEBUG - Config File content to be applied  !
                     username netops
                     group root-lr
                     group cisco-support
                     secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1
                     !
                     end
2016-12-17 04:23:28,546 - DebugZTPLogger - DEBUG - Received exec command request: "show
configuration commit changes last 1"
2016-12-17 04:23:28,546 - DebugZTPLogger - DEBUG - Response to any expected prompt ""
Building configuration...
2016-12-17 04:23:29,329 - DebugZTPLogger - DEBUG - Exec command output is ['!! IOS XR
Configuration version = 6.2.1.21I', 'username netops', 'group root-lr', 'group cisco-support',
 'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1', '!', 'end']
2016-12-17 04:23:29,330 - DebugZTPLogger - DEBUG - Config apply through file successful,
last change = ['!! IOS XR Configuration version = 6.2.1.21I', 'username netops', 'group
root-lr', 'group cisco-support', 'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1', '!', 'end']


###### Debugs Disabled ######


###### Executing a show command ######

Building configuration...
{'output': ['!! IOS XR Configuration version = 6.2.1.21I',
            '!! Last configuration change at Sat Dec 17 04:23:25 2016 by UNKNOWN',
            '!',
            'hostname customer2',
            'username root',
            'group root-lr',
            'group cisco-support',
            'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1',
            '!',
            'username noc',
            'group root-lr',
            'group cisco-support',
            'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1',
            '!',
            'username netops',
            'group root-lr',
            'group cisco-support',
            'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1',
            '!',
            'username netops2',
            'group root-lr',
            'group cisco-support',
            'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1',
            '!',
            'username netops3',
            'group root-lr',
            'group cisco-support',
            'secret 5 $1$7kTu$zjrgqbgW08vEXsYzUycXw1',
            '!',
```

```
            'cdp',
            'service cli interactive disable',
            'interface MgmtEth0/RP0/CPU0/0',
            'ipv4 address 11.11.11.59 255.255.255.0',
            '!',
            'interface TenGigE0/0/0/0/24',
            'shutdown',
            '!',
            'interface TenGigE0/0/0/0/25',
            'shutdown',
            '!',

            'router static',
            'address-family ipv4 unicast',
            '0.0.0.0/0 11.11.11.2',
            '!',
            '!',
            'end'],
 'status': 'success'}

###### Apply valid configuration using a file ######

Building configuration...
{'status': 'success', 'output': ['!! IOS XR Configuration version = 6.2.1.21I', 'hostname
customer', 'cdp', 'end']}

###### Apply valid configuration using a string ######

Building configuration...
{'output': ['!! IOS XR Configuration version = 6.2.1.21I',
            'hostname customer2',
            'end'],
 'status': 'success'}

###### Apply invalid configuration using a string ######

{'output': ['!! SYNTAX/AUTHORIZATION ERRORS: This configuration failed due to',
            '!! one or more of the following reasons:',
            '!!  - the entered commands do not exist,',
            '!!  - the entered commands have errors in their syntax,',
            '!!  - the software packages containing the commands are not active,',
```

Set Up DHCP Server for ZTP

For ZTP to operate a valid IPv4 or IPv6 address is required and the DHCP server must send a pointer to the configuration script.

The DHCP request from the router has the following DHCP options to identify itself:

- **Option 60**: “vendor-class-identifier” : Used to Identify the following four elements:
  - The type of client: For example, PXEClient
  - The architecture of The system (Arch): For example: 00009 Identify an EFI system using a x86-64 CPU
  - The Universal Network Driver Interface (UNDI):

    For example 003010 (first 3 octets identify the major version and last 3 octets identify the minor version)

    - The Product Identifier (PID):

- **Option 61**: “dhcp-client-identifier” : Used to identify the Serial Number of the device.
- **Option 66** : Used to request the TFTP server name.
- **Option 67**: Used request the TFTP filename.
- **Option 97**: “uuid” : Used to identify the Universally Unique Identifier a 128-bit value (not usable at this time)

Example

The following DHCP request sample provides a fixed IP address and a configuration file with the mac address of the management interface.

