Network Management Configuration Guide, Cisco IOS XE Everest 16.6.x (Catalyst 9500 Switches)

First Published: 2017-07-31

Last Modified: 2017-11-03

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

© 2017 Cisco Systems, Inc. All rights reserved.

CONTENTS

CHAPTER 1 **Configuring Autoconf** 1

Prerequisites for Autoconf 1

Restrictions for Autoconf 1

Information About Autoconf 2

Benefits of Autoconf 2

Identity Session Management and Templates 2

Autoconf Operation 3

Advantages of Using Templates 5

Autoconf Functionality 6

How to Configure Autoconf 7

Applying a Built-in Template to an End Device 7

Applying a Modified Built-in Template to an End Device 11

Migrating from ASP to Autoconf 12

Configuration Examples for Autoconf 13

Example: Applying a Built-in Template to an End Device 13

Example: Applying a Modified Built-in Template to an End Device 14

Example: Migrating from ASP Macros to Autoconf 14

Additional References for Autoconf 14

Feature History for Autoconf 15

CHAPTER 2 **Configuring Cisco Plug and Play** 17

Configuring Cisco Plug and Play 17

CHAPTER 3 **Configuring the Cisco Discovery Protocol** 19

Information About CDP 19

Cisco Discovery Protocol Overview 19

Default Cisco Discovery Protocol Configuration 20
How to Configure CDP 20
Configuring Cisco Discovery Protocol Characteristics 20
Disabling Cisco Discovery Protocol 21
Enabling Cisco Discovery Protocol 23
Disabling Cisco Discovery Protocol on an Interface 24
Enabling Cisco Discovery Protocol on an Interface 25
Monitoring and Maintaining Cisco Discovery Protocol 27

CHAPTER 4 **Configuring Cisco Discovery Protocol Bypass 29**
Restrictions for Cisco Discovery Protocol Bypass 29
Information about Cisco Discovery Protocol Bypass 29
How to configure Cisco Discovery Protocol Bypass 30
Configuration Examples for Cisco Discovery Protocol Bypass 31
Example: Enabling Cisco Discovery Protocol Bypass 31
Displaying Cisco Discovery Protocol neighbours 31
Example:Disabling Cisco Discovery Protocol Bypass 32
Feature Information for Cisco Discovery Protocol Bypass 32

CHAPTER 5 **Configuring Simple Network Management Protocol 33**
Prerequisites for SNMP 33
Restrictions for SNMP 35
Information About SNMP 35
SNMP Overview 35
SNMP Manager Functions 36
SNMP Agent Functions 36
SNMP Community Strings 37
SNMP MIB Variables Access 37
SNMP Notifications 37
SNMP ifIndex MIB Object Values 38
SNMP ENTITY-MIB Identifiers 38
Default SNMP Configuration 38
SNMP Configuration Guidelines 39
How to Configure SNMP 39

Configuring Community Strings 39
Configuring SNMP Groups and Users 42
Configuring SNMP Notifications 45
Setting the Agent Contact and Location Information 48
Limiting TFTP Servers Used Through SNMP 49
Disabling the SNMP Agent 50
Monitoring SNMP Status 52
SNMP Examples 52
Feature History for Simple Network Management Protocol 53

CHAPTER 6 **Configuring Service Level Agreements 55**
Restrictions on SLAs 55
Information About SLAs 55
Cisco IOS IP Service Level Agreements (SLAs) 55
Network Performance Measurement with Cisco IOS IP SLAs 56
IP SLA Responder and IP SLA Control Protocol 57
Response Time Computation for IP SLAs 58
IP SLAs Operation Scheduling 59
IP SLA Operation Threshold Monitoring 59
UDP Jitter 60
How to Configure IP SLAs Operations 60
Default Configuration 60
Configuration Guidelines 61
Configuring the IP SLA Responder 61
Implementing IP SLA Network Performance Measurement 63
Analyzing IP Service Levels by Using the UDP Jitter Operation 66
Analyzing IP Service Levels by Using the ICMP Echo Operation 70
Monitoring IP SLA Operations 73
Monitoring IP SLA Operation Examples 74
Additional References 75

CHAPTER 7 **Configuring SPAN and RSPAN 77**
Prerequisites for SPAN and RSPAN 77
Restrictions for SPAN and RSPAN 77

Information About SPAN and RSPAN 79
SPAN and RSPAN 79
Local SPAN 79
Remote SPAN 80
SPAN and RSPAN Concepts and Terminology 81
SPAN and RSPAN Interaction with Other Features 86
SPAN and RSPAN and Device Stacks 87
Flow-Based SPAN 87
Default SPAN and RSPAN Configuration 88
Configuration Guidelines 88
SPAN Configuration Guidelines 88
RSPAN Configuration Guidelines 88
FSPAN and FRSPAN Configuration Guidelines 89
How to Configure SPAN and RSPAN 89
Creating a Local SPAN Session 89
Creating a Local SPAN Session and Configuring Incoming Traffic 92
Specifying VLANs to Filter 94
Configuring a VLAN as an RSPAN VLAN 96
Creating an RSPAN Source Session 97
Specifying VLANs to Filter 99
Creating an RSPAN Destination Session 101
Creating an RSPAN Destination Session and Configuring Incoming Traffic 103
Configuring an FSPAN Session 105
Configuring an FRSPAN Session 108
Monitoring SPAN and RSPAN Operations 111
SPAN and RSPAN Configuration Examples 111
Example: Configuring Local SPAN 111
Examples: Creating an RSPAN VLAN 112
Feature History for SPAN and RSPAN 113

CHAPTER 8 **Configuring ERSPAN 115**
Configuring ERSPAN 115
Prerequisites for Configuring ERSPAN 115
Restrictions for Configuring ERSPAN 115

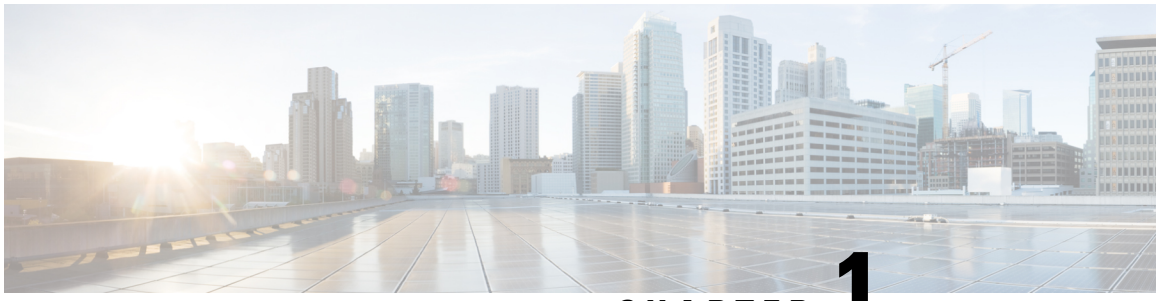
Information for Configuring ERSPAN 116
ERSPAN Overview 116
ERSPAN Sources 117
How to Configure ERSPAN 117
Configuring an ERSPAN Source Session 117
Configuration Examples for ERSPAN 119
Example: Configuring an ERSPAN Source Session 119
Verifying ERSPAN 119
Additional References 121
Feature Information for Configuring ERSPAN 121

