System Management Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.2.x

First Published: 2017-05-01

Last Modified: 2017-07-01

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

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.

Changes to this Document

Table 1: Changes to this Document

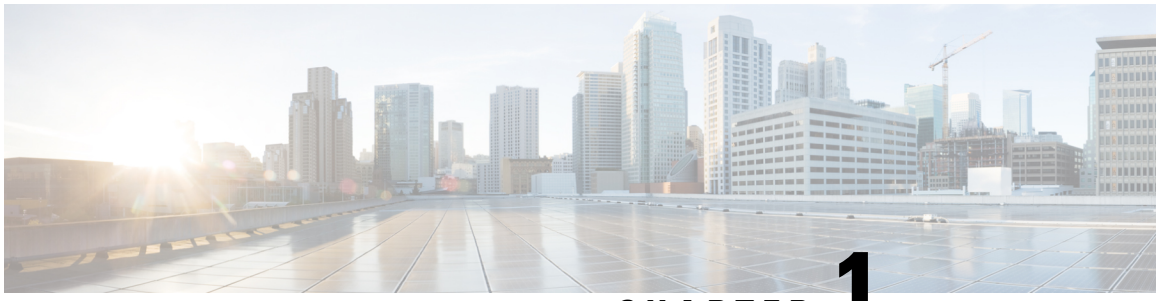
Date	Summary
March 2017	Initial release of this document.
July 2017	Republished for Release 6.2.2.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.



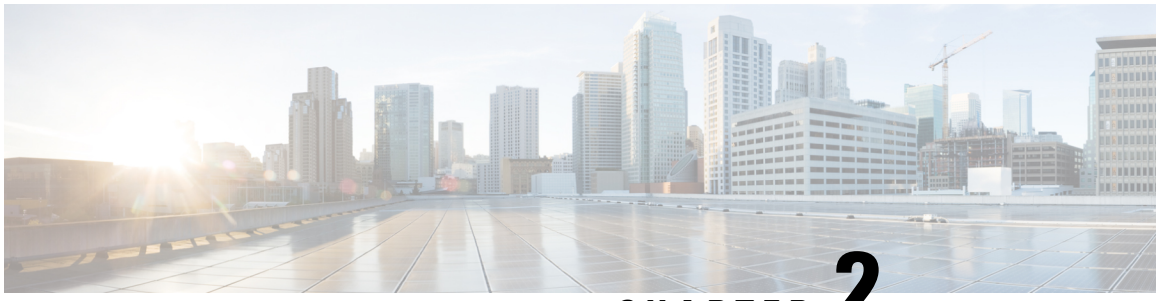
CHAPTER 1

New and Changed System Management Features

This chapter lists all the features that have been added or modified in this guide. The table also contains references to these feature documentation sections.

System Management Features Added or Modified in IOS XR Release 6.2.x

Feature	Description	Changed in Release	Where Documented
None	No new features introduced	Not applicable	Not applicable



CHAPTER 2

Configuring Flexible Command Line Interface

This module describes how to configure and use flexible command line interface (CLI) configuration groups.

Flexible CLI Configuration Groups

Flexible command line interface (CLI) configuration groups provide the ability to minimize repetitive configurations by defining a series of configuration statements in a configuration group, and then applying this group to multiple hierarchical levels in the router configuration tree.

Flexible CLI configuration groups utilize regular expressions that are checked for a match at multiple submodes of the configuration tree based on where the group is applied within the hierarchy. If a match is found at a configuration submode, the corresponding configuration defined in the group is inherited within the matched submode.

Flexible CLI configuration groups also provide an auto-inheritance feature. Auto-inheritance means that any change done to a CLI configuration group is automatically applied to the configuration in any matched submodes that have an apply-group at that hierarchical level. This allows you to make a configuration change or addition once, and have it applied automatically in multiple locations, depending on where you have applied the flexible CLI configuration group.

Flexible Configuration Restrictions

Note these restrictions while using flexible configuration groups:

- Flexible CLI configuration groups are not supported in administration configurations and corresponding apply-groups are not supported in administration configurations.
- Use of preconfigured interfaces in configuration groups is not supported.
- Downgrading from an image that supports configuration groups to an image that does not support them is not supported.

- Access lists, quality of service and route policy configurations do not support the use of configuration groups. Configurations such as these are not valid:

```
group g-not-supported
 ipv4 access-list ...
 !
 ipv6 access-list ...
 !
 ethernet-service access-list ...
 !
 class-map ...
 !
 policy-map ...
 !
 route-policy ...
 !
end-group
```

  You can, however, reference such configurations, as shown in this example:

```
group g-reference-ok
 router bgp 6500
  neighbor 7::7
   remote-as 65000
   bfd fast-detect
   update-source Loopback300
   graceful-restart disable
   address-family ipv6 unicast
       route-policy test1 in
    route-policy test2 out
    soft-reconfiguration inbound always
   !
  !
 !
 interface Bundle-Ether1005
   bandwidth 10000000
   mtu 9188
      service-policy output input_1
   load-interval 30
 !
end-group
```

- Some regular expressions are not supported within groups. For example, '?', '|' and '$,' are not supported within groups. Also some characters such as /d and /w are not supported.
  - The choice operator "|" to express multiple match expressions within a regular expression is not supported. For example, these expressions are not supported:

    `Gig.*|Gig.*\..*`—To match on either Gigabit Ethernet main interfaces or Gigabit Ethernet sub-interfaces.

    `Gig.*0/0/0/[1-5]|Gig.*0/0/0/[10-20]`—To match on either Gig.*0/0/0/[1-5] or Gig.*0/0/0/[10-20].

    `'TenGigE.*|HundredGigE.*`—To match on either TenGigE.* or HundredGigE.* .

- Commands that require a node identifier for the **location** keyword are not supported. For example, this configuration is not supported:

```
lpts pifib hardware police location 0/RP0/CPU0
```

- Overlapping regular expressions within a configuration group for the same configuration are not supported. For example:

```
group G-INTERFACE
interface 'gig.*a.*'
  mtu 1500
!
interface 'gig.*e.* '
  mtu 2000
!
end-group

interface gigabitethernet0/0/0/* ---- where * is 0 to 79 or 0 to 39
  apply-group G-INTERFACE
```

  This configuration is not permitted because it cannot be determined whether the `interface GigabitEthernet0/0/0/*` configuration inherits `mtu 1500` or `mtu 2000`. Both expressions in the configuration group match GigabitEthernet0/0/0/*.

- Up to eight configuration groups are permitted on one apply-group command.

Configuring a Configuration Group

A configuration group includes a series of configuration statements that can be used in multiple hierarchical levels in the router configuration tree. By using regular expressions in a configuration group, you can create generic commands that can be applied in multiple instances.

Use this task to create and use a configuration group.

Note Flexible CLI configurations are not available through the XML interface.

SUMMARY STEPS

1. **configure**
2. **group** *group-name*
3. Enter configuration commands, starting from global configuration mode. Use regular expressions for interface names and other variable instances.
4. **end-group**
5. **apply-group**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters mode.

Step 2 **group** *group-name*

Example:

```
RP/0/RP0/CPU0:router(config)# group g-interf
```

Specifies a name for a configuration group and enters group configuration mode to define the group.The *group-name* argument can have up to 32 characters and cannot contain any special characters.

Step 3 Enter configuration commands, starting from global configuration mode. Use regular expressions for interface names and other variable instances.

Example:

```
RP/0/RP0/CPU0:router(config)# group g-interf
RP/0/RP0/CPU0:router(config-GRP)# interface 'GigabitEthernet.*'
RP/0/RP0/CPU0:router(config-GRP-if)# mtu 1500
```

Specifies the configuration statements that you want included in this configuration group.

For more information regarding the use of regular expressions, see Configuration Group Inheritance with Regular Expressions: Example, on page 18. This example is applicable to all Gigabit Ethernet interfaces.

Step 4 **end-group**

Example:

```
RP/0/RP0/CPU0:router(config-GRP-if)# end-group
```

Completes the configuration of a configuration group and exits to global configuration mode.

Step 5 **apply-group**

Example:

```
RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/2/0/0
RP/0/RP0/CPU0:router(config-if)# apply-group g-interf
```

Adds the configuration of the configuration group into the router configuration applicable at the location that the group is applied. Groups can be applied in multiple locations, and their effect depends on the location and context.

The MTU value from the group g-interf is applied to the interface GigabitEthernet0/2/0/0. If this group is applied in global configuration mode, the MTU value is inherited by all Gigabit Ethernet interfaces that do not have an MTU value configured.

Simple Configuration Group: Example

This example shows how to use configuration groups to add a global configuration to the system:

```
RP/0/RP0/CPU0:router(config)# group g-logging
RP/0/RP0/CPU0:router(config-GRP)# logging trap notifications
RP/0/RP0/CPU0:router(config-GRP)# logging console debugging
RP/0/RP0/CPU0:router(config-GRP)# logging monitor debugging
RP/0/RP0/CPU0:router(config-GRP)# logging buffered 10000000
RP/0/RP0/CPU0:router(config-GRP)# end-group
```

```
RP/0/RP0/CPU0:router(config)# apply-group g-logging
```

When this configuration is committed, all commands contained in the g-logging configuration group are committed.

Configuration Group Applied to Different Places: Example

Configuration groups can be applied to different places, and their effect depends on the context within which they are applied. Consider this configuration group:

```
RP/0/RP0/CPU0:router(config)# group g-interfaces
RP/0/RP0/CPU0:router(config-GRP)# interface 'GigabitEthernet.*'
RP/0/RP0/CPU0:router(config-GRP-if)# mtu 1500
RP/0/RP0/CPU0:router(config-GRP-if)# exit
RP/0/RP0/CPU0:router(config-GRP)# interface 'GigabitEthernet.*'
RP/0/RP0/CPU0:router(config-GRP-if)# mtu 1000
RP/0/RP0/CPU0:router(config-GRP-if)# exit
RP/0/RP0/CPU0:router(config-GRP)# interface 'GigabitEthernet.*'
RP/0/RP0/CPU0:router(config-GRP-if)# mtu 2000
RP/0/RP0/CPU0:router(config-GRP-if)# end-group
```

This group can be applied to Gigabit Ethernet interface and in each instance the applicable MTU is applied. For instance, in this example, the Gigabit Ethernet interface is configured to have an MTU of 1000:

```
RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/2/0/0
RP/0/RP0/CPU0:router(config-if)# apply-group g-interfaces
RP/0/RP0/CPU0:router(config-if)# ipv4 address 2.2.2.2 255.255.255.0
```

In this example, the Gigabit Ethernet interface is configured to have an MTU of 1500:

```
RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/2/0/0
RP/0/RP0/CPU0:router(config-if)# apply-group g-interfaces
RP/0/RP0/CPU0:router(config-if)# ipv4 address 3.3.3.3 255.255.255.0
```

The same configuration group is used in both cases, but only the applicable configuration statements are used.

Verifying the Configuration of Configuration Groups

Use this task to verify the router configuration using configuration groups:

SUMMARY STEPS

1. **show running-config group** [*group-name*]
2. **show running-config**
3. **show running-config inheritance**

4. **show running-config interface x/y/z inheritance** *detail*

DETAILED STEPS

	Command or Action	Purpose

Step 1	**show running-config group** [*group-name*]
	Example:
	`RP/0/RP0/CPU0:router# show running-config group`
	`group g-int-ge`
	` interface 'GigabitEthernet.*'`
	`  mtu 1000`
	`  negotiation auto`
	` !`
	`end-group`
		Displays the contents of a specific or all configured configuration groups.

Step 2	**show running-config**
	Example:
	`RP/0/RP0/CPU0:router# show running-config`
	`group G-INTERFACE-MTU`
	` interface 'GigabitEthernet.*'`
	`  mtu 1500`
	` !`
	`end-group`
	`interface interface GigabitEthernet0/4/1/0`
	` apply-group G-INTERFACE-MTU`
	`!`
	`interface interface GigabitEthernet0/4/1/1`
	` apply-group G-INTERFACE-MTU`
	` mtu 2000`
	`!`
		Displays the running configuration. Any applied groups are displayed. There is no indication as to whether these configuration groups affect the actual configuration or not. In this example, although the group G-INTERFACE-MTU is applied to interface GigabitEthernet0/4/1/1, the configured MTU value is 2000 and not 1500. This happens if the command **mtu 2000** is configured directly on the interface. An actual configuration overrides a configuration group configuration if they are the same.

Step 3	**show running-config inheritance**
	Example:
	`RP/0/RP0/CPU0:router# show running-config inheritance`
	`.`
	`.`
	`group G-INTERFACE-MTU`
	` interface 'GigabitEthernet.*'`
	`  mtu 1500`
	` !`
	`end-group`
	`.`
	`.`
	`interface interface GigabitEthernet0/4/1/0`
	` ## Inherited from group G-INTERFACE-MTU`
	` mtu 1500`
	`!`
	`interface interface GigabitEthernet0/4/1/1`
		Displays the inherited configuration where ever a configuration group has been applied.

	Command or Action	Purpose
	mtu 2000 ! . .	

Step 4	**show running-config interface x/y/z inheritance** *detail*
	Example:
	`RP/0/RP0/CPU0:router# show running-config interface interface GigabitEthernet0/4/1/0 inheritance detail`
	`interface interface GigabitEthernet0/4/1/0`
	`## Inherited from group G-INTERFACE-MTU`
	`mtu 1500`
		Displays the inherited configuration for a specific configuration command.

Regular Expressions in Configuration Groups

Regular expressions are used in configuration groups to make them widely applicable. Portable Operating System Interface for UNIX (POSIX) 1003.2 regular expressions are supported in the names of configuration statements. Single quotes must be used to delimit a regular expression.

Note Not all POSIX regular expressions are supported.