```
host cisco-rp0 {
   hardware ethernet e4:c7:22:be:10:ba;
   fixed-address 172.30.12.54;
   filename "http://172.30.0.22/configs/cisco-1.config";
}
```

The following DHCP request sample provides a fixed IP address and a configuration file with the mac address of the management interface along with capability to re-image the system using iPXE (exr-config option):

```
host cisco-rp0 {
   hardware ethernet e4:c7:22:be:10:ba;
   fixed-address 172.30.12.54;
   if exists user-class and option user-class = "iPXE" {
      filename = "http://172.30.0.22/boot.ipxe";
   } elsif exists user-class and option user-class = "exr-config" {
      filename = "http://172.30.0.22/scripts/cisco-rp0_ztp.sh";
   }
}
```

Important In Cisco IOS XR Release 7.3.1 and earlier, the system accepts the device sending **user-class = "exr-config"**; however starting Cisco IOS XR Release 7.3.2 and later, you must use only **user-class = "xr-config"**.

In Cisco IOS XR Release 7.3.2 and later, use:

```
host cisco-rp0 {
   hardware ethernet e4:c7:22:be:10:ba;
   fixed-address 172.30.12.54;
   if exists user-class and option user-class = "iPXE" {
      filename = "http://172.30.0.22/boot.ipxe";
   } elsif exists user-class and option user-class = "xr-config" {
      filename = "http://172.30.0.22/scripts/cisco-rp0_ztp.sh";
   }
}
```

Also, when upgrading from any release that is Cisco IOS XR Release 7.3.1 or earlier to Cisco IOS XR Release 7.3.2 or later release, use the following:

```
host cisco-rp0 {
   hardware ethernet e4:c7:22:be:10:ba;
   fixed-address 172.30.12.54;
   if exists user-class and option user-class = "iPXE" {
      filename = "http://172.30.0.22/boot.ipxe";
   } elsif exists user-class and option user-class = "exr-config" {
      filename = "http://172.30.0.22/scripts/cisco-rp0_ztp.sh";
   }
}
```

DHCP server identifies the device and responds with either an IOS-XR configuration file or a ZTP script as the filename option.

The DHCP server responds with the following DHCP options:

- DHCPv4 using BOOTP filename to supply script/config location.
- DHCPv4 using Option 67 (bootfile-name) to supply script/config location.
- DHCPv6 using Option 59 (OPT_BOOTFILE_URL) to supply script/config location

The following sample shows the DHCP response with bootfile-name (option 67):

```
option space cisco-vendor-id-vendor-class code width 1 length width 1;
option vendor-class.cisco-vendor-id-vendor-class code 9 = {string};

######### Network 11.11.11.0/24 ################
shared-network 11-11-11-0 {

####### Pools ##############
  subnet 11.11.11.0 netmask 255.255.255.0 {
    option subnet-mask 255.255.255.0;
 option broadcast-address 11.11.11.255;
 option routers 11.11.11.2;
 option domain-name-servers 11.11.11.2;
 option domain-name "cisco.local";
 # DDNS statements
   ddns-domainname "cisco.local.";
 # use this domain name to update A RR (forward map)
   ddns-rev-domainname "in-addr.arpa.";
   # use this domain name to update PTR RR (reverse map)

    }
```

```
######## Matching Classes ##########

  class "cisco" {
       match if (substring(option dhcp-client-identifier,0,11) = "FGE194714QS");
  }


  pool {
      allow members of "cisco";
      range 11.11.11.47 11.11.11.50;
      next-server 11.11.11.2;

      if exists user-class and option user-class = "iPXE" {
         filename="http://11.11.11.2:9090/cisco-mini-x-6.2.25.10I.iso";
      }


      if exists user-class and option user-class = "exr-config"
       {
          if (substring(option vendor-class.cisco-vendor-id-vendor-class,19,99)="cisco")
          {
           option bootfile-name "http://11.11.11.2:9090/scripts/exhaustive_ztp_script.py";

          }
        }


      ddns-hostname "cisco-local";
      option routers 11.11.11.2;
  }
}
```

Authentication on Data Ports

On fresh boot, ZTP process is initiated from management ports and may switch to data ports. To validate the connection with DHCP server, authentication is performed on data ports through DHCP option 43 for IPv4 and option 17 for IPv6. These DHCP options are defined in option space and are included within **dhcpd.conf** and **dhcpd6.conf** configuration files. You must provide following parameters for authentication while defining option space:

- Authentication code—The authentication code is either 0 or 1; where 0 indicates that authentication is not required, and 1 indicates that MD5 checksum is required.
- Client identifier—The client identifier must be 'exr-config'.
- MD5 checksum—This is chassis serial number. It can be obtained using **echo -n $SERIALNUMBER | md5sum | awk '{print $1}'**.

Here is the sample **dhcpd.conf** configuration. In the example below, the option space called **VendorInfo** is defined with three parameters for authentication:

```
class "vendor-classes" {
   match option vendor-class-identifier;
}

option space VendorInfo;
option VendorInfo.clientId code 1 = string;
option VendorInfo.authCode code 2 = unsigned integer 8;
option VendorInfo.md5sum code 3 = string
option vendor-specific code 43 = encapsulate VendorInfo;
```

```
subnet 10.65.2.0 netmask 255.255.255.0 {
  option subnet-mask 255.255.255.0;
  option routers 10.65.2.1;
  range 10.65.2.1 10.65.2.200;
}
host cisco-mgmt {
   hardware ethernet 00:50:60:45:67:01;
   fixed-address 10.65.2.39;
   vendor-option-space VendorInfo;
   option VendorInfo.clientId "exr-config";
   option VendorInfo.authCode 1;
   option VendorInfo.md5sum "aedf5c457c36390c664f5942ac1ae3829";
   option bootfile-name "http://10.65.2.1:8800/admin-cmd.sh";
}
```

Here is the sample **dhcpd6.conf** configuration file. In the example below, the option space called **VendorInfo** is defined that has code width 2 and length width 2 (as per dhcp standard for IPv6) with three parameters for authentication:

```
log-facility local7;
option dhcp6.name-servers 2001:1451:c632:1::1;
option dhcp6.domain-search "cisco.com";
dhcpv6-lease-file-name "/var/lib/dhcpd/dhcpd6.leases";
option dhcp6.info-refresh-time 21600;
option dhcp6.bootfile-url code 59 = string;
option dhcp6.user-class code 15 = string;
option space CISCO-EXR-CONFIG code width 2 length width 2;
option CISCO-EXR-CONFIG.client-identifier code 1 = string;
option CISCO-EXR-CONFIG.authCode code 2 = integer 8;
option CISCO-EXR-CONFIG.md5sum code 3 = string;
option vsio.CISCO-EXR-CONFIG code 9 = encapsulate CISCO-EXR-CONFIG;
subnet6 2001:1451:c632:1::/64{
 range6 2001:1451:c632:1::2 2001:1451:c632:1::9;
 option CISCO-EXR-CONFIG.client-identifier "exr-config";
 option CISCO-EXR-CONFIG.authCode 1;
 #valid md5
 option CISCO-EXR-CONFIG.md5sum "90fd845ac82c77f834d57a034658d0f0";
 if option dhcp6.user-class = 00:04:69:50:58:45 {
  option dhcp6.bootfile-url "http://[2001:1851:c632:1::1]/cisco-2/image.iso";
 }
 else {
   #option dhcp6.bootfile-url "http://[2001:1851:c632:1::1]/cisco-2/cisco-mini-x.iso.sh";
   option dhcp6.bootfile-url "http://[2001:1851:c632:1::1]/cisco-2/ztp.cfg";
  }
}
```

Manual ZTP Invocation

You can ZTP manually through Command Line Interdace. This method is Ideal for verifying the ZTP configuration without a reboot. This manual way helps you to provision the router in stages. If you would like to invoke a ZTP on an interfaces (data ports or management port), you don't have to bring up and configure the interface first.

You can execute the ztp initiate command, even if the interface is down, ZTP script will bring it up and invoke dhclient. So ZTP could run over all interfaces no matter it is up or down.

Use the following commands to manually execute the ZTP commands to force ZTP to run over more interfaces:

- **ztp initiate**— Invokes a new ZTP DHCP session. Logs can be found in **/disk0:/ztp/ztp.log**.

Configuration Example:

```
Router#ztp initiate debug verbose interface HundredGigE 0/0/0/24
Invoke ZTP? (this may change your configuration) [confirm] [y/n] :
```

- **ztp terminate**—Terminates any ZTP session in progress.

  Configuration Example:

  ```
  Router #ztp terminate verbose
  Mon Oct 10 16:52:38.507 UTC
  Terminate ZTP? (this may leave your system in a partially configured state) [confirm]
  [y/n] :y
  ZTP terminated
  ```

- **ztp enable**—Enables the ZTP at boot.

  Configuration Example:

  ```
  Router#ztp enable
  Fri Jul 12 16:09:02.154 UTC
  Enable ZTP? [confirm] [y/n] :y
  ZTP Enabled.
  ```

- **ztp disable**—Disables the ZTP at boot.

  Configuration Example:

  ```
  Router#ztp disable
  Fri Jul 12 16:07:18.491 UTC
  Disable ZTP? [confirm] [y/n] :y
  ZTP Disabled.
  Run ZTP enable to run ZTP again.
  ```

- **ztp clean**—Removes only the ZTP state files.

  Configuration Example:

  ```
  Router#ztp clean verbose
  Mon Oct 10 17:03:43.581 UTC
  Remove all ZTP temporary files and logs? [confirm] [y/n] :y
  All ZTP files have been removed.
  If you now wish ZTP to run again from boot, do 'conf t/commit replace' followed by
  reload.
  ```

  The log file ztp.log is saved in **/var/log** folder, and a copy of log file is available at **/disk0:/ztp/ztp.log** location using a soft link. However, executing **ztp clean** clears files saved on disk and not on **/var/log** folder where current ZTP logs are saved. In order to have a log from current ZTP run, you must manually clear the ZTP log file from **/var/log/** folder.

Configuration

This task shows the most common use case of manual ZTP invocation: invoke ZTP.

1. Invoke DHCP sessions on all data ports which are up or could be brought up. ZTP runs in the background. Use show logging or look at /disk0:/ztp/ztp.log to check progress.

Configuration Example:

```
Router# ztp initiate dataport
```

Configure ZTP BootScript

If you want to hard code a script to be executed every boot, configure the following.

```
Router#configure
Router(config)#ztp bootscript /disk0:/myscript
Router(config)#commit
```

The above configuration will wait for the first data-plane interface to be configured and then wait an additional minute for the management interface to be configured with an IP address, to ensure that we have connectivity in the third party namespace for applications to use. If the delay is not desired, use:

```
Router#configure
Router(config)#ztp bootscript preip /disk0:/myscript
Router(config)#commit
```

Note When the above command is first configured, you will be prompted if you wish to invoke it now. The prompt helps with testing.

This is the example content of **/disk0:/myscript**:

```
#!/bin/bash
exec &> /dev/console # send logs to console
source /pkg/bin/ztp_helper.sh

# If we want to only run one time:
xrcmd "show running" | grep -q myhostname
if [[ $? -eq 0 ]]; then
    echo Already configured
fi

# Set the hostname
cat >/tmp/config <<%%
!! XR config example
hostname myhostname
%%
xrapply /tmp/config

#
# Force an invoke of ZTP again. If there was a username normally it would not run. This
forces it.
# Kill off ztp if it is running already and suppress errors to the console when ztp runs
below and
# cleans up xrcmd that invokes it. ztp will continue to run however.
#
xrcmd "ztp terminate noprompt" 2>/dev/null
xrcmd "ztp initiate noprompt" 2>/dev/null
```

Customize the ZTP Configurable Options

You can customize the following ZTP configurable options in the *ztp.ini* file:

- `ZTP`: You can enable or disable ZTP at boot using CLI or by editing the *ztp.ini* file.
- `Retry`: Set the ZTP DHCP retry mechanism: The available values are infinite and once.
- `Fetcher Priority`: Fetcher defines which port ZTP should use to get the provisioning details. By default, each port has a fetcher priority defined in the *ztp.ini* file. You can modify the default priority of the fetcher. Allowed range is from 0 to 9.

Note Lower the number higher the priority. The value 0 has the highest priority and 9 has the lowest priority.

In the following example, the Mgmt4 port has the highest priority:

```
[Fetcher Priority]
Mgmt4: 0
Mgmt6: 1
DPort4: 2
DPort6: 3
```

- `progress_bar`: Enable progress bar on the console. By default, the progress bar is disabled. To enable the progress bar, add the following entry in the `ztp.ini` file.

```
[Options]
progress_bar: True
```

By default, the `ztp.ini` file is located in the `/pkg/etc/` location. To modify the ZTP configurable options, make a copy of the file in the `/disk0:/ztp/` directory and then edit the `ztp.ini` file.

To reset to the default options, delete the ztp.ini file in the `/disk0:/ztp/` directory.

Note Do not edit or delete the `ztp.ini` file in the `/pkg/etc/` location to avoid issues during installation.

The following example shows the sample of the `ztp.ini` file:

```
[Startup]
start: True
retry_forever: True

[Fetcher Priority]
Mgmt4:  1
Mgmt6:  2
DPort4: 3
DPort6: 4
```

Enable ZTP Using CLI

If you want to enable ZTP using CLI, use the **ztp enable** command.

Configuration example

```
Router#ztp enable
Fri Jul 12 16:09:02.154 UTC
Enable ZTP? [confirm] [y/n] :y
ZTP Enabled.
```

Disable ZTP Using CLI

If you want to disable ZTP using CLI, use the **ztp disable** command.

Configuration example

```
Router#ztp disable
Fri Jul 12 16:07:18.491 UTC
Disable ZTP? [confirm] [y/n] :y
ZTP Disabled.
Run ZTP enable to run ZTP again.
```