CHAPTER 9 **Configuring Packet Capture** 123
Prerequisites for Configuring Packet Capture 123
Prerequisites for Configuring Wireshark 123
Prerequisites for Configuring Embedded Packet Capture 123
Restrictions for Configuring Packet Capture 124
Restrictions for Configuring Wireshark 124
Restrictions for Embedded Packet Capture 125
Information About Packet Capture 126
About Wireshark 126
Capture Points 126
Attachment Points 126
Filters 127
Actions 127
Storage of Captured Packets to Buffer in Memory 128
Storage of Captured Packets to a .pcap File 128
Packet Decoding and Display 129
Packet Storage and Display 129
Wireshark Capture Point Activation and Deactivation 130
Wireshark Features 130
Guidelines for Configuring Wireshark 131
Default Wireshark Configuration 133
Feature Enhancements for Wireshark 133
About Embedded Packet Capture 134

Benefits of Embedded Packet Capture 134
Packet Data Capture 134
Feature Enhancements for Embedded Packet Capture 135
How to Configure Packet Capture 135
How to Configure Wireshark 135
Defining a Capture Point 135
Adding or Modifying Capture Point Parameters 140
Deleting Capture Point Parameters 142
Deleting a Capture Point 143
Activating and Deactivating a Capture Point 144
Clearing the Capture Point Buffer 147
How to Implement Embedded Packet Capture 148
Managing Packet Data Capture 148
Monitoring and Maintaining Captured Data 150
Configuration Examples for Packet Capture 151
Configuration Examples for Wireshark 151
Example: Displaying a Brief Output from a .pcap File 151
Example: Displaying Detailed Output from a .pcap File 152
Example: Displaying a Packet Dump Output from a .pcap File. 153
Example: Displaying Packets from a .pcap File using a Display Filter 154
Example: Displaying the Number of Packets Captured in a .pcap File 154
Example: Displaying a Single Packet Dump from a .pcap File 154
Example: Displaying Statistics of Packets Captured in a .pcap File 155
Example: Simple Capture and Display 155
Example: Simple Capture and Store 156
Example: Using Buffer Capture 158
Example: Simple Capture and Store of Packets in Egress Direction 164
Configuration Examples for Embedded Packet Capture 166
Example: Managing Packet Data Capture 166
Example: Monitoring and Maintaining Captured Data 166
Additional References 168
Feature History for Configuring Packet Capture 168

CHAPTER 10 **Configuring Flexible NetFlow** 171

Prerequisites for Flexible NetFlow 171
Restrictions for Flexible NetFlow 172
Information About Flexible Netflow 174
Flexible NetFlow Overview 174
Original NetFlow and Benefits of Flexible NetFlow 174
Flexible NetFlow Components 175
Flow Records 176
Flow Exporters 180
Flow Monitors 181
Flow Samplers 183
Supported Flexible NetFlow Fields 184
Default Settings 188
Flexible NetFlow—Ingress VRF Support Overview 188
How to Configure Flexible Netflow 189
Creating a Customized Flow Record 189
Creating a Flow Exporter 191
Creating a Customized Flow Monitor 193
Creating a Flow Sampler 195
Applying a Flow to an Interface 197
Configuring a Bridged NetFlow on a VLAN 198
Configuring Layer 2 NetFlow 199
Monitoring Flexible NetFlow 200
Configuration Examples for Flexible NetFlow 200
Example: Configuring a Flow 200
Example: Monitoring IPv4 ingress traffic 201
Example: Monitoring IPv4 egress traffic 202
Example: Configuring Flexible NetFlow for Ingress VRF Support 203
Additional References for NetFlow 204
Feature Information for Flexible NetFlow 204



CHAPTER 1

Configuring Autoconf

Autoconf is a solution that can be used to manage port configurations for data or voice VLAN, quality of service (QoS) parameters, storm control, and MAC-based port security on end devices that are deployed in the access layer of a network.

- Prerequisites for Autoconf, on page 1
- Restrictions for Autoconf, on page 1
- Information About Autoconf, on page 2
- How to Configure Autoconf, on page 7
- Configuration Examples for Autoconf, on page 13
- Additional References for Autoconf, on page 14
- Feature History for Autoconf, on page 15

Prerequisites for Autoconf

- Before enabling Autoconf, disable the Auto SmartPort (ASP) macro, device classifier, and then access the session monitor.

Restrictions for Autoconf

- ASP macro and Autoconf are not supported on the same interface at the same time. Either Autoconf or ASP must be disabled on a per-interface level.
- Interface templates are not applicable for wireless sessions.
- When the Autoconf feature is enabled using the **autoconf enable** command, the default Autoconf service policy is applied to all the interfaces. No other service policy can be applied globally using the **service-policy** command. To apply a different service policy, you must disable Autoconf on that interface. When a service policy is applied globally, you must disable it before enabling the Autoconf feature.
- When both local (interface level) and global service policies exist, the local policy takes precedence. The global service policy comes into effect only when the local policy is removed.
- Service templates cannot be applied to interfaces, and interface templates cannot be applied to service instances.
- Only one service template can be nested inside an interface template.

- Autoconf does not support the switchover feature.

Information About Autoconf

Benefits of Autoconf

The Autoconf feature permits hardbinding between the end device and the interface. Autoconf falls under the umbrella of the Smart Operations solution. Smart Operations is a comprehensive set of capabilities that can simplify and improve LAN switch deployment. Smart Operations help organizations deliver operational excellence and scale services on the network.

The Autoconf feature automatically applies the needed configurations on the device ports to enable the efficient performance of each directly connected end device using a set of interface configurations that are configured inside an interface template.

- Autoconf efficiently applies commands to an interface because the parser does not need to parse each command each time.
- Configurations that are applied through the Autoconf feature can be reliably removed from a port without impacting previous or subsequent configurations on the port.
- The Autoconf feature provides built-in and user-defined configurations using interface and service templates. Configurations applied through templates can be centrally updated with a single operation.
- Using the Autoconf feature, a configuration can be applied to ports and access sessions.
- The Autoconf feature reduces ongoing maintenance for devices and attached end devices by making them intuitive and autoconfigurable. This reduces operation expenses (OPEX) and lowers the total cost of ownership (TCO).

Identity Session Management and Templates

A key advantage of the Autoconf feature is that the core session management capability is decoupled from the application-specific logic; thus, allowing the same framework to be used regardless of the criteria for policy determination or the nature of the policies applied.

The identity session management infrastructure allows configurations and/or policies to be applied as templates.

Both service and interface templates are named containers of configuration and policy. Service templates may be applied only to access sessions, while interface templates may be applied only to ports. When a service template is applied to an access session, the contained configuration/policy is applied only to the target session and has no impact on other sessions that may be hosted on the same access port. Similarly, when an interface template is applied to an access port, it impacts all traffic exchanged on the port.

The Autoconf feature uses a set of built-in maps and built-in templates. The built-in templates are designed based on best practices for interface configurations. Built-in templates can be modified by the user to include customized configurations, limiting the need to create a new template.

The templates created by users are referred to as user-defined templates. User-defined templates can be defined on the device and can be mapped to any built-in or user-defined trigger.