Regular Expressions for Interface Identifiers

Configuration groups do not accept exact interface identifiers. You must use a regular expression to identify a group of interfaces that are applicable to the configuration group. The regular expression '.*' is not allowed. You must begin the regular expression for an interface identifier with an unambiguous word, followed by the regular expression. For example, to configure Gigabit Ethernet interfaces, use the regular expression 'GigabitEthernet.*'.

To display a list of available interface types for your router configuration, enter **interface ?** at the configuration group prompt:

```
RP/0/RP0/CPU0:router(config-GRP)# interface ?

  ATM              'RegExp': ATM Network Interface(s)
  BVI              'RegExp': Bridge-Group Virtual Interface
  Bundle-Ether     'RegExp': Aggregated Ethernet interface(s)
  GigabitEthernet  'RegExp': GigabitEthernet/IEEE 802.3 interface(s)
  IMA              'RegExp': ATM Network Interface(s)
  Loopback         'RegExp': Loopback interface(s)
  MgmtEth          'RegExp': Ethernet/IEEE 802.3 interface(s)
  Multilink        'RegExp': Multilink network interface(s)
  Null             'RegExp': Null interface
  PW-Ether         'RegExp': PWHE Ethernet Interface
  PW-IW            'RegExp': PWHE VC11 IP Interworking Interface
  Serial           'RegExp': Serial network interface(s)
  tunnel-ip        'RegExp': GRE/IPinIP Tunnel Interface(s)
```

```
  tunnel-mte       'RegExp': MPLS Traffic Engineering P2MP Tunnel interface(s)
  tunnel-te        'RegExp': MPLS Traffic Engineering Tunnel interface(s)
  tunnel-tp        'RegExp': MPLS Transport Protocol Tunnel interface
```

Note Although you are required to enter only enough characters for the interface type to be unique, it is recommended that you enter the entire phrase. All interface types used in regular expressions are case-sensitive.

To specify a subinterface, prefix the expression with the characters \. (backslash period). For example, use `interface 'GigabitEthernet.*\..*'` to configure all Gigabit Ethernet subinterfaces.

You can specify Layer 2 transport interfaces or point-to-point interfaces as shown in these examples:

```
group g-l2t
  interface 'Gi.*\..*' l2transport
.
.
end-group
group g-ptp
  interface 'Gi.*\..*' point-to-point
.
.
end-group
```

Regular Expressions for an OSPF Configuration

Exact router process names and OSPF areas cannot be used. You must use a regular expression to specify a process name or group of OSPF areas. To specify that the OSFP area can be either a scalar value or an IP address, use the regular expression '.*', as in this example:

```
group g-ospf
router ospf '.*'
area '.*'
mtu-ignore enable
!
!
end-group
```

To specify that the OSPF area must be an IP address, use the expression '\.' as in this example:

```
group g-ospf-ipaddress
router ospf '.*\..*\..*\..*'
area '.*'
passive enable
!
!
 end-group
```

To specify that the OSPF area must be a scalar value, use the expression '1.*', as in this example:

```
group g-ospf-match-number
router ospf '.*'
area '1.*'
passive enable
```

```
!
!
end-group
```

Regular Expressions for a BGP AS

Exact BGP AS values cannot be used in configuration groups. Use a regular expression to specify either AS plain format, or AS dot format as in the format X.Y. To match AS plain format instances, use a simple regular expression. To match AS dot format instances, use two regular expressions separated by a dot, as shown in this example:

```
group g-bgp
router bgp '*'.'*'
address-family ipv4 unicast
!
!
end-group
```

Regular Expressions for ANCP

Exact Access Node Control Protocol (ANCP) sender-name identifiers cannot be used in configuration groups. Because the sender name argument can be either an IP address or a MAC address, you must specify in the regular expression which one is being used. Specify an IP address as `'.*\..*\..*\..*'`; specify a MAC address as `'.*\..*\..*'`.

Resolving to a Uniform Type

Regular expressions must resolve to a uniform type. This is an example of an illegal regular expression:

```
group g-invalid
 interface '.*'
  bundle port-priority 10
 !
interface '.*Ethernet.*'
  bundle port-priority 10
 !
end-group
```

In this example, the **bundle** command is supported for interface type GigabitEthernet but not for interface type 'FastEthernet'. The regular expressions '.*' and '.*Ethernet.*' match both GigabitEthernet and FastEthernet types. Because the **bundle** command is not applicable to both these interface types, they do not resolve to a uniform type and therefore the system does not allow this configuration.

Note If the system cannot determine from the regular expression what the configuration should be, the expression is not considered valid.

Note The regular expression '.*' is not allowed when referring to an interface identifier. You must begin the regular expression for an interface identifier with an unambiguous word, followed by the regular expression. Refer to *Regular Expressions for Interface Identifiers* in this section for more information.

Overlapping Regular Expressions

Regular expressions are used in names of configuration statements within a configuration group. This permits inheritance by the configuration when applied to matching names. Single quotes are used to delimit the regular expression. Overlapping regular expression within a configuration group for the same configuration is permitted.

The example, given below, illustrates the process of creating and applying multiple configuration groups:

```
RP/0//CPU0:router(config)#group FB_flexi_snmp
RP/0//CPU0:router(config-GRP)# snmp-server vrf '.*'
RP/0//CPU0:router(config-GRP-snmp-vrf)# host 1.1.1.1 traps version 2c group_1
RP/0//CPU0:router(config-GRP-snmp-vrf)# host 1.1.1.1 informs version 2c group_1
RP/0//CPU0:router(config-GRP-snmp-vrf)# context group_1

RP/0//CPU0:router(config-GRP-snmp-vrf)#
RP/0//CPU0:router(config-GRP-snmp-vrf)#commit

RP/0//CPU0:router(config-GRP-snmp-vrf)#root
RP/0//CPU0:router(config)#
RP/0//CPU0:router(config)#snmp-server vrf vrf1
RP/0//CPU0:router(config-snmp-vrf)#snmp-server vrf vrf10
RP/0//CPU0:router(config-snmp-vrf)#!
RP/0//CPU0:router(config-snmp-vrf)#snmp-server vrf vrf100
RP/0//CPU0:router(config-snmp-vrf)#
RP/0//CPU0:router(config-snmp-vrf)#commit

RP/0//CPU0:router(config-snmp-vrf)#root
RP/0//CPU0:router(config)#
RP/0//CPU0:router(config)#apply-group FB_flexi_snmp
RP/0//CPU0:router(config)#do sh running-config group
group FB_flexi_snmp
 snmp-server vrf '.*'
  host 1.1.1.1 traps version 2c group_1
  host 1.1.1.1 informs version 2c group_1
  context group_1
 !
end-group
apply-group FB_flexi_snmp
snmp-server vrf vrf1
!
snmp-server vrf vrf10
!
snmp-server vrf vrf100
!
RP/0//CPU0:ios#show running-config inheritance detail


group FB_flexi_snmp
 snmp-server vrf '.*'
  host 1.1.1.1 traps version 2c group_1
  host 1.1.1.1 informs version 2c group_1
  context group_1
 !
end-group
snmp-server vrf vrf1
 ## Inherited from group FB_flexi_snmp
 host 1.1.1.1 traps version 2c group_1
 ## Inherited from group FB_flexi_snmp
 host 1.1.1.1 informs version 2c group_1
 ## Inherited from group FB_flexi_snmp
 context group_1
!
```

```
snmp-server vrf vrf10
 ## Inherited from group FB_flexi_snmp
 host 1.1.1.1 traps version 2c group_1
 ## Inherited from group FB_flexi_snmp
 host 1.1.1.1 informs version 2c group_1
 ## Inherited from group FB_flexi_snmp
 context group_1
!
snmp-server vrf vrf100
 ## Inherited from group FB_flexi_snmp
 host 1.1.1.1 traps version 2c group_1
 ## Inherited from group FB_flexi_snmp
 host 1.1.1.1 informs version 2c group_1
 ## Inherited from group FB_flexi_snmp
 context group_1
```

The example given below demonstrates the regular expression. In this example `snmp-server vrf '.*'` and `snmp-server vrf '[\w]+` are two different regular expressions.

```
group FB_flexi_snmp
snmp-server vrf '.*'
host 1.1.1.1 traps version 2c group_1
host 1.1.1.1 informs version 2c group_1
context group_1
!
snmp-server vrf '[\w]+'
host 2.2.2.2 traps version 2c group_2
host 2.2.2.2 informs version 2c group_2
context group_2
!
end-group
```

This individual regular expression gets combined to all the three expressions - `snmp-server vrf vrf1`, `snmp-server vrf vrf10` and `snmp-server vrf vrf100` as given below.

```
apply-group FB_flexi_snmp
snmp-server vrf vrf1
!
snmp-server vrf vrf10
!
snmp-server vrf vrf100
!
```

In a configuration group, there can be instances of regular expressions overlap. In such cases, the regular expression with the highest priority is activated and inherited, when applied. It has that regular expression, which comes first in the lexicographic order that has the highest priority.

The following example shows how to use overlapping regular expressions and how the expression with higher priority is applied:

```
 group FB_flexi_snmp

 snmp-server vrf '.*'
```

```
 host 1.1.1.1 traps version 2c group_1

 host 1.1.1.1 informs version 2c group_1

 context group_1

!

snmp-server vrf '[\w]+’

 host 2.2.2.2 traps version 2c group_2
 host 2.2.2.2 informs version 2c group_2
 context group_2
!
end-group
```

The expression shown below has the highest priority:

```
 group FB_flexi_snmp

 snmp-server vrf '.*’

 host 1.1.1.1 traps version 2c group_1

 host 1.1.1.1 informs version 2c group_1

 context group_1
```

The examples given above, show two different regular expression `snmp-server vrf '.*’` and `snmp-server vrf '[\w]+'`.

The expression below, shows how these two expressions get merged together:

```
apply-group FB_flexi_snmp

snmp-server vrf vrf1
!
snmp-server vrf vrf10
!
snmp-server vrf vrf100
!
```

Any change in a regular expression with lower priority will not affect the inheritance.

Any changes made to an existing regular expression, which is of less (non-top) priority, it will not have any effect on the inheritance.

```
 snmp-server vrf '[\w]+’

 host 2.2.2.2 traps version 2c group_2
 host 2.2.2.2 informs version 2c group_2
 context group_2
```

The expression with the higher priority gets inherited, as shown below:

```
group FB_flexi_snmp

snmp-server vrf '.*’
```

```
host 1.1.1.1 traps version 2c group_1

host 1.1.1.1 informs version 2c group_1

context group_1
```

Apply Groups Priority Inheritance

Priority governs inheritance.

Note From the Cisco IOS XR, Release 6.3.1 onwards, you are able to enter the Flexible CLI config group definition, **apply-group** and **exclude-group** command in any order as long as the entire commit has all the group definitions needed.

Apply groups priority inheritance helps flexible configuration groups to handle common configuration statements between groups. When multiple configuration groups have common configuration statements, the inheritance priority is such that the configuration statements present in inner groups have precedence over those configuration statements present in outer groups. In case of tiebreakers, the priority is assigned in accordance to the lexicographical order of regular expressions. User defined order of commands are not accepted.

For example, a configuration statement in configuration group ONE has precedence over another group. A configuration statement in configuration group SEVEN is used only if it does not exist in any other group. Within a configuration group, inheritance priority is the longest match.

```
   apply-group SIX SEVEN
    router ospf 0
   apply-group FOUR FIVE
    area 0
   apply-group THREE
    interface GigabitEthernet0/0/0/0
   apply-group ONE TWO

  !
 !
!
```

The above example shows two scenarios. The inner most group (**apply-group ONE TWO**) has the highest priority. Case 1

The first scenario shows which group gets the priority. The example states which group is applied between different configuration groups (different groups with nothing in common). While applying group one (ONE TWO), all the seven groups matches the interface `interface GigabitEthernet0/0/0/0`- is applied.

Case 2

Here, when all have the same (common) configuration, group one will be active. That is `apply-group ONE TWO` is active. If group ONE is deleted, then group TWO will be active.

Configuration Examples Using Regular Expressions

Configuration Group with Regular Expression: Example

This example shows the definition of a configuration group for configuring Gigabit Ethernet interfaces with ISIS routing parameters, using regular expressions for the exact interface:

```
RP/0/RP0/CPU0:router(config)# group g-isis-gige
RP/0/RP0/CPU0:router(config-GRP)# router isis '.*'
RP/0/RP0/CPU0:router(config-GRP-isis)# interface 'GigabitEthernet.*'
RP/0/RP0/CPU0:router(config-GRP-isis-if)# lsp-interval 20
RP/0/RP0/CPU0:router(config-GRP-isis-if)# hello-interval 40
RP/0/RP0/CPU0:router(config-GRP-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-GRP-isis-if-af)# metric 10
RP/0/RP0/CPU0:router(config-GRP-isis-if-af)# end-group
RP/0/RP0/CPU0:router(config)#
```

To illustrate the use of this configuration group, assume that you want to configure these Gigabit Ethernet interfaces with the ISIS routing parameters:

```
router isis green
 interface GigabitEthernet0/0/0/0
  lsp-interval 20
  hello-interval 40
  address-family ipv4 unicast
   metric 10
  !
 !
 interface GigabitEthernet0/0/0/1
  lsp-interval 20
  hello-interval 40
  address-family ipv4 unicast
   metric 10
  !
 !
 interface GigabitEthernet0/0/0/2
  lsp-interval 20
  hello-interval 40
  address-family ipv4 unicast
   metric 10
  !
 !
 interface GigabitEthernet0/0/0/3
  lsp-interval 20
  hello-interval 40
  address-family ipv4 unicast
   metric 10
  !
 !
!
```

There are three possible ways to use the configuration group to configure these interfaces. The first is by applying the group within the interface configuration, as shown here:

```
router isis green
 interface GigabitEthernet0/0/0/0
     apply-group g-isis-gige
```

```
 !
!
interface GigabitEthernet0/0/0/1
    apply-group g-isis-gige
 !
!
interface GigabitEthernet0/0/0/2
    apply-group g-isis-gige
 !
!
interface GigabitEthernet0/0/0/3
    apply-group g-isis-gige
 !
!
```

In this situation, only the interfaces to which you apply the configuration group inherit the configuration.