Use the **show derived-config** command, to view the overall applied configurations applied by Autoconf template and manual configuration. The interface commands shown in the output of **show running-config interface** *type number* command are not necessarily the operational configuration. The Autoconf feature dynamically applies a template to the interface, and overrides any conflicting static configuration that is already applied.

Autoconf Operation

Autoconf uses the Device Classifier to identify the end devices that are connected to a port.

The Autoconf feature uses the device classification information gleaned from Cisco Discovery Protocol, LLDP, DHCP, MAC addresses, and the Organizationally Unique Identifier (OUI) that is identified by the Device Classifier.

The Device Classifier provides improved device classification capabilities and accuracy, and increased device visibility for enhanced configuration management.

Device classification is enabled when you enable the Autoconf feature using **autoconf enable** command in global configuration mode .

The device detection acts as an event trigger, which in turn applies the appropriate automatic template to the interface.

The Autoconf feature is based on a three-tier hierarchy.

- A policy map identifies the trigger type for applying the Autoconf feature.
- A parameter map identifies the appropriate template that must be applied, based on the end device.
- The templates contain the configurations to be applied.

The Autoconf built-in templates and triggers perform the these three steps automatically.

The Autoconf feature provides the following built-in templates:

- AP_INTERFACE_TEMPLATE
- DMP_INTERFACE_TEMPLATE
- IP_CAMERA_INTERFACE_TEMPLATE
- IP_PHONE_INTERFACE_TEMPLATE
- LAP_INTERFACE_TEMPLATE
- MSP_CAMERA_INTERFACE_TEMPLATE
- MSP_VC_INTERFACE_TEMPLATE
- PRINTER_INTERFACE_TEMPLATE
- ROUTER_INTERFACE_TEMPLATE
- SWITCH_INTERFACE_TEMPLATE
- TP_INTERFACE_TEMPLATE

Note By default built-in templates are not displayed under running configuration. The built-in templates show in the running configuration only if you edit them.

The template that is selected is based on parameter map information applied to an interface. This information can be based on the following criteria:

- End Device type
- MAC address
- OUI
- Platform type
- User role
- Username

The Autoconf feature provides one built-in parameter map BUILTIN_DEVICE_TO_TEMPLATE with the following configuration:

```
Parameter-map name: BUILTIN_DEVICE_TO_TEMPLATE
 Map: 10 map device-type regex "Cisco-IP-Phone"
  Action(s):
   20 interface-template IP_PHONE_INTERFACE_TEMPLATE
 Map: 20 map device-type regex "Cisco-IP-Camera"
  Action(s):
   20 interface-template IP_CAMERA_INTERFACE_TEMPLATE
 Map: 30 map device-type regex "Cisco-DMP"
  Action(s):
   20 interface-template DMP_INTERFACE_TEMPLATE
 Map: 40 map oui eq "00.0f.44"
  Action(s):
   20 interface-template DMP_INTERFACE_TEMPLATE
 Map: 50 map oui eq "00.23.ac"
  Action(s):
   20 interface-template DMP_INTERFACE_TEMPLATE
 Map: 60 map device-type regex "Cisco-AIR-AP"
  Action(s):
   20 interface-template AP_INTERFACE_TEMPLATE
 Map: 70 map device-type regex "Cisco-AIR-LAP"
  Action(s):
   20 interface-template LAP_INTERFACE_TEMPLATE
 Map: 80 map device-type regex "Cisco-TelePresence"
  Action(s):
   20 interface-template TP_INTERFACE_TEMPLATE
 Map: 90 map device-type regex "Surveillance-Camera"
  Action(s):
   10 interface-template MSP_CAMERA_INTERFACE_TEMPLATE
 Map: 100 map device-type regex "Video-Conference"
  Action(s):
   10 interface-template MSP_VC_INTERFACE_TEMPLATE
```

Note Use the **show parameter-map type subscriber attribute-to-service All** command to view the configuration for the built-in parameter map.

The Autoconf feature provides one built-in policy map BUILTIN_AUTOCONF_POLICY with the following configuration:

```
BUILTIN_AUTOCONF_POLICY
  event identity-update match-all
    10 class always do-until-failure
      10 map attribute-to-service table BUILTIN_DEVICE_TO_TEMPLATE
```

Note Use the **show policy-map type control subscriber BUILTIN_AUTOCONF_POLICY** command to view the configuration for the built-in policy map.

You can also manually create policy maps, parameter maps, and templates.

When a trigger is created that is based on specific user information, a local 802.1X Cisco Identity Services Engine (ISE) server authenticates it ensuring the security of the operation.

An interface template can be dynamically activated (on an interface) using any of the following methods:

- RADIUS CoA—While Change of Authorization (CoA) commands are targeted to one or more access sessions, any referenced template must be applied to the interface hosting the referenced session.
- RADIUS Access-Accept for client authentication or authorization—Any referenced interface template returned in an Access-Accept must be applied to the port that is hosting the authorized access session.
- Service template—If an interface template is referenced in a service template that is either locally defined or sourced from the AAA server, the interface template must be applied to the interface hosting any access-session on which the service template is applied (add a new command for interface template reference from within a locally defined service template).
- Subscriber control-policy action—A mapping action under the subscriber control policy activates service and/or interface template (as referenced in a parameter map) based on the type of filter, and removes any templates associated with a previous policy.
- Device-to-template parameter map—A subscriber parameter map that allows the filter type to service and/or interface template mappings to be specified in an efficient and readable manner.

Advantages of Using Templates

Using templates for autoconfiguration has the following benefits:

- Templates are parsed once when they are being defined. This makes dynamic application of the templates very efficient.
- Templates can be applied to an Ethernet interface that is connected to an end device, based on the type of the end device.
- Service templates allow the activation of session-oriented features, whereas interface templates apply configurations to the interface that is hosting a session.
- Service templates are applied to access sessions and hence only impact the traffic exchanged with a single endpoint on a port.
- Startup and running configurations of the device are not modified by the dynamic application of the template.
- Policy application is synchronized with the access-session life cycle, which is tracked by the framework by using all available techniques, including just link-up/link-down.

- Templates can be updated with a single operation. All applied instances of the templates are updated.
- Constituent commands of the templates do not appear in the running configuration.
- Templates can be removed with no impact on previous or subsequent configurations.
- Template application is acknowledged, allowing for synchronization and performing remedial actions where failures occur.
- Data VLAN, quality of service (QoS) parameters, storm control, and MAC-based port security are configured automatically based on the end device that is connected to the switch.
- The switch port is cleaned up completely by removing configurations when the device is disconnected from a port.
- Human error is reduced in the installation and configuration process.

Autoconf Functionality

The Autoconf feature is disabled by default in global configuration mode. When you enable the Autoconf feature in global configuration mode, it is enabled by default at the interface level. The built-in template configurations are applied based on the end devices detected on all interfaces.

Use the **access-session inherit disable autoconf** command to manually disable Autoconf at the interface level, even when Autoconf is enabled at the global level.

If you disable Autoconf at the global level, all interface-level configurations are disabled.

Global	Interface Level	AutoConf Status
Disable	Disable	No automatic configurations are applied when an end device is connected.
Enable	Enabled by default	If Autoconf is enabled at the global level, it is enabled at the interface level by default. Built-in template configurations are applied based on the end devices that are detected on all interfaces.
Enable	Disable	Enabled at global level. Disabled at interface level. No automatic configurations are applied when an end device is connected to the interface on which Autoconf is disabled.

Autoconf allows you to retain the template even when the link to the end device is down or the end device is disconnected, by configuring the Autoconf sticky feature. Use the **access-session interface-template sticky** command to configure the Autoconf sticky feature in global configuration mode. The Autoconf sticky feature avoids the need for detecting the end device and applying the template every time the link flaps or device is removed and connected back.

The **access-session interface-template sticky** command is mandatory to apply an inbuilt template that contains **access-session** commands on an interface. Configure the **access-session interface-template sticky** command to apply interface template on a port using a service policy.

If you want to disable the Autoconf feature on a specific interface, use the **access-session inherit disable interface-template-sticky** command in interface configuration mode.

How to Configure Autoconf

Applying a Built-in Template to an End Device

The following task shows how to apply a built-in template on an interface that is connected to an end device, for example, a Cisco IP phone.

Before you begin

Make sure that the end device, for example, a Cisco IP phone, is connected to the switch port.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device(config)# configure terminal`	Enters global configuration mode.
Step 3	**autoconf enable** **Example:** `Device(config)# autoconf enable`	Enables the Autoconf feature.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and enters privileged EXEC mode.
Step 5	**show device classifier attached interface** *interface-type interface-number* **Example:** `Device# show device classifier attached interface Gi3/0/26`	(Optional) Displays whether the end device is classified by the device classifier with correct attributes.
Step 6	**show template binding target** *interface-type interface-number* **Example:** `Device# show template binding target gi3/0/26`	Displays the configuration applied through the template on the interface.

Example

The following example shows that an IP phone is classified by the Device Classifier with correct attributes:

```
Device# show device classifier attached interface GigabitEthernet 3/0/26

Summary:

MAC_Address       Port_Id       Profile Name                        Device Name
==============    ==========    ==================                  ==================
0026.0bd9.7bbb    Gi3/0/26      Cisco-IP-Phone-7962                 Cisco IP Phone 7962
```

The following example shows that a built-in interface template is applied on the interface:

```
Device# show template binding target GigabitEthernet 3/0/26

Interface Templates
===================
Interface: Gi4/0/11
Method              Source            Template-Name
------              ------            -------------
dynamic             Built-in          IP_PHONE_INTERFACE_TEMPLATE
```

The following example shows how to verify the interface configuration after the interface template is applied to the IP phone connected to the GigabitEthernet interface 3/0/26:

```
Device# show running-config interface GigabitEthernet 3/0/26
Building configuration...

Current configuration : 624 bytes
!
interface GigabitEthernet3/0/26
!
End

Device# show derived-config interface GigabitEthernet 3/0/26

Building configuration...

Derived configuration : 649 bytes
!
interface GigabitEthernet3/0/26
 switchport mode access
 switchport block unicast
 switchport port-security maximum 3
 switchport port-security maximum 2 vlan access
 switchport port-security violation restrict
 switchport port-security aging time 2
 switchport port-security aging type inactivity
 switchport port-security
 load-interval 30
 storm-control broadcast level pps 1k
 storm-control multicast level pps 2k
 storm-control action trap
 spanning-tree portfast
 spanning-tree bpduguard enable
 service-policy input AutoConf-4.0-CiscoPhone-Input-Policy
 service-policy output AutoConf-4.0-Output-Policy
 ip dhcp snooping limit rate 15
end
```

The following example shows how to verify the global configuration after applying autoconf:

```
Device# show running config
class-map match-any AutoConf-4.0-Scavenger-Queue
 match dscp cs1
 match cos  1
 match access-group name AutoConf-4.0-ACL-Scavenger
class-map match-any AutoConf-4.0-VoIP
 match dscp ef
 match cos  5
class-map match-any AutoConf-4.0-Control-Mgmt-Queue
 match cos  3
 match dscp cs7
 match dscp cs6
 match dscp cs3
 match dscp cs2
 match access-group name AutoConf-4.0-ACL-Signaling
class-map match-any AutoConf-4.0-Multimedia-Conf
 match dscp af41
 match dscp af42
 match dscp af43
class-map match-all AutoConf-4.0-Broadcast-Vid
 match dscp cs5
class-map match-any AutoConf-4.0-Bulk-Data
 match dscp af11
 match dscp af12
 match dscp af13
class-map match-all AutoConf-4.0-Realtime-Interact
 match dscp cs4
class-map match-any AutoConf-4.0-VoIP-Signal
 match dscp cs3
 match cos  3
class-map match-any AutoConf-4.0-Trans-Data-Queue
 match cos  2
 match dscp af21
 match dscp af22
 match dscp af23
 match access-group name AutoConf-4.0-ACL-Transactional-Data
class-map match-any AutoConf-4.0-VoIP-Data
 match dscp ef
 match cos  5
class-map match-any AutoConf-4.0-Multimedia-Stream
 match dscp af31
 match dscp af32
 match dscp af33
class-map match-all AutoConf-4.0-Internetwork-Ctrl
 match dscp cs6
class-map match-all AutoConf-4.0-VoIP-Signal-Cos
 match cos  3
class-map match-any AutoConf-4.0-Multimedia-Stream-Queue
 match dscp af31
 match dscp af32
 match dscp af33
class-map match-all AutoConf-4.0-Network-Mgmt
 match dscp cs2
class-map match-all AutoConf-4.0-VoIP-Data-Cos
 match cos  5
class-map match-any AutoConf-4.0-Priority-Queue
 match cos  5
 match dscp ef
 match dscp cs5
 match dscp cs4
class-map match-any AutoConf-4.0-Bulk-Data-Queue
 match cos  1
 match dscp af11
 match dscp af12
```

```
 match dscp af13
 match access-group name AutoConf-4.0-ACL-Bulk-Data
class-map match-any AutoConf-4.0-Transaction-Data
 match dscp af21
 match dscp af22
 match dscp af23
class-map match-any AutoConf-4.0-Multimedia-Conf-Queue
 match cos  4
 match dscp af41
 match dscp af42
 match dscp af43
 match access-group name AutoConf-4.0-ACL-Multimedia-Conf
class-map match-all AutoConf-4.0-Network-Ctrl
 match dscp cs7
class-map match-all AutoConf-4.0-Scavenger
 match dscp cs1
class-map match-any AutoConf-4.0-Signaling
 match dscp cs3
 match cos  3
!
!
policy-map AutoConf-4.0-Cisco-Phone-Input-Policy
 class AutoConf-4.0-VoIP-Data-Cos
  set dscp ef
  police cir 128000 bc 8000
   exceed-action set-dscp-transmit cs1
   exceed-action set-cos-transmit 1
 class AutoConf-4.0-VoIP-Signal-Cos
  set dscp cs3
  police cir 32000 bc 8000
   exceed-action set-dscp-transmit cs1
   exceed-action set-cos-transmit 1
 class class-default
  set dscp default
  set cos 0
policy-map AutoConf-4.0-Output-Policy
 class AutoConf-4.0-Scavenger-Queue
  bandwidth remaining percent 1
 class AutoConf-4.0-Priority-Queue
  priority
  police cir percent 30 bc 33 ms
 class AutoConf-4.0-Control-Mgmt-Queue
  bandwidth remaining percent 10
 class AutoConf-4.0-Multimedia-Conf-Queue
  bandwidth remaining percent 10
 class AutoConf-4.0-Multimedia-Stream-Queue
  bandwidth remaining percent 10
 class AutoConf-4.0-Trans-Data-Queue
  bandwidth remaining percent 10
    dbl
 class AutoConf-4.0-Bulk-Data-Queue
  bandwidth remaining percent 4
    dbl
 class class-default
  bandwidth remaining percent 25
    dbl
policy-map AutoConf-DMP
 class class-default
  set dscp cs2
policy-map AutoConf-IPVSC
 class class-default
  set cos dscp table AutoConf-DscpToCos
policy-map AutoConf-4.0-Input-Policy
 class AutoConf-4.0-VoIP
```

```
class AutoConf-4.0-Broadcast-Vid
class AutoConf-4.0-Realtime-Interact
class AutoConf-4.0-Network-Ctrl
class AutoConf-4.0-Internetwork-Ctrl
class AutoConf-4.0-Signaling
class AutoConf-4.0-Network-Mgmt
class AutoConf-4.0-Multimedia-Conf
class AutoConf-4.0-Multimedia-Stream
class AutoConf-4.0-Transaction-Data
class AutoConf-4.0-Bulk-Data
class AutoConf-4.0-Scavenger
```