The second way to configure these interfaces using the configuration group is to apply the configuration group within the **router isis** configuration, as shown here:

```
router isis green
    apply-group g-isis-gige
 interface GigabitEthernet0/0/0/0
 !
 interface GigabitEthernet0/0/0/1
 !
 interface GigabitEthernet0/0/0/2
 !
 interface GigabitEthernet0/0/0/3
 !
!
```

In this way, any other Gigabit Ethernet interfaces that you configure in the ISIS green configuration also inherit these configurations.

The third way to configure these interfaces using the configuration group is to apply the group at the global level as shown here:

```
   apply-group g-isis-gige
router isis green
 interface GigabitEthernet0/0/0/0
 !
 interface GigabitEthernet0/0/0/1
 !
 interface GigabitEthernet0/0/0/2
 !
 interface GigabitEthernet0/0/0/3
 !
!
```

In this example, the configuration of the group is applied to all Gigabit Ethernet interfaces configured for ISIS.

Configuration Group Inheritance with Regular Expressions: Example

Local Configuration Has Precedence Over Configuration Group

An explicit configuration takes precedence over a configuration applied from a configuration group. For example, assume that this configuration is running on the router:

```
router ospf 100
 packet-size 1000
!
```

You configure this configuration group, apply it, and commit it to the configuration.

```
RP/0/RP0/CPU0:router(config)# group g-ospf
RP/0/RP0/CPU0:router(config-GRP)# router ospf '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf)# nsf cisco
RP/0/RP0/CPU0:router(config-GRP-ospf)# packet-size 3000
RP/0/RP0/CPU0:router(config-GRP-ospf)# end-group

RP/0/RP0/CPU0:router(config)# apply-group g-ospf
```

The result is effectively this configuration:

```
router ospf 100
 packet-size 1000
 nsf cisco
```

Note that `packet-size 3000` is not inherited from the configuration group because the explicit local configuration has precedence.

Compatible Configuration Is Inherited

The configuration in the configuration group must match the configuration on the router to be inherited. If the configuration does not match, it is not inherited. For example, assume that this configuration is running on the router:

```
router ospf 100
 auto-cost disable
!
```

You configure this configuration and commit it to the configuration.

```
RP/0/RP0/CPU0:router(config)# group g-ospf
RP/0/RP0/CPU0:router(config-GRP)# router ospf '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf)# area '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf-ar)# packet-size 2000
RP/0/RP0/CPU0:router(config-GRP-ospf)# end-group

RP/0/RP0/CPU0:router(config)# apply-group g-ospf

RP/0/RP0/CPU0:router(config)# router ospf 200
RP/0/RP0/CPU0:router(config-ospf)# area 1
```

The result is effectively this configuration:

```
router ospf 100
  auto-cost disable

router ospf 200
  area 1
  packet-size 2000
```

The packet size is inherited by the ospf 200 configuration, but not by the ospf 100 configuration because the area is not configured.

Layer 2 Transport Configuration Group: Example

This example shows how to configure and apply a configuration group with Layer 2 transport subinterfaces:

```
RP/0/RP0/CPU0:router(config)# group g-l2trans-if
RP/0/RP0/CPU0:router(config-GRP)# interface 'TenGigE.*\..*' l2transport
RP/0/RP0/CPU0:router(config-GRP)# mtu 1514
RP/0/RP0/CPU0:router(config-GRP)# end-group

RP/0/RP0/CPU0:router(config)# interface TenGigE0/0/0/0.1 l2transport
RP/0/RP0/CPU0:router(config-if)# apply-group g-l2trans-if
```

When this configuration is committed, the Ten Gigabit Ethernet interface 0/0/0/0.1 inherits the 1514 MTU value. This is the output displayed from the **show running-config inheritence** command for the Ten Gigabit Ethernet interface:

```
interface TenGigE0/0/0/0.1 l2transport
 ## Inherited from group g-l2trans-if
 mtu 1514
!
```

Configuration Group Precedence: Example

When similar configuration statements are contained in multiple configuration groups, groups applied in inner configuration modes take precedence over groups applied in outer modes. This example shows two configuration groups that configure different cost values for OSPF.

```
RP/0/RP0/CPU0:router(config)# group g-ospf2
RP/0/RP0/CPU0:router(config-GRP)# router ospf '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf)# area '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf-ar)# cost 2
RP/0/RP0/CPU0:router(config-GRP-ospf-ar)# end-group

RP/0/RP0/CPU0:router(config)# group g-ospf100
RP/0/RP0/CPU0:router(config-GRP)# router ospf '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf)# area '.*'
RP/0/RP0/CPU0:router(config-GRP-ospf-ar)# cost 100
RP/0/RP0/CPU0:router(config-GRP-ospf-ar)# end-group
```

If these configuration groups are applied as follows, the cost 2 specified in g-ospf2 is inherited by OSPF area 0 because the group is applied in a more inner configuration mode. In this case, the configuration in group g-ospf100 is ignored.

```
RP/0/RP0/CPU0:router(config)# router ospf 0
RP/0/RP0/CPU0:router(config-ospf)# apply-group g-ospf100
RP/0/RP0/CPU0:router(config-ospf)# area 0
RP/0/RP0/CPU0:router(config-ospf-ar)# apply-group g-ospf2
```

Changes to Configuration Group are Automatically Inherited: Example

When you make changes to a configuration group that is committed and applied to your router configuration, the changes are automatically inherited by the router configuration. For example, assume that this configuration is committed:

```
group g-interface-mtu
 interface 'GigabitEthernet.*'
  mtu 1500
 !
end-group

interface POS0/4/1/0
 apply-group g-interface-mtu
!
```

Now you change the configuration group as in this example:

```
RP/0/RP0/CPU0:router(config)# group g-interface-mtu
RP/0/RP0/CPU0:router(config-GRP)# interface 'GigabitEthernet.*'
RP/0/RP0/CPU0:router(config-GRP-if)# mtu 2000
RP/0/RP0/CPU0:router(config-GRP-if)# end-group
```

When this configuration group is committed, the MTU configuration for interface GigabitEthernet0/4/1/0 is automatically updated to 2000.

Configuration Examples for Flexible CLI Configuration

Basic Flexible CLI Configuration: Example

This example shows that the Media Access Control (MAC) accounting configuration from the gd21 configuration group is applied to all Gigabit Ethernet interfaces in slot 2, ports 1 to 9.

1. Configure the configuration group that configures MAC accounting:

```
RP/0/RP0/CPU0:router# show running group gd21

group gd21
interface 'GigabitEthernet0/0/0/2[1-9]'
description general interface inheritance check
```

```
load-interval 30
mac-accounting ingress
mac-accounting egress
!
end-group
```

2. Check that the corresponding apply-group is configured in global configuration or somewhere in the hierarchy:

```
RP/0/RP0/CPU0:router# show running | in apply-group gd21

Building configuration...
apply-group gd21
```

3. Check the concise local view of the configuration of some of the interfaces:

```
RP/0/RP0/CPU0:router# show running interface

interface GigabiEthernet0/0/0/21
!
interface GigabitEthernet0/0/0/22
!
```

4. Verify that the match and inheritance occur on these interfaces:

```
RP/0/RP0/CPU0:router# show running-config inheritance interface

interface GigabitEthernet0/0/0/21
## Inherited from group gd21
description general interface inheritance check
## Inherited from group gd21
load-interval 30
## Inherited from group gd21
mac-accounting ingress
## Inherited from group gd21
mac-accounting egress
!
Interface GigabitEthernet0/0/0/22
## Inherited from group gd21
description general interface inheritance check
## Inherited from group gd21
load-interval 30
## Inherited from group gd21
mac-accounting ingress
## Inherited from group gd21
mac-accounting egress
!
!
```

5. Verify that the inherited configuration actually takes effect:

```
RP/0/RP0/CPU0:router# show mac-accounting GigabitEthernet0/0/0/21

GigabitEthernet0/0/0/21
  Input (96 free)
    6c9c.ed35.90fd:  1271 packets, 98426 bytes
        Total: 1271 packets, 98426 bytes
  Output (96 free)
    6c9c.ed35.90fd:   774 packets, 63265 bytes
```

```
                Total: 774 packets, 63264 bytes
```

Interface MTU Settings for Different Interface Types: Example

This example shows that an MTU value is configured on different interface types.

1. Configure an interface MTU configuration group and apply this group:

```
RP/0/RP0/CPU0:router# show running group l2tr

group l2tr
interface 'GigabitEthernet0/0/0/3.*'
mtu 1500
!
interface 'GigabitEthernet0/0/0/9\..*'
mtu 1400
!
interface 'GigabitEthernet0/0/0/9\..*' l2transport
mtu 1400
!
end-group

RP/0/RP0/CPU0:router# show running | inc apply-group

Building configuration...

apply-group l2tr
```

2. Check the concise view and the inheritance view of the various interfaces:

```
RP/0/RP0/CPU0:router# show running interface gigabitEthernet0/0/0/30

interface GigabitEthernet0/0/0/30
!
RP/0/RP0/CPU0:router# show running interface gigabitEthernet0/0/0/30  inheritance detail

interface GigabitEthernet0/0/0/30
## Inherited from group l2tr
mtu 1500
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet0/0/0/9.800

interface GigabitEthernet0/0/0/9.800
  encapsulation dot1q 800
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet0/0/0/9.800 inheritance
detail

interface GigabitEthernet0/0/0/9.800
## Inherited from group l2tr
mtu 1400
encapsulation dot1q800
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet0/0/0/9.250
```

```
interface GigabitEthernet0/0/0/9.250 l2transport
  encapsulation dot1q 250
!

RP/0/RP0/CPU0:router# show running interface  gigabitEthernet0/0/0/9.800 inheritance
detail

interface GigabitEthernet0/0/0/9.250 l2transport
encapsulation dot1q250
## Inherited from group l2tr
mtu 1400
!
```

3. Verify that the correct values from the group do take effect:

```
RP/0/RP0/CPU0:router# show interface gigabitEthernet 0/0/0/30

GigabitEthernet0/0/0/30 is down, line protocol is down
  Interface state transitions: 0
  Hardware is GigabitEthernet, address is 0026.9824.ee56 (bia 0026.9824.ee56)
  Internet address is Unknown
  MTU 1500 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 1000Mb/s, link type is force-up
  output flow control is off, input flow control is off
  loopback not set,
  Last input never, output never
  Last clearing of "show interface" counters never
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out

RP/0/RP0/CPU0:router# show interface gigabitEthernet 0/0/0/9.801

GigabitEthernet0/0/0/9.801 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is 0026.9824.ee41
  Internet address is Unknown
  MTU 1400 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation 802.1Q Virtual LAN, VLAN Id 801,  loopback not set,
  Last input never, output never
  Last clearing of "show interface" counters never
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
     0 packets output, 0 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
```

```
RP/0/RP0/CPU0:router# show interface gigabitEthernet 0/0/0/9.250

GigabitEthernet0/0/0/9.250 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is 0026.9824.ee41
  Layer 2 Transport Mode
  MTU 1400 bytes, BW 1000000 Kbit (Max: 1000000 Kbit)
     reliability Unknown, txload Unknown, rxload Unknown
  Encapsulation 802.1Q Virtual LAN,
    Outer Match: Dot1Q VLAN 250
    Ethertype Any, MAC Match src any, dest any
  loopback not set,
  Last input never, output never
  Last clearing of "show interface" counters never
     0 packets input, 0 bytes
     0 input drops, 0 queue drops, 0 input errors
     0 packets output, 0 bytes

     0 output drops, 0 queue drops, 0 output errors
```

ACL Referencing: Example

This example shows how to reference access-lists on a number of interfaces using configuration groups.

1. Configure the configuration group and apply-group:

```
RP/0/RP0/CPU0:router# show running group acref

group acref
 interface 'GigabitEthernet0/0/0/3.*'
  ipv4 access-group adem ingress
  ipv4 access-group adem egress
 !
end-group

RP/0/RP0/CPU0:router# show running | inc apply-group

Building configuration...

apply-group isis l2tr isis2 mpp bundle1 acref
```

2. Check the concise and inheritance view of the matching configurations:

```
RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/30

interface GigabitEthernet0/0/0/30
!

RP/0/RP0/CPU0:router# show running interface GigabitEthernet 0/0/0/30 inheritance detail

interface GigabitEthernet0/0/0/30
 ## Inherited from group l2tr
 mtu 1500
 ## Inherited from group acref
 ipv4 access-group adem ingress
 ## Inherited from group acref
```

```
 ipv4 access-group adem egress
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/31

interface GigabitEthernet0/0/0/31
!

RP/0/RP0/CPU0:router# show running interface GigabitEthernet 0/0/0/31 inheritance detail

interface GigabitEthernet0/0/0/31
 ## Inherited from group l2tr
 mtu 1500
 ## Inherited from group acref
 ipv4 access-group adem ingress
 ## Inherited from group acref
 ipv4 access-group adem egress
```

3. Check that the ACL group configuration actually got configured by using a traffic generator and watching that denied traffic is dropped.

Local Configuration Takes Precedence: Example

This example illustrates that local configurations take precedence when there is a discrepancy between a local configuration and the configuration inherited from a configuration group.