Applying a Modified Built-in Template to an End Device

The following task shows how to modify a built-in template when multiple wireless access points and IP cameras are connected to a switch.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device(config)# configure terminal`	Enters global configuration mode.
Step 3	**template** *template-name* **Example:** `Device(config)# template AP_INTERFACE_TEMPLATE`	Enters template configuration mode for the builtin template.
Step 4	**switchport access vlan** *vlan-id* **Example:** `Device(config-template)# switchport access vlan 20`	Sets the VLAN when the interface is in access mode.
Step 5	**description** *description* **Example:** `Device(config-template)# description modifiedAP`	Modifies the description of the built-in template.
Step 6	**exit** **Example:** `Device(config-template)# exit`	Exits template configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 7	**autoconf enable** **Example:** `Device(config)# autoconf enable`	Enables the Autoconf feature.
Step 8	**end** **Example:** `Device(config)# end`	Exits global configuration mode and enters privileged EXEC mode.
Step 9	**show template interface binding all** **Example:** `Device# show template interface binding all`	Displays whether the template is applied on the interface.

Example

The following example shows that the IP camera and access points are classified by the Device Classifier with correct attributes:

```
Device# show device classifier attached detail

DC default profile file version supported = 1

Detail:
MAC_Address     Port_Id    Cert Parent Proto    ProfileType     Profile Name
Device_Name
=============================================================================================================
001d.a1ef.23a8  Gi1/0/7     30    3    C     M    Default      Cisco-AIR-AP-1130      cisco
 AIR-AP1131AG-A-K9
001e.7a26.eb05  Gi1/0/30    70    2    C     M    Default      Cisco-IP-Camera        Cisco
 IP Camera
```

The following example shows that a built-in interface template is applied on the interface:

```
Device# show template interface binding all

Template-Name                   Source            Method          Interface
-------------                   ------            ------          ---------
IP_CAMERA_INTERFACE_TEMPLATE     Built-in          dynamic         Gi1/0/30
AP_INTERFACE_TEMPLATE          Modified-Built-in   dynamic         Gi1/0/7
```

Migrating from ASP to Autoconf

Before you begin

Verify that the AutoSmart Port (ASP) macro is running using the **show running-config | include macro auto global** command.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no macro auto global processing** **Example:** `Device(config)# no macro auto global processing`	Disables ASP on a global level.
Step 4	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 5	**clear macro auto configuration all** **Example:** `Device# clear macro auto configuration all`	Clears macro configurations for all interfaces.
Step 6	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 7	**autoconf enable** **Example:** `Device(config)# autoconf enable`	Enables the Autoconf feature.
Step 8	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuration Examples for Autoconf

Example: Applying a Built-in Template to an End Device

The following example shows how to apply a built-in template to an end device connected to an interface.

```
Device> enable
Device(config)# configure terminal
Device(config)# autoconf enable
Device(config)# end
Device# show device classifier attached interface Gi3/0/26
Device# show template binding target GigabitEthernet 3/0/26
```

Example: Applying a Modified Built-in Template to an End Device

The following example shows how to modified built-in template and verify the configuration:

```
Device> enable
Device(config)# configure terminal
Device(config)# template AP_INTERFACE_TEMPLATE
Device(config-template)# switchport access vlan 20
Device(config-template)# description modifiedAP
Device(config-template)# exit
Device(config)# autoconf enable
Device(config)# end
Device# show template interface binding all
```

Example: Migrating from ASP Macros to Autoconf

The following example shows how to migrate from ASP to Autoconf:

```
Device> enable
Device# configure terminal
Device(config)# no macro auto global processing
Device(config)# exit
Device# clear macro auto configuration all
Device# configure terminal
Device(config)# autoconf enable
Device(config)# end
```

Additional References for Autoconf

Related Documents

Related Topic	Document Title
Cisco identity-based networking services commands	Cisco IOS Identity-Based Networking Services Command Reference
Interface Templates	"Interface Templates" chapter in Identity-Based Networking Services Configuration Guide.

Standards and RFCs

Standard/RFC	Title
IEEE 802.1X	*Port Based Network Access Control*

Technical Assistance

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

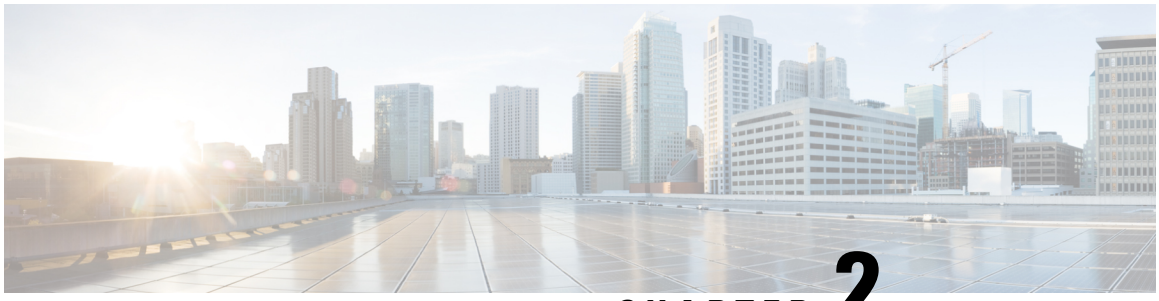
Feature History for Autoconf

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.6.1	Autoconf	The feature was introduced. The Autoconf feature permits hardbinding between an end device and an interface. The following commands were added or modified: **autoconf enable**, **map attribute-to-service (autoconf)**, **map device-type (service-template)**, **parameter-map type subscriber (service-template)**, **show parameter-map type subscriber attribute-to-service all**, **show template interface**.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com.