1. Configure a local configuration in a configuration submode with an access list:

```
RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/39

interface GigabitEthernet0/0/0/39
 ipv4 access-group smany ingress
 ipv4 access-group smany egress
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/38

interface GigabitEthernet0/0/0/38
!

RP/0/RP0/CPU0:router# show running ipv4 access-list smany

ipv4 access-list smany
 10 permit ipv4 any any
!

RP/0/RP0/CPU0:router# show running ipv4 access-list adem

ipv4 access-list adem
 10 permit ipv4 21.0.0.0 0.255.255.255 host 55.55.55.55
 20 deny ipv4 any any
!
```

2. Configure and apply the access list group configuration:

```
RP/0/RP0/CPU0:router# show running group acref

group acref
```

```
 interface 'GigabitEthernet0/0/0/3.*'
  ipv4 access-group adem ingress
  ipv4 access-group adem egress
 !
end-group

RP/0/RP0/CPU0:router# show running | inc apply-group

Building configuration...
apply-group isis l2tr isis2 mpp bundle1 acref
```

3. Check the concise and inheritance views for the matching interface where the access list reference is configured locally:

```
RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/39

interface GigabitEthernet0/0/0/39
 ipv4 access-group smany ingress
 ipv4 access-group smany egress
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/39 inheritance detail

interface GigabitEthernet0/0/0/39
 ## Inherited from group l2tr
 mtu 1500
 ipv4 access-group smany ingress
 ipv4 access-group smany egress     << no config inherited, local config prioritized
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/38

interface GigabitEthernet0/0/0/38
!

RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/0/0/38 inheritance detail

interface GigabitEthernet0/0/0/38
 ## Inherited from group l2tr
 mtu 1500
 ## Inherited from group acref
 ipv4 access-group adem ingress
 ## Inherited from group acref
 ipv4 access-group adem egress
!
```

4. Use a traffic generator to verify that the traffic pattern for interface GigabitEthernet0/0/0/39 gets acted on by the access list in the local configuration (smany) and not according to the inherited referenced access list (adem).

ISIS Hierarchical Configuration: Example

This example illustrates inheritance and priority handling with two ISIS groups using an ISIS configuration.

1. Configure the local ISIS configuration:

```
RP/0/RP0/CPU0:router# show running router isis

router isis vink
 net 49.0011.2222.2222.2222.00
 address-family ipv4 unicast
  mpls traffic-eng level-1-2
  mpls traffic-eng router-id Loopback0
  redistribute connected
 !
 interface Bundle-Ether1
  address-family ipv4 unicast
  !
 !
 interface Bundle-Ether2
 !
 interface Loopback0
 !
 interface TenGigE0/2/0/0.3521
  address-family ipv4 unicast
  !
 !
 interface TenGigE0/2/0/0.3522
  address-family ipv4 unicast
  !
 !
 interface TenGigE0/2/0/0.3523
  address-family ipv4 unicast
  !
 !
 interface TenGigE0/2/0/0.3524
  address-family ipv4 unicast
  !
 !
 interface TenGigE0/2/0/0.3525
  address-family ipv4 unicast
  !
 !
 interface TenGigE0/2/0/0.3526
 !
 interface TenGigE0/2/0/0.3527
 !
 interface TenGigE0/2/0/0.3528
 !
 interface TenGigE0/2/0/1
  address-family ipv4 unicast
  !
 !
!
```

2. Configure two ISIS groups and apply these to the configuration:

```
RP/0/RP0/CPU0:router# show running group isis

group isis
 router isis '.*'
  address-family ipv4 unicast
   mpls traffic-eng level-1-2
   mpls traffic-eng router-id Loopback0
   redistribute connected
   redistribute ospf 1 level-1-2
  !
```

```
  interface 'TenGig.*'
   lsp-interval 40
   hello-interval 15
   address-family ipv4 unicast
    metric 50
   !
  !
  interface 'Bundle-Ether.*'
   address-family ipv4 unicast
    metric 55
   !
  !
 !
end-group

RP/0/RP0/CPU0:router# show running group isis2

group isis2
 router isis '.*'
 !
 router isis '^(vink)'
  address-family ipv4 unicast
  !
  interface '(^Ten)Gig.*'
  !
  interface '^(Ten)Gig.*'
   address-family ipv4 unicast
    metric 66
   !
  !
 !
end-group

RP/0/RP0/CPU0:router# show running | inc apply-group

Building configuration...

apply-group isis  l2tr  isis2  mpp  bundle1  acref
```

3. Check the inheritance view of the ISIS configuration:

```
RP/0/RP0/CPU0:router# show running router isis inheritance detail

router isis vink
 net 49.0011.2222.2222.2222.00
 address-family ipv4 unicast
  mpls traffic-eng level-1-2
  mpls traffic-eng router-id Loopback0
  redistribute connected
  ## Inherited from group isis
  redistribute ospf 1 level-1-2
 !
 interface Bundle-Ether1
  address-family ipv4 unicast
   ## Inherited from group isis
   metric 55
  !
 !
 interface Bundle-Ether2
  ## Inherited from group isis
  address-family ipv4 unicast
   ## Inherited from group isis
```

```
  metric 55
 !
!
interface Loopback0
!
interface TenGigE0/2/0/0.3521
 ## Inherited from group isis
 lsp-interval 40
 ## Inherited from group isis
 hello-interval 15
 address-family ipv4 unicast
  ## Inherited from group isis
  metric 50
 !
!
interface TenGigE0/2/0/0.3522
 ## Inherited from group isis
 lsp-interval 40
 ## Inherited from group isis
 hello-interval 15
 address-family ipv4 unicast
  ## Inherited from group isis
  metric 50
 !
!
interface TenGigE0/2/0/0.3523
 ## Inherited from group isis
 lsp-interval 40
 ## Inherited from group isis
 hello-interval 15
 address-family ipv4 unicast
  ## Inherited from group isis
  metric 50
 !
!
interface TenGigE0/2/0/0.3524
 ## Inherited from group isis
 lsp-interval 40
 ## Inherited from group isis
 hello-interval 15
 address-family ipv4 unicast
  ## Inherited from group isis
  metric 50
 !
!
interface TenGigE0/2/0/0.3525
 ## Inherited from group isis
 lsp-interval 40
 ## Inherited from group isis
 hello-interval 15
 address-family ipv4 unicast
  ## Inherited from group isis
  metric 50
 !
!
interface TenGigE0/2/0/0.3526
 ## Inherited from group isis
 lsp-interval 40
 ## Inherited from group isis
 hello-interval 15
 ## Inherited from group isis
 address-family ipv4 unicast
  ## Inherited from group isis
  metric 50
```

```
 !
!
interface TenGigE0/2/0/0.3527
 ## Inherited from group isis
 lsp-interval 40
 ## Inherited from group isis
 hello-interval 15
 ## Inherited from group isis
 address-family ipv4 unicast
  ## Inherited from group isis
  metric 50
 !
!
interface TenGigE0/2/0/0.3528
 ## Inherited from group isis
 lsp-interval 40
 ## Inherited from group isis
 hello-interval 15
 ## Inherited from group isis
 address-family ipv4 unicast
  ## Inherited from group isis
  metric 50
 !
!
interface TenGigE0/2/0/1
 ## Inherited from group isis
 lsp-interval 40
 ## Inherited from group isis
 hello-interval 15
 address-family ipv4 unicast
  ## Inherited from group isis
  metric 50
 !
!
!
```

4. Verify the actual functionality:

```
RP/0/RP0/CPU0:router# show isis interface TenGigE0/2/0/0.3528 | inc Metric

Metric (L1/L2):         50/50
```

OSPF Hierarchy: Example

This example illustrates hierarchical inheritance and priority. The configuration that is lower in hierarchy gets the highest priority.

1. Configure a local OSPF configuration:

```
RP/0/RP0/CPU0:router# show running router ospf

router ospf 1
 apply-group go-c
 nsr
 router-id 121.121.121.121
 nsf cisco
 redistribute connected
 address-family ipv4 unicast
```

```
 area 0
  apply-group go-b
  interface GigabitEthernet0/0/0/0
   apply-group go-a
  !
  interface GigabitEthernet0/0/0/1
  !
  interface GigabitEthernet0/0/0/3
  !
  interface GigabitEthernet0/0/0/4
  !
  interface GigabitEthernet0/0/0/21
   bfd minimum-interval 100
   bfd fast-detect
   bfd multiplier 3
  !
  interface TenGigE0/2/0/0.3891
  !
  interface TenGigE0/2/0/0.3892
  !
  interface TenGigE0/2/0/0.3893
  !
  interface TenGigE0/2/0/0.3894
  !
 !
!
router ospf 100
!
router ospf 1000
!
router ospf 1001
!
```

2. Configure a configuration group and apply it in a configuration submode:

```
RP/0/RP0/CPU0:router# show running group go-a

group go-a
 router ospf '.*'
  area '.*'
   interface 'Gig.*'
    cost 200
   !
  !
 !
end-group

RP/0/RP0/CPU0:router# show running group go-b

group go-b
 router ospf '.*'
  area '.*'
   interface 'Gig.*'
    cost 250
   !
  !
 !
end-group

RP/0/RP0/CPU0:router# show running group go-c

group go-c
 router ospf '.*'
```

```
  area '.*'
   interface 'Gig.*'
    cost 300
   !
  !
 !

end-group
```

3. Check the inheritance view and verify that the apply-group in the lowest configuration submode gets the highest priority:

```
RP/0/RP0/CPU0:router# show running router ospf 1 inheritance detail

router ospf 1
 nsr
 router-id 121.121.121.121
 nsf cisco
 redistribute connected
 address-family ipv4 unicast
 area 0
  interface GigabitEthernet0/0/0/0
   ## Inherited from group go-a
   cost 200                    << apply-group in lowest submode  gets highest priority

  !
  interface GigabitEthernet0/0/0/1
   ## Inherited from group go-b
   cost 250
  !
  interface GigabitEthernet0/0/0/3
   ## Inherited from group go-b
   cost 250
  !
  interface GigabitEthernet0/0/0/4
   ## Inherited from group go-b
   cost 250
  !
  interface GigabitEthernet0/0/0/21
   bfd minimum-interval 100
   bfd fast-detect
   bfd multiplier 3
   ## Inherited from group go-b
   cost 250
  !
  interface TenGigE0/2/0/0.3891
  !
  interface TenGigE0/2/0/0.3892
  !
  interface TenGigE0/2/0/0.3893
  !
  interface TenGigE0/2/0/0.3894
  !
 !
!
```

4. Check the functionality of the cost inheritance through the groups:

```
RP/0/RP0/CPU0:router# show ospf 1 interface GigabitEthernet 0/0/0/0

GigabitEthernet0/0/0/0 is up, line protocol is up
  Internet Address 1.0.1.1/30, Area 0
  Process ID 1, Router ID 121.121.121.121, Network Type BROADCAST, Cost: 200
```

```
Transmit Delay is 1 sec, State DR, Priority 1, MTU 1500, MaxPktSz 1500
Designated Router (ID) 121.121.121.121, Interface address 1.0.1.1
No backup designated router on this network
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Non-Stop Forwarding (NSF) enabled
  Hello due in 00:00:02
Index 5/5, flood queue length 0
Next 0(0)/0(0)
Last flood scan length is 1, maximum is 40
Last flood scan time is 0 msec, maximum is 7 msec
LS Ack List: current length 0, high water mark 0
Neighbor Count is 1, Adjacent neighbor count is 0
Suppress hello for 0 neighbor(s)
Multi-area interface Count is 0
```

Link Bundling Usage: Example

This example shows how to configure interface membership in a bundle link:

1. Configure the configuration groups:

```
RP/0/RP0/CPU0:router# show running group bundle1

group bundle1
 interface 'GigabitEthernet0/1/0/1[1-6]'
  bundle id 1 mode active
 !
end-group

RP/0/RP0/CPU0:router# show running | inc apply-group

Building configuration...

apply-group isis l2tr  isis2  mpp  bundle1
```

2. Check the local configuration:

```
RP/0/RP0/CPU0:router# show running interface gigabitEthernet 0/1/0/11

interface GigabitEthernet0/1/0/11
!

RP/0/RP0/CPU0:router# show running interface Bundle-Ether1

interface Bundle-Ether1
 ipv4 address 108.108.1.1 255.255.255.0
 bundle maximum-active links 10
 bundle minimum-active links 5
!
```

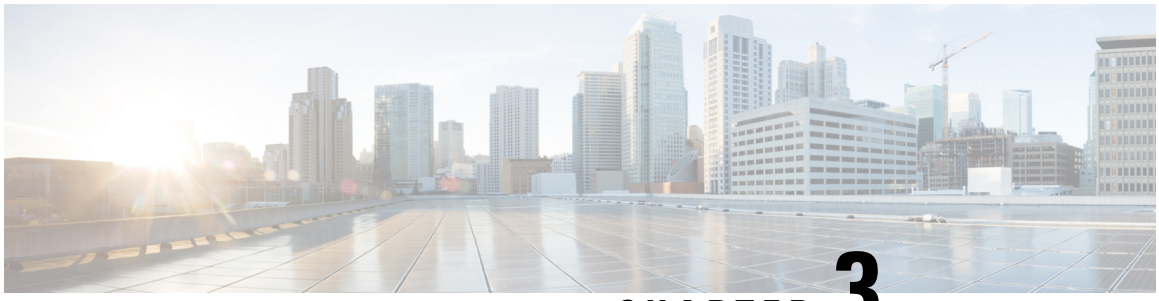
3. Check the inheritance configuration view:

```
RP/0/RP0/CPU0:router# show running interface GigabitEthernet 0/1/0/11  inheritance detail

interface GigabitEthernet0/1/0/11
```

```
 ## Inherited from group bundle1
 bundle id 1 mode active
!
```