CHAPTER 2

Configuring Cisco Plug and Play

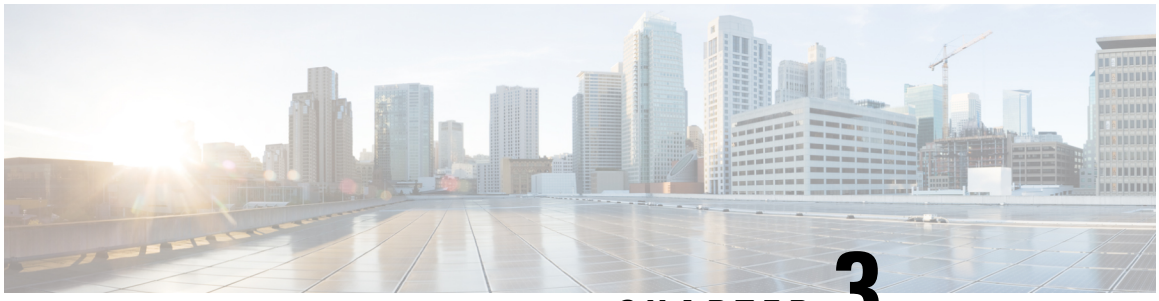
- Configuring Cisco Plug and Play, on page 17

Configuring Cisco Plug and Play

For information about configuring Plug and Play, see

- Cisco Plug and Play Feature Guide
- Configuration Guide for Cisco Network Plug and Play on APIC-EM

Configuring Cisco Plug and Play



CHAPTER 3

Configuring the Cisco Discovery Protocol

Cisco Discovery Protocol is a Layer 2, media-independent, and network-independent protocol that runs on Cisco devices and enables networking applications to learn about directly connected devices nearby. This protocol facilitates the management of Cisco devices by discovering these devices, determining how they are configured, and allowing systems using different network-layer protocols to learn about each other.

This module describes Cisco Discovery Protocol Version 2 and how it functions with SNMP.

- Information About CDP, on page 19
- How to Configure CDP, on page 20
- Monitoring and Maintaining Cisco Discovery Protocol, on page 27

Information About CDP

Cisco Discovery Protocol Overview

Cisco Discovery Protocol is a device discovery protocol that runs over Layer 2 (the data-link layer) on all Cisco-manufactured devices (routers, bridges, access servers, controllers, and switches) and allows network management applications to discover Cisco devices that are neighbors of already known devices. With Cisco Discovery Protocol, network management applications can learn the device type and the SNMP agent address of neighboring devices running lower-layer, transparent protocols. This feature enables applications to send SNMP queries to neighboring devices.

Cisco Discovery Protocol runs on all media that support Subnetwork Access Protocol (SNAP). Because Cisco Discovery Protocol runs over the data-link layer only, two systems that support different network-layer protocols can learn about each other.

Each Cisco Discovery Protocol-configured device sends periodic messages to a multicast address, advertising at least one address at which it can receive SNMP messages. The advertisements also contain time-to-live, or holdtime information, which is the length of time a receiving device holds Cisco Discovery Protocol information before discarding it. Each device also listens to the messages sent by other devices to learn about neighboring devices.

On the device, Cisco Discovery Protocol enables Network Assistant to display a graphical view of the network. The device uses Cisco Discovery Protocol to find cluster candidates and maintain information about cluster members and other devices up to three cluster-enabled devices away from the command device by default.

- Cisco Discovery Protocol identifies connected endpoints that communicate directly with the device.

- To prevent duplicate reports of neighboring devices, only one wired device reports the location information.
- The wired device and the endpoints both send and receive location information.

Default Cisco Discovery Protocol Configuration

This table shows the default Cisco Discovery Protocol configuration.

Feature	Default Setting
Cisco Discovery Protocol global state	Enabled
Cisco Discovery Protocol interface state	Enabled
Cisco Discovery Protocol timer (packet update frequency)	60 seconds
Cisco Discovery Protocol holdtime (before discarding)	180 seconds
Cisco Discovery Protocol Version-2 advertisements	Enabled

How to Configure CDP

Configuring Cisco Discovery Protocol Characteristics

You can configure these Cisco Discovery Protocol characteristics:

- Frequency of Cisco Discovery Protocol updates
- Amount of time to hold the information before discarding it
- Whether or not to send Version 2 advertisements

Note Steps 3 through 5 are all optional and can be performed in any order.

Follow these steps to configure the Cisco Discovery Protocol characteristics.

Procedure

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

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**cdp timer** *seconds* **Example:** `Device(config)# cdp timer 20`	(Optional) Sets the transmission frequency of Cisco Discovery Protocol updates in seconds. The range is 5 to 254; the default is 60 seconds.
Step 4	**cdp holdtime** *seconds* **Example:** `Device(config)# cdp holdtime 60`	(Optional) Specifies the amount of time a receiving device should hold the information sent by your device before discarding it. The range is 10 to 255 seconds; the default is 180 seconds.
Step 5	**cdp advertise-v2** **Example:** `Device(config)# cdp advertise-v2`	(Optional) Configures Cisco Discovery Protocol to send Version 2 advertisements. This is the default state.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

Use the **no** form of the Cisco Discovery Protocol commands to return to the default settings.

Disabling Cisco Discovery Protocol

Cisco Discovery Protocol is enabled by default.

Note Device clusters and other Cisco devices (such as Cisco IP Phones) regularly exchange Cisco Discovery Protocol messages. Disabling Cisco Discovery Protocol can interrupt cluster discovery and device connectivity.

Follow these steps to disable the Cisco Discovery Protocol device discovery capability.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no cdp run** **Example:** `Device(config)# no cdp run`	Disables Cisco Discovery Protocol.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

You must reenable Cisco Discovery Protocol to use it.

Enabling Cisco Discovery Protocol

Cisco Discovery Protocol is enabled by default.

Note Device clusters and other Cisco devices (such as Cisco IP Phones) regularly exchange Cisco Discovery Protocol messages. Disabling Cisco Discovery Protocol can interrupt cluster discovery and device connectivity.

Follow these steps to enable Cisco Discovery Protocol when it has been disabled.

Before you begin

Cisco Discovery Protocol must be disabled, or it cannot be enabled.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**cdp run** **Example:** `Device(config)# cdp run`	Enables Cisco Discovery Protocol if it has been disabled.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device# copy running-config startup-config`	

What to do next

Use the **show run all** command to show that Cisco Discovery Protocol has been enabled. If you enter only **show run**, the enabling of Cisco Discovery Protocol may not be displayed.

Disabling Cisco Discovery Protocol on an Interface

Cisco Discovery Protocol is enabled by default on all supported interfaces to send and to receive Cisco Discovery Protocol information.

Note Device clusters and other Cisco devices (such as Cisco IP Phones) regularly exchange Cisco Discovery Protocol messages. Disabling Cisco Discovery Protocol can interrupt cluster discovery and device connectivity.

Note Cisco Discovery Protocol bypass is not supported and may cause a port go into err-disabled state.

Follow these steps to disable Cisco Discovery Protocol on a port.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Specifies the interface on which you are disabling Cisco Discovery Protocol, and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**no cdp enable** **Example:** `Device(config-if)# no cdp enable`	Disables Cisco Discovery Protocol on the interface specified in Step 3.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Enabling Cisco Discovery Protocol on an Interface

Cisco Discovery Protocol is enabled by default on all supported interfaces to send and to receive Cisco Discovery Protocol information.

Note Device clusters and other Cisco devices (such as Cisco IP Phones) regularly exchange Cisco Discovery Protocol messages. Disabling Cisco Discovery Protocol can interrupt cluster discovery and device connectivity.

Note Cisco Discovery Protocol bypass is not supported and may cause a port go into err-disabled state.

Follow these steps to enable Cisco Discovery Protocol on a port on which it has been disabled.

Before you begin

Cisco Discovery Protocol must be disabled on the port that you are trying to Cisco Discovery Protocol enable on, or it cannot be enabled.

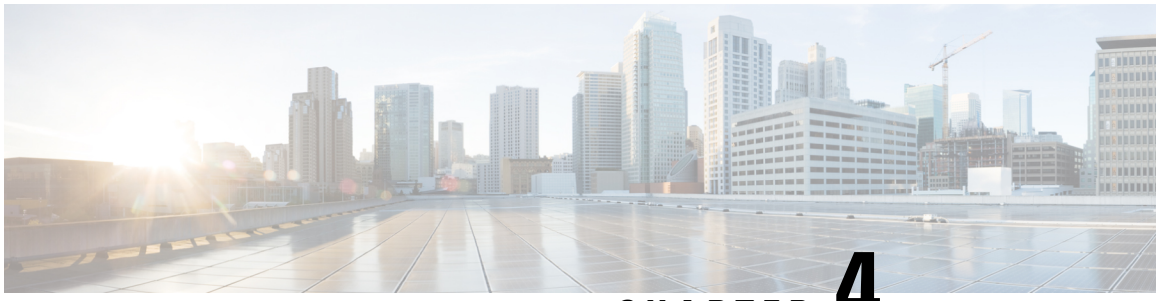
Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet1/0/1`	Specifies the interface on which you are enabling Cisco Discovery Protocol, and enters interface configuration mode.
Step 4	**cdp enable** **Example:** `Device(config-if)# cdp enable`	Enables Cisco Discovery Protocol on a disabled interface.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring and Maintaining Cisco Discovery Protocol

Table 1: Commands for Displaying Cisco Discovery Protocol Information

Command	Description
clear cdp counters	Resets the traffic counters to zero.
clear cdp table	Deletes the Cisco Discovery Protocol table of information about
show cdp	Displays global information, such as frequency of transmissions an for packets being sent.
show cdp entry *entry-name* [**version**] [**protocol**]	Displays information about a specific neighbor. You can enter an asterisk (*) to display all Cisco Discovery Proto or you can enter the name of the neighbor about which you want You can also limit the display to information about the protocols e specified neighbor or information about the version of software r device.
show cdp interface [*interface-id*]	Displays information about interfaces where Cisco Discovery Proto You can limit the display to the interface about which you want i
show cdp neighbors [*interface-id*] [*detail*]	Displays information about neighbors, including device type, inte number, holdtime settings, capabilities, platform, and port ID. You can limit the display to neighbors of a specific interface or e display to provide more detailed information.
show cdp traffic	Displays Cisco Discovery Protocol counters, including the numb sent and received and checksum errors.



CHAPTER 4

Configuring Cisco Discovery Protocol Bypass

In Cisco Discovery Protocol Bypass mode Cisco Discovery Protocol packets are received and transmitted unchanged. Received packets are not processed. No packets are generated. In this mode, 'bump-in-the-wire' behavior is applied to Cisco Discovery Protocol packets. This is a backward compatible mode, equivalent to not having Cisco Discovery Protocol support.

- Restrictions for Cisco Discovery Protocol Bypass, on page 29
- Information about Cisco Discovery Protocol Bypass, on page 29
- How to configure Cisco Discovery Protocol Bypass, on page 30
- Configuration Examples for Cisco Discovery Protocol Bypass, on page 31
- Feature Information for Cisco Discovery Protocol Bypass, on page 32

Restrictions for Cisco Discovery Protocol Bypass

Cisco Discovery Protocol Bypass does not support standard ACLs on the switch port.

Information about Cisco Discovery Protocol Bypass

When a Cisco IP Phone is plugged into a port that is configured with a Voice VLAN and single-host mode, the phone will be silently allowed onto the network by way of a feature known as Cisco Discovery Protocol Bypass. The phone (or any device) that sends the appropriate Type Length Value (TLV) in a Cisco Discovery Protocol message will be allowed access to the voice VLAN.

In Cisco Discovery Protocol Bypass mode, Cisco Disocvery Protocol packets are received and transmitted unchanged. Received packets are not processed. No packets are generated. In this mode, 'bump-in-the-wire' behaviour is applied to Cisco Discovery Protocol packets. This is a backward compatible mode, equivalent to not having Cisco Discovery Protocol support.

In Cisco Discovery Protocol Bypass mode authentication sessions are established in single and multi-host modes for IP Phones. However, if voice VLAN and 802.1x on an interface port is enabled, then Cisco Discovery Protocol Bypass is enabled when the host mode is set to single or multi-host mode.

It is possible to use the Multi-Domain Authentication (MDA) feature instead of Cisco Discovery Protocol Bypass feature as it provides better Access Control, Visibility and Authorization.

Note By default the host mode is set to single mode in legacy mode and multi-authentication in the edge mode.

Cisco Discovery Protocol Enhancement for Second Port Disconnect—Allows a Cisco IP phone to send a Cisco Discovery Protocol message to the switch when a host unplugs from behind the phone. The switch is then able to clear any authenticated session for the indirectly connected host, the same as if the host had been directly connected and the switch had detected a link down event. This is supported in latest IP telephones.

Cisco Discovery Protocol Bypass provides no support for third-party phones—Cisco Discovery Protocol Bypass works only with Cisco phones.

How to configure Cisco Discovery Protocol Bypass

Follow these steps to enable Cisco Discovery Protocol Bypass:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters the global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface GigabitEthernet1/0/12`	Specifies a physical port, and enters interface configuration mode. • Valid interfaces are physical ports.
Step 4	**switchport mode access** **Example:** `Device(config-if)# switchport mode access`	Specifies that the interface is in access mode.
Step 5	**switchport access vlan** *vlan id* **Example:** `Device(config-if)# switchport access vlan 10`	Assigns all ports as static-access ports in the same VLAN • If you configure the port as a static-access port, assign it to only one VLAN. The range is 1 to 4094.
Step 6	**switchport voice vlan** *vlan-id* **Example:** `Device(config-if)# switchport voice vlan 3`	Instruct the Cisco IP phone to forward all voice traffic through the specified VLAN. By default, the Cisco IP phone forwards the voice traffic with an 802.1Q priority of 5.