4. Check that the inheritance configuration took effect:

```
RP/0/RP0/CPU0:router# show interface Bundle-Ether1

Bundle-Ether1 is up, line protocol is up
  Interface state transitions: 1
  Hardware is Aggregated Ethernet interface(s), address is 0024.f71f.4bc3
  Internet address is 108.108.1.1/24
  MTU 1514 bytes, BW 6000000 Kbit (Max: 6000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 6000Mb/s
  loopback not set,
  ARP type ARPA, ARP timeout 04:00:00
    No. of members in this bundle: 6
      GigabitEthernet0/1/0/11      Full-duplex  1000Mb/s     Active
      GigabitEthernet0/1/0/12      Full-duplex  1000Mb/s     Active
      GigabitEthernet0/1/0/13      Full-duplex  1000Mb/s     Active
      GigabitEthernet0/1/0/14      Full-duplex  1000Mb/s     Active
      GigabitEthernet0/1/0/15      Full-duplex  1000Mb/s     Active
      GigabitEthernet0/1/0/16      Full-duplex  1000Mb/s     Active
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters never
  5 minute input rate 8000 bits/sec, 1 packets/sec
  5 minute output rate 3000 bits/sec, 1 packets/sec
     2058 packets input, 1999803 bytes, 426 total input drops
     0 drops for unrecognized upper-level protocol
     Received 1 broadcast packets, 2057 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     1204 packets output, 717972 bytes, 0 total output drops
     Output 2 broadcast packets, 1202 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```



CHAPTER 3

Configuring Network Time Protocol

Network Time Protocol (NTP) is a protocol designed to time-synchronize devices within a network. Cisco IOS XR software implements NTPv4. NTPv4 retains backwards compatibility with the older versions of NTP, including NTPv3 and NTPv2 but excluding NTPv1, which has been discontinued due to security vulnerabilities.

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.

The Cisco implementation of NTP supports three ways that a networking device can obtain NTP time information on a network:

- By polling host servers
- By listening to NTP broadcasts
- By listening to NTP multicasts
- By polling host servers
- By listening to NTP broadcasts

In a LAN environment, NTP can be configured to use IP broadcast or multicast messages. As compared to polling, IP broadcast or multicast 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.

An NTP multicast server periodically sends a message to a designated IPv4 or IPv6 local multicast group address. An NTP multicast client listens on this address for NTP messages.

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.

NTP-PTP Interworking

NTP-PTP interworking provides the ability to use PTP, as well as other valid time of day (TOD) sources such as Data over Cable Service Interface Specification (DOCSIS) Timing Interface (DTI) and global positioning system (GPS), as the time source for the operating system. Prior to the support of NTP-PTP interworking, only backplane time was supported for the operating system time.

NTP-PTP interworking also provides the means to communicate status changes between PTP and NTP processes. It also supports the unambiguous control of the operating system time and backplane time in the event of bootup, switchovers or card and process failures.

For information regarding configuring NTP-PTP interworking, refer to *System Management Configuration Guide for Cisco NCS 5500 Series Routers*. For information regarding commands, refer to .

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.

SUMMARY STEPS

1. **configure**
2. **ntp**
3. **server** *ip-address* [**version** *number*] [**key** *key-id*] [**minpoll** *interval*] [**maxpoll** *interval*] [**source** *type interface-path-id*] [**prefer**] [**burst**] [**iburst**]
4. **peer** *ip-address* [**version** *number*] [**key** *key-id*] [**minpoll** *interval*] [**maxpoll** *interval*] [**source** *type interface-path-id*] [**prefer**]
5. 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	**ntp** **Example:** `RP/0/RP0/CPU0:router(config)# ntp`	Enters NTP configuration mode.

	Command or Action	Purpose

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 tengige 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.

SUMMARY STEPS

1. **configure**
2. **ntp**
3. (Optional) **broadcastdelay** *microseconds*
4. **interface** *type interface-path-id*
5. **broadcast client**
6. **broadcast** [**destination** *ip-address*] [**key** *key-id*] [**version** *number*]
7. 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	**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.

	Command or Action	Purpose
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 the 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. NTP communication consists of time requests and control queries. A *time request* is a request for time synchronization from an NTP server. A *control query* is a request for configuration information from an NTP server.

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).

SUMMARY STEPS

1. **configure**
2. **ntp**
3. **access-group**{**peer** | **query-only** | **serve** | **serve-only**} *access-list-name*
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	**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:	Saves configuration changes.

	Command or Action	Purpose

	• **end**
	• **commit**
	Example:
	`RP/0/RP0/CPU0:router(config-ntp)# end`
	or
	`RP/0/RP0/CPU0:router(config-ntp)# commit`
		• 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.

Note No specific command enables NTP; the first NTP configuration command that you issue enables NTP.

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.

SUMMARY STEPS

1. **configure**

2. **ntp**
3. **authenticate**
4. **authentication-key** *key-number* **md5** [**clear** | **encrypted**] *key-name*
5. **trusted-key** *key-number*
6. 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	**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	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
	`RP/0/RP0/CPU0:router(config-ntp)# 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.

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.

SUMMARY STEPS

1. **configure**
2. **ntp**
3. Use one of the following commands:
   - **no interface** *type interface-path-id*
   - **interface** *type interface-path-id* **disable**
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	**ntp** **Example:**	Enters NTP configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config)# ntp`	

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.

Note No specific command enables NTP; the first NTP configuration command that you issue enables NTP.

SUMMARY STEPS

1. **configure**
2. **ntp**
3. **source** *type interface-path-id*
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	**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 37.

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.

	Command or Action	Purpose
		• 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.

Note No specific command enables NTP; the first NTP configuration command that you issue enables NTP.

SUMMARY STEPS

1. **configure**
2. **ntp**
3. **master** *stratum*
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	**ntp** **Example:** `RP/0/RP0/CPU0:router(config)# ntp`	Enters NTP configuration mode.
Step 3	**master** *stratum* **Example:**	`Makes the router an authoritative NTP server.`

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-ntp)# master 9`	**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.

Configuring NTP-PTP Interworking

Use this task to configure NTP to use PTP as the time source.

Before you begin

PTP must be supported and enabled on the router before NTP-PTP interworking can be configured. If PTP is not enabled, you receive an error message similar to the following when you try to commit the configuration:

```
RP/0/RP0/CPU0:router(config)# ntp master primary-reference-clock
RP/0/RP0/CPU0:router(config)# commit

% Failed to commit one or more configuration items. Please issue
'show configuration failed' from this session to view the errors

RP/0/RP0/CPU0:router(config)# show configuration failed
[:::]
ntp
```

```
 master primary-reference-clock
!!% 'ip-ntp' detected the 'fatal' condition 'PTP is not supported on this platform'
!
end
```

SUMMARY STEPS

1. **configure**
2. **ntp**
3. **master primary-reference-clock**
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	**ntp** **Example:** `RP/0/RP0/CPU0:router(config)# ntp`	Enters NTP configuration mode.
Step 3	**master primary-reference-clock** **Example:** `RP/0/RP0/CPU0:router(config-ntp)# master primary-reference-clock`	Specifies PTP to be the NTP time source.

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.

	Command or Action	Purpose
		• 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.

Note No specific command enables NTP; the first NTP configuration command that you issue enables NTP.

SUMMARY STEPS

1. **configure**
2. **ntp**
3. **update-calendar**
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	**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.

	Command or Action	Purpose

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.

SUMMARY STEPS

1. **show ntp associations** [**detail**] [**location** *node-id*]
2. **show ntp status** [**location** *node-id*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show ntp associations** [**detail**] [**location** *node-id*] **Example:** `RP/0/RP0/CPU0:router# show ntp associations`	Displays the status of NTP associations.

	Command or Action	Purpose
Step 2	**show ntp status** [**location** *node-id*] **Example:** `RP/0/RP0/CPU0:router# show ntp status`	Displays the status of NTP.

Examples

The following is sample output from the **show ntp associations** command:

```
RP/0/RP0/CPU0:router# show ntp associations

Tue Oct  7 11:22:46.839 JST

      address         ref clock     st  when  poll reach  delay  offset    disp
*~192.168.128.5    10.81.254.131     2     1    64  377    7.98  -0.560   0.108
+~dead:beef::2 vrf testAA
                   171.68.10.80      3    20    64  377    6.00  -2.832   0.046
* sys_peer, # selected, + candidate, - outlayer, x falseticker, ~ configured


RP/0/RP0/CPU0:router# show ntp associations

      address         ref clock     st  when  poll reach  delay  offset    disp
+~127.127.1.1      127.127.1.1       5     5  1024   37     0.0    0.00   438.3
*~172.19.69.1      172.24.114.33     3    13  1024    1     2.0   67.16     0.0
 * master (synced), # master (unsynced), + selected, - candidate, ~ configured
```

The following is sample output from the **show ntp status** command:

```
RP/0/RP0/CPU0:router# show ntp status

Tue Oct  7 11:22:54.023 JST

Clock is synchronized, stratum 3, reference is 192.168.128.5
nominal freq is 1000.0000 Hz, actual freq is 1000.2725 Hz, precision is 2**24
reference time is CC95463C.9B964367 (11:21:48.607 JST Tue Oct  7 2008)
clock offset is -1.738 msec, root delay is 186.050 msec
root dispersion is 53.86 msec, peer dispersion is 0.09 msec
loopfilter state is 'CTRL' (Normal Controlled Loop), drift is -0.0002724105 s/s
system poll interval is 64, last update was 66 sec ago


RP/0/RP0/CPU0:router# show ntp status

Clock is synchronized, stratum 4, reference is 172.19.69.1
nominal freq is 1000.0000 Hz, actual freq is 999.9988 Hz, precision is 2**26
reference time is C54C131B.9EECF6CA (07:26:19.620 UTC Mon Nov 24 2008)
clock offset is 66.3685 msec, root delay is 7.80 msec
root dispersion is 950.04 msec, peer dispersion is 3.38 msec
```

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 tengige 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 tengige 0/2/0/2
    broadcast
```

Configuring Multicast-Based Associations: Example

The following example shows an NTP multicast client configuration where 10-Gigabit Ethernet interface 0/1/1/0 is configured to be a multicast client and to join the default multicast group (IPv4 address 224.0.1.1):

```
ntp interface TenGigE 0/1/1/0
  multicast client
```

The following example shows an NTP multicast server configuration where 10-Gigabit Ethernet interface 0/1/1/0 is configured to be a multicast server:

```
ntp interface TenGigE 0/1/1/0
```

```
  multicast destination 224.0.1.1
```

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 tengige 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
```

Configuring NTP server inside VRF interface

This task explains how to configure NTP server inside VRF interface.

Note No specific command enables NTP; the first NTP configuration command that you issue enables NTP.

SUMMARY STEPS

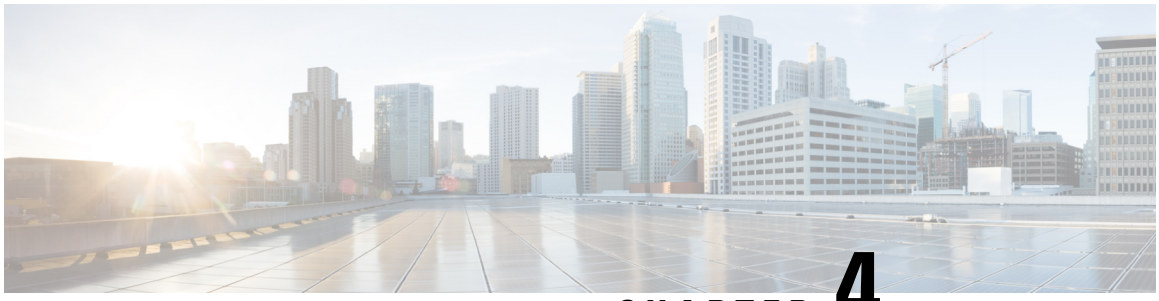
1. **configure**
2. **ntp**
3. **vrf** *vrf-name*
4. **source** *interface-type interface-instance*
5. Use one of the following commands:
   - **end**
   - **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**ntp** **Example:** `RP/0/RP0/CPU0:router(config)# ntp`	Enters NTP configuration mode.

	Command or Action	Purpose
Step 3	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config)# ntp vrf Customer_A`	Specify name of a VRF (VPN- routing and forwarding) instance to configure.
Step 4	**source** *interface-type interface-instance* **Example:** `RP/0/RP0/CPU0:router(config)# ntp vrf Customer_A source bvi 70`	Configures an interface from which the IP source address is taken. This allows IOS-XR to respond to NTP queries on VRF interfaces, in this case the source is BVI. **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 37.

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.



CHAPTER 4

Configuring Manageability

This module describes the configuration required to enable the Extensible Markup Language (XML) agent services. The XML Parser Infrastructure provides parsing and generation of XML documents with Document Object Model (DOM), Simple Application Programming Interface (API) for XML (SAX), and Document Type Definition (DTD) validation capabilities:

- DOM allows customers to programmatically create, manipulate, and generate XML documents.
- SAX supports user-defined functions for XML tags.
- DTD allows for validation of defined document types.

Information about XML Manageability

The Cisco IOS XR Extensible Markup Language (XML) API provides a programmable interface to the router for use by external management applications. This interface provides a mechanism for router configuration and monitoring utilizing XML formatted request and response streams. The XML interface is built on top of the Management Data API (MDA), which provides a mechanism for Cisco IOS XR components to publish their data models through MDA schema definition files.

Cisco IOS XR software provides the ability to access the router via XML using a dedicated TCP connection, Secure Socket Layer (SSL), or a specific VPN routing and forwarding (VRF) instance.

How to Configure Manageability

Configuring the XML Agent

This explains how to configure the XML agent.

SUMMARY STEPS

1. **xml agent** [**ssl**]
2. **iteration on size** *iteration-size*
3. **session timeout** *timeout*

4. **throttle** {**memory** *size* | **process-rate** *tags*}
5. **vrf** { **vrfname** | **ipv4**} [**access-list** *access-list-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**xml agent** [**ssl**] **Example:** `RP/0/RP0/CPU0:router(config)# xml agent ssl`	Enables Extensible Markup Language (XML) requests over a dedicated TCP connection and enters XML agent configuration mode. Use the **ssl** keyword to enable XML requests over Secure Socket Layer (SSL).
Step 2	**iteration on size** *iteration-size* **Example:** `RP/0/RP0/CPU0:router(config-xml-agent)# iteration on size 500`	Configures the iteration size for large XML agent responses in KBytes. The default is 48.
Step 3	**session timeout** *timeout* **Example:** `RP/0/RP0/CPU0:router(config-xml-agent)# session timeout 5`	Configures an idle timeout for the XML agent in minutes. By default, there is no timeout.
Step 4	**throttle** {**memory** *size*	**process-rate** *tags*} **Example:** `RP/0/RP0/CPU0:router(config-xml-agent)# throttle memory 300`	Configures the XML agent processing capabilities. • Specify the memory size in Mbytes. Values can range from 100 to 600. The default is 300. • Specify the process-rate as the number of tags that the XML agent can process per second. Values can range from 1000 to 30000. By default the process rate is not throttled.
Step 5	**vrf** { **vrfname**	**ipv4**} [**access-list** *access-list-name*] **Example:** `RP/0/RP0/CPU0:router(config-xml-agent)# vrf vrf1`	Configures the dedicated agent or SSL agent to receive and send messages via the specified VPN routing and forwarding (VRF) instance.

Configuration Examples for Manageability

Enabling VRF on an XML Agent: Examples

The following example illustrates how to configure the dedicated XML agent to receive and send messages via VRF1, VRF2 and the default VRF:

```
RP/0/RP0/CPU0:router(config)# xml agent
RP/0/RP0/CPU0:router(config-xml-agent)# vrf VRF1
RP/0/RP0/CPU0:router(config-xml-agent)# vrf VRF2
```

The following example illustrates how to remove access to VRF2 from the dedicated agent:

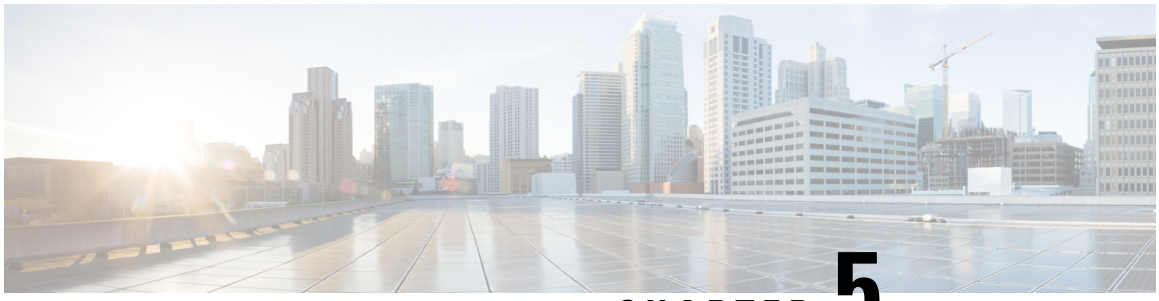
```
RP/0/RP0/CPU0:router(config)# xml agent ssl
RP/0/RP0/CPU0:router(config-xml-ssl)# vrf VRF1
RP/0/RP0/CPU0:router(config-xml-ssl-vrf)# vrf VRF2

RP/0/RP0/CPU0:router(config)# xml agent
RP/0/RP0/CPU0:router(config-xml-agent)# no vrf VRF1
```

The following example shows how to configure the XML SSL agent to receive and send messages through VRF1, VRF2 and the default VRF:

```
RP/0/RP0/CPU0:router(config)# xml agent ssl
RP/0/RP0/CPU0:router(config-xml-agent)# vrf VRF1
RP/0/RP0/CPU0:router(config-xml-agent)# vrf VRF2
```

The following example removes access for VRF2 from the dedicated XML agent:

```
RP/0/RP0/CPU0:router(config)# xml agent ssl
RP/0/RP0/CPU0:router(config-xml-agent)# no vrf VRF2
```



CHAPTER 5

Configuring Object Tracking

This module describes the configuration of object tracking on your Cisco IOS XR network. For complete descriptions of the commands listed in this module, see **Additional References** section. To locate documentation for other commands that might appear in the course of performing a configuration task, see **Technical Documentation** section in the Additional References topic.

Configuring Object Tracking

This module describes the configuration of object tracking on your Cisco IOS XR network. For complete descriptions of the commands listed in this module, see **Additional References** section. To locate documentation for other commands that might appear in the course of performing a configuration task, see **Technical Documentation** section in the Additional References topic.

Prerequisites for Implementing Object Tracking

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.

Note Object Tracking is an optional package. You must check if this package is installed on your system by running the command **show install active summary**.

Information about Object Tracking

Object tracking is a mechanism to track an object and to take an action on another object with no relationship to the tracked objects, based on changes to the properties of the object being tracked.

Each tracked object is identified by a unique name specified on the tracking command-line interface (CLI). Cisco IOS XR processes then use this name to track a specific object.

The tracking process periodically polls the tracked object and reports any changes to its state in terms of its being up or down, either immediately or after a delay, as configured by the user.

Multiple objects can also be tracked by means of a list, using a flexible method for combining objects with Boolean logic. This functionality includes:

- **Boolean AND function—**When a tracked list has been assigned a Boolean AND function, each object defined within a subset must be in an up state, so that the tracked object can also be in the up state.
- **Boolean OR function—**When the tracked list has been assigned a Boolean OR function, it means that at least one object defined within a subset must also be in an up state, so that the tracked object can also be in the up state.

How to Implement Object Tracking

This section describes the various object tracking procedures.

Tracking the Line Protocol State of an Interface

Perform this task in global configuration mode to track the line protocol state of an interface.

A tracked object is considered up when a line protocol of the interface is up.

After configuring the tracked object, you may associate the interface whose state should be tracked and specify the number of seconds to wait before the tracking object polls the interface for its state.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type line-protocol state**
4. **interface** *type interface-path-id*
5. **exit**
6. (Optional) **delay** {**up** *seconds* | **down** *seconds*}
7. 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	**track** *track-name* **Example:** `RP/0/RP0/CPU0:router(config)# track track1`	Enters track configuration mode. • *track-name*—Specifies a name for the object to be tracked.
Step 3	**type line-protocol state** **Example:** `RP/0/RP0/CPU0:router(config-track)# type line-protocol state`	Creates a track based on the line protocol of an interface.
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-track-line-prot)# interface atm 0/2/0/0.1`	Specifies the interface to track the protocol state. • *type*—Specifies the interface type. For more information, use the question mark (?) online help function. • *interface-path-id*—Identifies a physical interface or a virtual interface. **Note** Use the **show interfaces** command to see a list of all possible interfaces currently configured on the router. **Note** The loopback and null interfaces are always in the up state and, therefore, cannot be tracked.
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-track-line-prot)# exit`	Exits the track line protocol configuration mode.
Step 6	(Optional) **delay** {**up** *seconds* | **down** *seconds*} **Example:** `RP/0/RP0/CPU0:router(config-track)# delay up 10`	Schedules the delay that can occur between tracking whether the object is up or down.
Step 7	Use one of the following commands: • **end** • **commit** **Example:** `RP/0/RP0/CPU0:router(config-track)# 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-track)# 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.

Tracking IP Route Reachability

When a host or a network goes down on a remote site, routing protocols notify the router and the routing table is updated accordingly. The routing process is configured to notify the tracking process when the route state changes due to a routing update.

A tracked object is considered up when a routing table entry exists for the route and the route is accessible.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type route reachability**
4. Use one of the following commands:
   - **vrf** *vrf-table-name*
   - **route ipv4** *IP-prefix/mask*
5. **exit**
6. (Optional) **delay** {**up** *seconds* | **down** *seconds*}
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	**track** *track-name* **Example:**	Enters track configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config)# track track1`	• *track-name*—Specifies a name for the object to be tracked.
Step 3	**type route reachability** **Example:** `RP/0/RP0/CPU0:router(config-track)# type route reachability vrf internet`	Configures the routing process to notify the tracking process when the state of the route changes due to a routing update.

Step 4	Use one of the following commands:
	• **vrf** *vrf-table-name*
	• **route ipv4** *IP-prefix/mask*
	Example:
	`RP/0/RP0/CPU0:router(config-track-route)# vrf vrf-table-4`
	or
	`RP/0/RP0/CPU0:router(config-track-route)# route ipv4 10.56.8.10/16`
		Configures the type of IP route to be tracked, which can consist of either of the following, depending on your router type:
		• *vrf-table-name*—A VRF table name.
		• *IP-prefix/mask*—An IP prefix consisting of the network and subnet mask (for example, 10.56.8.10/16).

Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-track-line-prot)# exit`	Exits the track line protocol configuration mode.
Step 6	(Optional) **delay** {**up** *seconds* | **down** *seconds*} **Example:** `RP/0/RP0/CPU0:router(config-track)# delay up 10`	Schedules the delay that can occur between tracking whether the object is up or down.
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 mode, without committing the configuration changes.

Building a Track Based on a List of Objects

Perform this task in the global configuration mode to create a tracked list of objects (which, in this case, are lists of interfaces or prefixes) using a Boolean expression to determine the state of the list.

A tracked list contains one or more objects. The Boolean expression enables two types of calculations by using either AND or OR operators. For example, when tracking two interfaces, using the AND operator, up means that *both* interfaces are up, and down means that *either* interface is down.

Note An object must exist before it can be added to a tracked list.

The NOT operator is specified for one or more objects and negates the state of the object.

After configuring the tracked object, you must associate the interface whose state should be tracked and you may optionally specify the number of seconds to wait before the tracking object polls the interface for its state.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type list boolean** { **and** | **or** }
4. **object** *object-name* [**not**]
5. **exit**
6. (Optional) **delay** {**up** *seconds* | **down** *seconds*}
7. 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	**track** *track-name* **Example:** `RP/0/RP0/CPU0:router(config)# track track1`	Enters track configuration mode. • *track-name*—Specifies a name for the object to be tracked.
Step 3	**type list boolean** { **and**	**or** } **Example:** `RP/0/RP0/CPU0:router(config-track)# type list boolean and`	Configures a Boolean list object and enters track list configuration mode. • **boolean**—Specifies that the state of the tracked list is based on a Boolean calculation. • **and**—Specifies that the list is up if all objects are up, or down if one or more objects are down. For example when tracking two interfaces, up means that both interfaces are up, and down means that either interface is down.

	Command or Action	Purpose
		• **or**—Specifies that the list is up if at least one object is up. For example, when tracking two interfaces, up means that either interface is up, and down means that both interfaces are down.
Step 4	**object** *object-name* [**not**] **Example:** `RP/0/RP0/CPU0:router(config-track-list)# object 3 not`	Specifies the object to be tracked by the list • *obect-name*—Name of the object to track. • **not**—Negates the state of the object.
Step 5	**exit** **Example:** `RP/0/RP0/CPU0:router(config-track-line-prot)# exit`	Exits the track line protocol configuration mode.
Step 6	(Optional) **delay** {**up** *seconds* | **down** *seconds*} **Example:** `RP/0/RP0/CPU0:router(config-track)# delay up 10`	Schedules the delay that can occur between tracking whether the object is up or down.

Step 7	Use one of the following commands:
	• **end**
	• **commit**
	Example:
	`RP/0/RP0/CPU0:router(config-track)# end`
	or
	`RP/0/RP0/CPU0:router(config-track)# 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.

Building a Track Based on a List of Objects - Threshold Percentage

Perform this task in the global configuration mode to create a tracked list of objects (which, in this case, are lists of interfaces or prefixes) using a threshold percentage to determine the state of the list.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type list threshold percentage**
4. **object** *object-name*
5. **threshold percentage up** *percentage* **down** *percentage*
6. 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	**track** *track-name* **Example:** `RP/0/RP0/CPU0:router(config)# track track1`	Enters track configuration mode. • *track-name*—Specifies a name for the object to be tracked.
Step 3	**type list threshold percentage** **Example:** `RP/0/RP0/CPU0:router(config-track)# type list threshold percentage`	Configures a track of type threshold percentage list.

Step 4	**object** *object-name*
	Example:
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 1`
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 2`
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 3`
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 4`
		Configures object 1, object 2, object 3 and object 4 as members of track type track1.

Step 5	**threshold percentage up** *percentage* **down** *percentage* **Example:** `RP/0/RP0/CPU0:router(config-track-list-threshold)# threshold percentage up 50 down 33`	Configures the percentage of objects that need to be UP or DOWN for the list to be considered UP or Down respectively. For example, if object 1, object 2, and object 3 are in the UP state and object 4 is in the DOWN state, the list is considered to be in the UP state.

	Command or Action	Purpose

Step 6	Use one of the following commands:
	• **end**
	• **commit**
	Example:
	`RP/0/RP0/CPU0:router(config-track)# end`
	or
	`RP/0/RP0/CPU0:router(config-track)# 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.

Building a Track Based on a List of Objects - Threshold Weight

Perform this task in the global configuration mode to create a tracked list of objects (which, in this case, are lists of interfaces or prefixes) using a threshold weight to determine the state of the list.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type list threshold weight**
4. **object** *object-name* **weight** *weight*
5. **threshold weight up** *weight* **down** *weight*
6. Use one of the following commands:
   - **end**
   - **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# configure`	
Step 2	**track** *track-name* **Example:** `RP/0/RP0/CPU0:router(config)# track track1`	Enters track configuration mode. • *track-name*—Specifies a name for the object to be tracked.
Step 3	**type list threshold weight** **Example:** `RP/0/RP0/CPU0:router(config-track)# type list threshold weight`	Configures a a track of type, threshold weighted list.

Step 4	**object** *object-name* **weight** *weight*
	Example:
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 1 weight 10`
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 2 weight 5`
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 3 weight 3`
		Configures object 1, object 2 and object 3 as members of track t1 and with weights 10, 5 and 3 respectively.

Step 5	**threshold weight up** *weight* **down** *weight* **Example:** `RP/0/RP0/CPU0:router(config-track-list-threshold)# threshold weight up 10 down 5`	Configures the range of weights for the objects that need to be UP or DOWN for the list to be considered UP or DOWN respectively. In this example, the list is considered to be in the DOWN state because objects 1 and 2 are in the UP state and the cumulative weight is 15 (not in the 10-5 range).

Step 6	Use one of the following commands:
	• **end**
	• **commit**
	Example:
	`RP/0/RP0/CPU0:router(config-track)# end`
	or
	`RP/0/RP0/CPU0:router(config-track)# 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.

	Command or Action	Purpose
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Tracking IPSLA Reachability

Use this task to enable the tracking of the return code of IP service level agreement (SLA) operations.

SUMMARY STEPS

1. **configure**
2. **track** *track-name*
3. **type rtr** *ipsla-no* **reachability**
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:** `RP/0/RP0/CPU0:router(config)# track t1`	Enters track configuration mode.
Step 3	**type rtr** *ipsla-no* **reachability** **Example:** `RP/0/RP0/CPU0:router(config-track)# type rtr 100 reachability`	Specifies the IP SLA operation ID to be tracked for reachability. Values for the *ipsla-no* can range from1 to 2048.
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 IPSLA Tracking: Example

This example shows the configuration of IPSLA tracking:

```
RP/0/RP0/CPU0:router(config)# track track1
RP/0/RP0/CPU0:router(config-track)# type rtr 1 reachability
RP/0/RP0/CPU0:router(config-track)# delay up 5
RP/0/RP0/CPU0:router(config-track)# delay down 10
```

Configuration Examples for Configuring Object Tracking

Tracking Whether the Interface Is Up or Down: Running Configuration Example

```
track connection100
  type list boolean and
    object object3 not
    delay up 10
    !
interface service-ipsec 23
  line-protocol track connection100
  !
```

Tracking the Line Protocol State of an Interface: Running Configuration Example

In this example, traffic arrives from interface service-ipsec1 and exits through interface gigabitethernet0/0/0/3:

```
track IPSec1
  type line-protocol state
    interface gigabitethernet0/0/0/3
    !
interface service-ipsec 1
  ipv4 address 70.0.0.1 255.255.255.0
  profile vrf1_profile_ipsec
  line-protocol track IPSec1
  tunnel source 80.0.0.1
  tunnel destination 80.0.0.2
  service-location preferred-active 0/0/1
  !
```

This example displays the output from the **show track** command after performing the previous example:

```
RP/0/RP0/CPU0:router# show run track

  Track IPSec1
  Interface GigabitEthernet0_0_0_3 line-protocol
  !
    Line protocol is UP
```

```
    1 change, last change 10:37:32 UTC Thu Sep 20 2007
    Tracked by:
    service-ipsec1
    !
```

Tracking IP Route Reachability: Running Configuration Example

In this example, traffic arriving from interface service-ipsec1 has its destination in network 7.0.0.0/24. This tracking procedure follows the state of the routing protocol prefix to signal when there are changes in the routing table.

```
track PREFIX1
  type route reachability
    route ipv4 7.0.0.0/24
    !
  interface service-ipsec 1
  vrf 1
  ipv4 address 70.0.0.2 255.255.255.0
  profile vrf_1_ipsec
  line-protocol track PREFIX1
  tunnel source 80.0.0.2
  tunnel destination 80.0.0.1
  service-location preferred-active 0/2/0
```

Building a Track Based on a List of Objects: Running Configuration Example

In this example, traffic arriving from interface service-ipsec1 exits through interface gigabitethernet0/0/0/3 and interface ATM 0/2/0/0.1. The destination of the traffic is at network 7.0.0.0/24.

If either one of the interfaces or the remote network goes down, the flow of traffic must stop. To do this, we use a Boolean AND expression.

```
track C1
 type route reachability
  route ipv4 3.3.3.3/32
 !
!
track C2
 type route reachability
  route ipv4 1.2.3.4/32
 !
!
track C3
 type route reachability
  route ipv4 10.0.20.2/32
 !
!
track C4
 type route reachability
  route ipv4 10.0.20.0/24
 !
!
track OBJ
 type list boolean and
  object C1
  object C2
```

```
 !
!
track OBJ2
 type list boolean or
  object C1
  object C2
 !
```

Additional References

The following sections provide references related to implementing object tracking for IPSec network security.

Related Documents

Related Topic	Document Title
IP SLA configuration information	*Implementing IP Service Level Agreements on* module in *System Monitoring Configuration Guide for Cisco NCS 5500 Series Routers*
IP SLA commands	*IP Service Level Agreement Commands on* module in *System Monitoring Command Reference for Cisco NCS 5500 Series Routers and Cisco NCS 540 and NCS 560 Series Routers*
Object tracking commands	*Object Tracking Commands on* module in

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

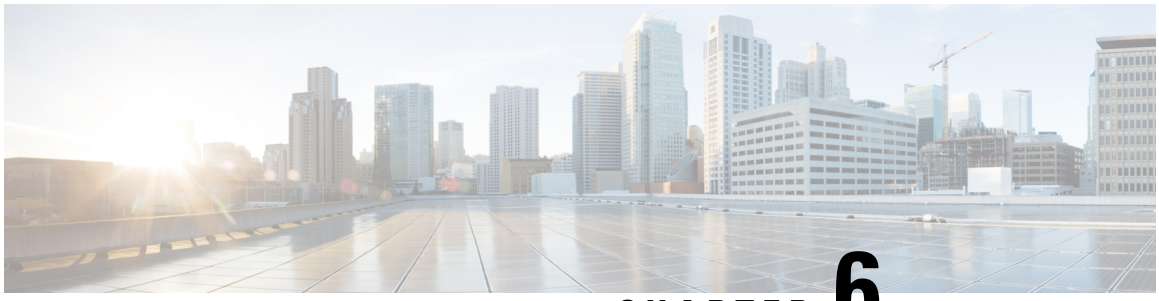
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 2401	*Security Architecture for the Internet Protocol*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 6

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.

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.

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.

Note Before creating or modifying the vty pools, enable the telnet server using the **telnet server** command in XR Config mode. See Cisco IOS XR IP Addresses and Services Configuration Guide and Cisco IOS XR IP Addresses and Services Command Reference 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`
	or
	`RP/0/RP0/CPU0:router(config-line)# 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.

Creating and Modifying vty Pools

This task explains how to create and modify vty pools.

You can omit Step 3 to Step 5 (**line template** and **exit** commands) if you are configuring the default line template to reference a vty pool.

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.
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*}]	Creates or modifies vty pools.

	Command or Action	Purpose

	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`
	
	`RP/0/RP0/CPU0:router(config)#vty-pool default 0 5 line-template template1`
		• 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).
		• **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.
		  • 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:

```
RP/0/RP0/CPU0:router# show running-config line console
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:router# show line console location 0/0/CPU0
Tue Nov 24 03:10:24.656 UTC
Tty            Speed     Overruns           Acc I/O
*con0/0/CPU0      9600       0/0               -/-

Line "con0_RP1_CPU0", Location "0/RP1/CPU0", Type "Console"
Length: 24 lines, Width: 80 columns
Baud rate (TX/RX) is 9600, "No" Parity, 2 stopbits, 8 databits
Template: console
Capabilities: Timestamp Enabled
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

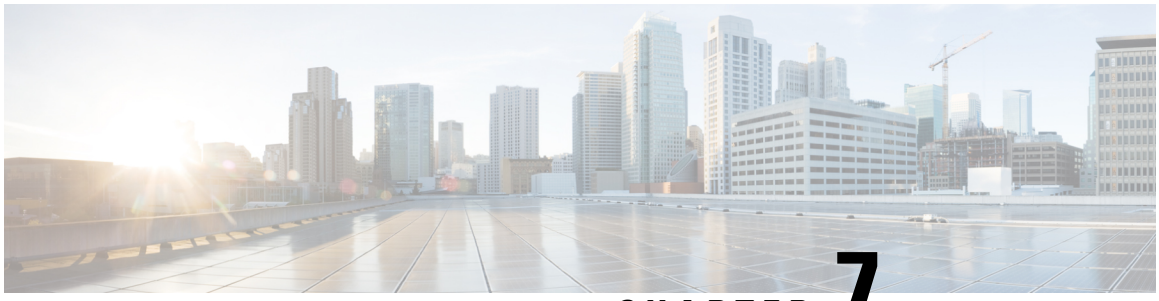
This configuration example shows how to configure a user-defined line template (named test2 in this example) for vtys and to configure the line template test to reference a user-defined vty pool (named pool1 in this example):

```
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 100 105 line-template test3
```



CHAPTER 7

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 A 10 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.

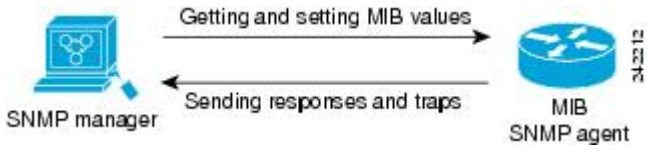
MIB

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



Note A few exceptions while performing SNMP walk on the NC55-6X200-DWDM-S line card on the NCS 5500 Series Routers are as follows:

1. Though the below mentioned OIDs are valid, they are marked as inaccessible in the OTN MIB standard. Hence they will not be polled during MIB walk.
   - OtnNearEndCurIntervalType : .1.3.6.1.4.1.9.9.639.1.2.3.1.1
   - OtnNearEndCurrentMonType : .1.3.6.1.4.1.9.9.639.1.2.3.1.2
   - OtnFarEndCurIntervalType : .1.3.6.1.4.1.9.9.639.1.2.4.1.1
   - OtnFarEndCurrentMonType : .1.3.6.1.4.1.9.9.639.1.2.4.1.2
2. OtnStatus : .1.3.6.1.4.1.9.9.639.1.1.1.1.5 OID is implicitly enabled for the interfaces of NC55-6X200-DWDM-S line card. Hence a MIB walk corresponding to the OtnStatus is not supported.

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 92 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 2: SNMPv1, v2c, and v3 Feature Support

Feature	SNMP v1	SNMP v2c	SNMP v3
Get-Bulk Operation	No	Yes	Yes
Inform Operation	No	Yes (No on the Cisco IOS XR software)	Yes (No on the Cisco IOS XR software)
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 3: 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 (k9sec) 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 4: 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.

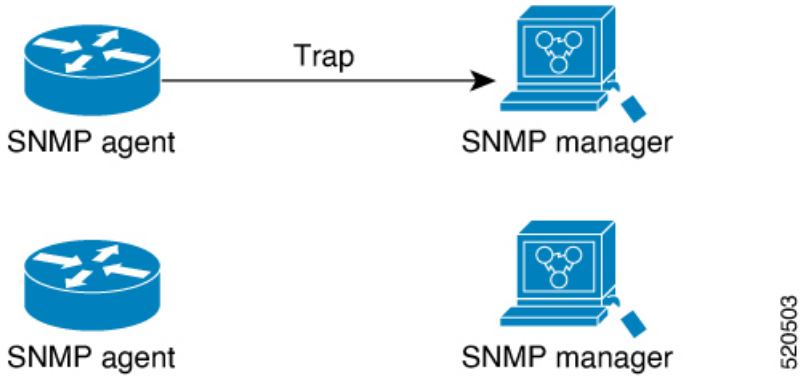
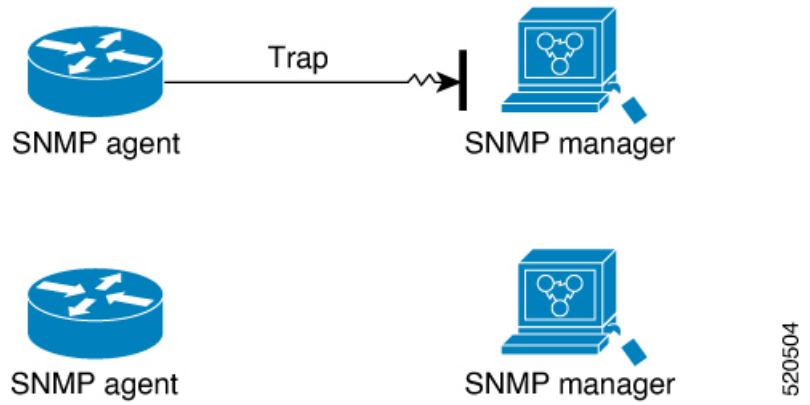


Figure 3: Trap Not Received by the SNMP Manager

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 NCS 5500 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.

SUMMARY STEPS

1. **configure**
2. (Optional) **snmp-server engineid local** *engine-id*
3. **snmp-server view** *view-name oid-tree* {**included** | **excluded**}
4. **snmp-server group** *name* {**v1** | **v2c** | **v3** {**auth** | **noauth** | **priv**}} [**read** *view*] [**write** *view*] [**notify** *view*] [*access-list-name*]
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*]
6. Use the **commit** or **end** command.
7. (Optional) **show snmp**
8. (Optional) **show snmp engineid**
9. (Optional) **show snmp group**
10. (Optional) **show snmp users**
11. (Optional) **show snmp view**

DETAILED STEPS

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

	Command or Action	Purpose
Step 2	(Optional) **snmp-server engineid local** *engine-id* **Example:** `RP/0/RP0/CPU0:router(config)# 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(config)# 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(config)# 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(config)# 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.

	Command or Action	Purpose
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.

SUMMARY STEPS

1. **configure**
2. **snmp-server drop unknown-user**
3. 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	**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.

	Command or Action	Purpose
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:

```
config
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:

```
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 Because 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 Because 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

This task explains how to configure the router to send SNMP trap notifications.

Note You can omit #unique_100 if you have already completed the steps documented under the #unique_100 task.

SUMMARY STEPS

1. **configure**
2. **snmp-server group** *name* {**v1 v2 v3** {**auth** | **noauth** | **priv**}} [**read** *view*] **write** *view*] [**notify** *view*] [*access-list-name*]
3. **snmp-server user** *username groupname* {**v1 v2c v3** {**auth** | **md5** | **sha**} {**clear** | **encrypted**} *auth-password*] [**priv des56** {**clear** | *access-list-name*]
4. [**snmp-server host** *address* [**traps**] [**version** {**1** | **2c** | **3** [**auth** | **priv**]}] *community-string* [**udp-port** *port*] [*notification-type*]
5. **snmp-server traps** [*notification-type*]
6. Use the **commit** or **end** command.
7. (Optional) **show snmp host**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 2	**snmp-server group** *name* {**v1 v2 v3** {**auth** | **noauth** | **priv**}} [**read** *view*] **write** *view*] [**notify** *view*] [*access-list-name*] **Example:** `RP/0/RP0/CPU0:router(config)# 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-server user** *username groupname* {**v1 v2c v3** {**auth** | **md5** | **sha**} {**clear** | **encrypted**} *auth-password*] [**priv des56** {**clear** | *access-list-name*] **Example:** `RP/0/RP0/CPU0:router(config)# 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.

	Command or Action	Purpose
Step 4	[**snmp-server host** *address* [**traps**] [**version** {**1** | **2c** | **3** [**auth** | **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 5	**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 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 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.

SUMMARY STEPS

1. **configure**
2. **snmp-server drop unknown-user**
3. 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	**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
!
```

In the following example, the output of the **show snmp host** commaand 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

```
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.

SUMMARY STEPS

1. **configure**
2. (Optional) **snmp-server contact** *system-contact-string*
3. (Optional) **snmp-server location** *system-location*
4. (Optional) **snmp-server chassis-id** *serial-number*
5. 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	(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.

SUMMARY STEPS

1. **configure**
2. (Optional) **snmp-server packetsize** *byte-count*
3. 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	(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.

SUMMARY STEPS

1. **configure**
2. (Optional) **snmp-server trap-source** *type interface-path-id*
3. (Optional) **snmp-server queue-length** *length*

4. (Optional) **snmp-server trap-timeout** *seconds*
5. 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	(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 IP Precedence or IP DSCP for SNMP traffic.

Before you begin

SNMP must be configured.

SUMMARY STEPS

1. **configure**
2. Use one of the following commands:
   - **snmp-server ipv4 precedence** *value*
   - **snmp-server ipv4 dscp** *value*
3. 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	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 IP precedence or IP 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 IP Precedence value to 7:

```
configure
  snmp-server ipv4 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 IP 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
```

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.

SUMMARY STEPS

1. **show snmp context-mapping**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**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.

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**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.

SUMMARY STEPS

1. (Optional) **snmp-server entityindex persist**
2. (Optional) **snmp-server mibs cbqosmib persist**

3. (Optional) **snmp-server cbqosmib cache refresh time** *time*
4. (Optional) **snmp-server cbqosmib cache service-policy count** *count*
5. **snmp-server ifindex persist**

DETAILED STEPS

	Command or Action	Purpose
Step 1	(Optional) **snmp-server entityindex persist** **Example:** `RP/0/RP0/CPU0:router(config)# snmp-server entityindex persist`	Enables the persistent storage of ENTITY-MIB data.
Step 2	(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 3	(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 4	(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 5	**snmp-server ifindex persist** **Example:** `RP/0/RP0/CPU0:router(config)# snmp-server ifindex persist`	Enables ifIndex 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.

SUMMARY STEPS

1. **configure**
2. **snmp-server interface subset** *subset-number* **regular-expression** *expression*
3. **notification linkupdown disable**
4. Use the **commit** or **end** command.
5. (Optional) **show snmp interface notification subset** *subset-number*
6. (Optional) **show snmp interface notification regular-expression** *expression*
7. (Optional) **show snmp interface notification** *type interface-path-id*

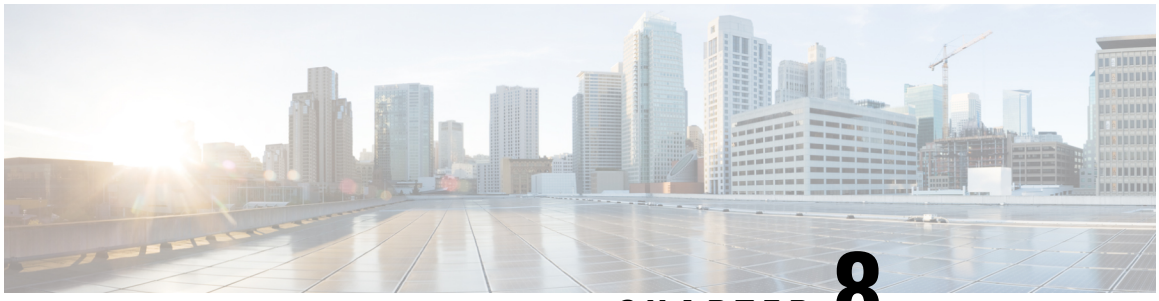
DETAILED STEPS

	Command or Action	Purpose
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.
		Refer to the *Understanding Regular Expressions, Special Characters, and Patterns* module in for more information regarding regular expressions.

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.

	Command or Action	Purpose
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 8

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.

SUMMARY STEPS

1. **configure**
2. **snmp-server mib bulkstat object-list** *list-name*
3. **add** {**oid** | *object-name*}
4. 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	**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:

	Command or Action	Purpose
		• **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.

SUMMARY STEPS

1. **configure**
2. **snmp-server mib bulkstat schema** *schema-name*
3. **object-list** *list-name*
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*
5. **poll-interval** *minutes*
6. 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	**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.

	Command or Action	Purpose
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.

SUMMARY STEPS

1. **configure**
2. **snmp-server mib bulkstat transfer-id** *transfer-id*
3. **buffer-size** *bytes*
4. **format** {**bulkBinary** | **bulkASCII** | **schemaASCII**}
5. **schema** *schema-name*
6. **transfer-interval** *minutes*
7. **url primary** *url*
8. **url secondary** *url*
9. **retry** *number*
10. **retain** *minutes*
11. **enable**
12. **commit** *minutes*

DETAILED STEPS

	Command or Action	Purpose
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.

	Command or Action	Purpose
		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	**format** {**bulkBinary** | **bulkASCII** | **schemaASCII**} **Example:** `RP/0/RP0/CPU0:router(config-bulk-tr)# format schemaASCII`	(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.

	Command or Action	Purpose
		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.

	Command or Action	Purpose
		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

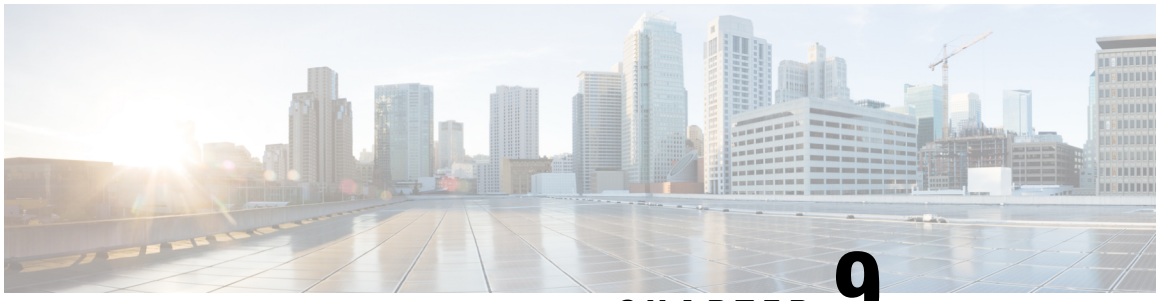
This example shows how to configure periodic MIB data collection and transfer:

```
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 9

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

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.

Note CDP is an optional package. You must check if this package is installed on your system by running the command **show install active summary**.

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 5: 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 task explains how to enable CDP globally on the router and then enable CDP on an interface.

SUMMARY STEPS

1. **configure**
2. **cdp**
3. **interface** *type interface-path-id*
4. **cdp**
5. 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	**cdp** **Example:** `RP/0/RP0/CPU0:router(config)# cdp`	Enables CDP globally.
Step 3	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# int TenGigE 0/5/0/11/1`	Enters interface configuration mode.
Step 4	**cdp** **Example:** `RP/0/RP0/CPU0:router(config-if)# cdp`	Enables CDP on an interface.
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.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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.

SUMMARY STEPS

1. **configure**
2. **cdp advertise v1**
3. **cdp holdtime** *seconds*
4. **cdp timer** *seconds*
5. Use the **commit** or **end** command.
6. (Optional) **show cdp**

DETAILED STEPS

	Command or Action	Purpose
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.

	Command or Action	Purpose
		• 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.

SUMMARY STEPS

1. **show cdp entry** {***** | *entry-name*} [**protocol** | **version**]
2. **show cdp interface** [*type interface-path-id* | **location** *node-id*]
3. **show cdp neighbors** [*type interface-path-id* | **location** *node-id*] [**detail**]
4. **show cdp traffic** [**location** *node-id*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show cdp entry** {***** | *entry-name*} [**protocol** | **version**] **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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
  asr9k-rtr1       Te0/5/0/11/1    152      R           ASR9K Ser Te0/1/0/9
  asr9k-rtr1       Te0/5/0/11/2    156      R           ASR9K Ser Te0/1/0/10
  asr9k-rtr1       Te0/5/0/11/3    160      R           ASR9K Ser Te0/1/0/11
```

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 TenGigE 0/5/0/11/1 detail

  -------------------------
  Device ID: asr9k-rtr1
  SysName : asr9k-rtr1
  Entry address(es):
  IPv4 address: 90.0.0.2
  Platform: cisco ASR9K Series, Capabilities: Router
  Interface: TenGigE 0/5/0/11/1
  Port ID (outgoing port): TenGigE 0/1/0/9
  Holdtime : 155 sec

  Version :
  Cisco IOS XR Software, Version 5.3.1.10I[Default]
  Copyright (c) 2015 by Cisco Systems, Inc.

  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 asr9k-rtr1

-------------------------
Device ID: asr9k-rtr1
SysName : asr9k-rtr1
Entry address(es):
IPv4 address: 110.0.0.2
Platform: cisco ASR9K Series, Capabilities: Router
Interface: TenGigE 0/5/0/11/3
Port ID (outgoing port): TenGigE 0/1/0/11
Holdtime : 173 sec

Version :
Cisco IOS XR Software, Version 5.3.1.10I[Default]
Copyright (c) 2015 by Cisco Systems, Inc.

advertisement version: 2
Duplex: full

-------------------------
Device ID: asr9k-rtr1
SysName : asr9k-rtr1
Entry address(es):
IPv4 address: 100.0.0.2
Platform: cisco ASR9K Series, Capabilities: Router
Interface: TenGigE 0/5/0/11/2
Port ID (outgoing port): TenGigE 0/1/0/10
Holdtime : 169 sec

Version :
Cisco IOS XR Software, Version 5.3.1.10I[Default]
Copyright (c) 2015 by Cisco Systems, Inc.

advertisement version: 2
Duplex: full

-------------------------
Device ID: asr9k-rtr1
SysName : asr9k-rtr1
```

```
Entry address(es):
IPv4 address: 90.0.0.2
Platform: cisco ASR9K Series, Capabilities: Router
Interface: TenGigE 0/5/0/11/1
Port ID (outgoing port): TenGigE 0/1/0/10
Holdtime : 165 sec

Version :
Cisco IOS XR Software, Version 5.3.1.10I[Default]
Copyright (c) 2015 by Cisco Systems, Inc.

advertisement version: 2
Duplex: full
```

The following is sample output for the **show cdp interface** command. In this example, CDP information related to Packet over SONET/SDH (POS) interface 0/4/0/0 is displayed.

```
RP/0/RP0/CPU0:router#  show cdp interface TenGigE 0/5/0/11/1

  TenGigE 0/5/0/11/1 is Up
    Encapsulation ether
    Sending CDP packets every 20 seconds
    Holdtime is 30 seconds
```

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

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

  CDP counters :
          Packets output: 250, Input: 120
          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: 250, Input: 120
          Unrecognize Hdr version: 0, File open failed: 0
```

The following is sample output for the **show cdp traffic** command. In this example, the optional **location** keyword and *node-id* argument are used to display information about the traffic gathered between devices using CDP from the specified node.

```
RP/0/RP0/CPU0:router# show cdp traffic 0/5/CPU0

  CDP counters :
          Packets output: 318, Input: 141
          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: 318, Input: 141
          Unrecognize Hdr version: 0, File open failed: 0
```

Configuration Examples for Implementing CDP

Enabling CDP: Example

The following example shows how to configure CDP globally and then enable CDP on Ethernet interface TenGigE 0/5/0/11/1:

```
cdp
interface 0/5/0/11/1
cdp
```

Modifying Global CDP Settings: Example

The following example shows how to modify global CDP settings. In this example, the timer setting is set to 20 seconds, the hold-time setting is set to 30 seconds, and the version of CDP used to communicate with neighboring devices is set to CDPv1:

```
cdp timer 20
cdp holdtime 30
cdp advertise v1
```

The following example shows how to use the **show cdp** command to verify the CDP global settings:

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

  Global CDP information:
  Sending CDP packets every 20 seconds
  Sending a holdtime value of 30 seconds
  Sending CDPv2 advertisements is not enabled
```