	Command or Action	Purpose
		Valid VLAN IDs are from 1 to 4094 when the enhanced software image (EI) is installed and 1 to 1001 when the standard software image is installed. Do not enter leading zeros.
Step 7	**authentication port-control auto** **Example:** `Device(config-if)# authentication port-control auto`	Enables 802.1x authentication on the port.
Step 8	**authentication host-mode** {**single-host** | **multi-host**} **Example:** `Device(config-if)# authentication host-mode single | multi-host`	The keywords allow the following: • single-host-Single host (client) on an IEEE 802.1X-authorized port. • multi-host-Multiple hosts on an 802.1X-authorized port after a authenticating a single host.
Step 9	**dot1x pae authenticator** **Example:** `Device(config-if)# dot1x pae authenticator`	Enables 802.1X authentication on the port with default parameters

Configuration Examples for Cisco Discovery Protocol Bypass

Example: Enabling Cisco Discovery Protocol Bypass

Cisco Discovery Protocol Bypass is enabled by default once 'Authetication port-control auto' is configured with dotx1 or MAB or if voice vlan is configured on interface along with single/multiple host mode.

This following configuration example configures Cisco Disovery Protocol Bypass when authenticating using MAB.

```
 Device(config)# interface GigabitEthernet1/0/12
Device(config-if)# switchport mode access
Device(config-if)# switchport access vlan 10
Device(config-if)# switchport voice vlan 3
Device(config-if)# authentication port-control auto
Device(config-if)# mab
```

Displaying Cisco Discovery Protocol neighbours

The following configuration example displays Cisco Discovery Protocol neighbours.

```
Device# show cdp neighbors g1/0/37 detail
Device ID: SEP24B657B038DF
Entry address(es):
Platform: Cisco IP Phone 9971,  Capabilities: Host Phone Two-port Mac Relay
Interface: GigabitEthernet1/0/37,  Port ID (outgoing port): Port 1
Holdtime : 157 sec
Second Port Status: Down <<<<<<<<<<
Version :
sip9971.9-1-1SR1
advertisement version: 2
Duplex: full
Power drawn: 12.804 Watts
Power request id: 57146, Power management id: 4
Power request levels are:12804 0 0 0 0
Total cdp entries displayed : 1
```

Example:Disabling Cisco Discovery Protocol Bypass

To disable Cisco Discovery Protocol Bypass,‘Authetication port-control auto’ needs to be removed from the interface.

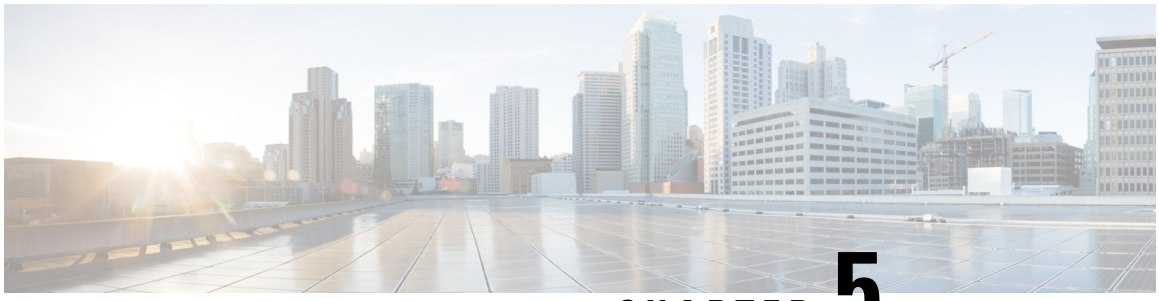
Feature Information for Cisco Discovery Protocol Bypass

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

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

Table 2: Feature Information for Cisco Discovery Protocol Bypass

Releases	Feature Information
Cisco IOS XE Everest 16.5.1a	The feature was introduced.



CHAPTER 5

Configuring Simple Network Management Protocol

- Prerequisites for SNMP, on page 33
- Restrictions for SNMP, on page 35
- Information About SNMP, on page 35
- How to Configure SNMP, on page 39
- Monitoring SNMP Status, on page 52
- SNMP Examples, on page 52
- Feature History for Simple Network Management Protocol, on page 53

Prerequisites for SNMP

Supported SNMP Versions

This software release supports the following SNMP versions:

- SNMPv1—The Simple Network Management Protocol, a Full Internet Standard, defined in RFC 1157.
- SNMPv2C replaces the Party-based Administrative and Security Framework of SNMPv2Classic with the community-string-based Administrative Framework of SNMPv2C while retaining the bulk retrieval and improved error handling of SNMPv2Classic. It has these features:
  - SNMPv2—Version 2 of the Simple Network Management Protocol, a Draft Internet Standard, defined in RFCs 1902 through 1907.
  - SNMPv2C—The community-string-based Administrative Framework for SNMPv2, an Experimental Internet Protocol defined in RFC 1901.
- SNMPv3—Version 3 of the SNMP is an interoperable standards-based protocol defined in RFCs 2273 to 2275. SNMPv3 provides secure access to devices by authenticating and encrypting packets over the network and includes these security features:
  - Message integrity—Ensures that a packet was not tampered with in transit.
  - Authentication—Determines that the message is from a valid source.
  - Encryption—Mixes the contents of a package to prevent it from being read by an unauthorized source.

Note To select encryption, enter the **priv** keyword.

Both SNMPv1 and SNMPv2C use a community-based form of security. The community of managers able to access the agent's MIB is defined by an IP address access control list and password.

SNMPv2C includes a bulk retrieval function and more detailed error message reporting to management stations. The bulk retrieval function retrieves tables and large quantities of information, minimizing the number of round-trips required. The SNMPv2C improved error-handling 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 in SNMPv2C report the error type.

SNMPv3 provides for both security models and security levels. A security model is an authentication strategy set up for a user and the group within which the user resides. A security level is the permitted level of security within a security model. A combination of the security level and the security model determine which security method is used when handling an SNMP packet. Available security models are SNMPv1, SNMPv2C, and SNMPv3.

The following table identifies characteristics and compares different combinations of security models and levels:

Table 3: SNMP Security Models and Levels

Model	Level	Authentication	Encryption	Result
SNMPv1	noAuthNoPriv	Community string	No	Uses a community string match for authentication.
SNMPv2C	noAuthNoPriv	Community string	No	Uses a community string match for authentication.
SNMPv3	noAuthNoPriv	Username	No	Uses a username match for authentication.
SNMPv3	authNoPriv	Message Digest 5 (MD5) or Secure Hash Algorithm (SHA)	No	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms.

Model	Level	Authentication	Encryption	Result
SNMPv3	authPriv	MD5 or SHA	Data Encryption Standard (DES) or Advanced Encryption Standard (AES)	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Allows specifying the User-based Security Model (USM) with these encryption algorithms: • DES 56-bit encryption in addition to authentication based on the CBC-DES (DES-56) standard. • 3DES 168-bit encryption • AES 128-bit, 192-bit, or 256-bit encryption

You must configure the SNMP agent to use the SNMP version supported by the management station. Because an agent can communicate with multiple managers, you can configure the software to support communications using SNMPv1, SNMPv2C, or SNMPv3.

Restrictions for SNMP

Version Restrictions

- SNMPv1 does not support informs.

Information About SNMP

SNMP Overview

SNMP is an application-layer protocol that provides a message format for communication between managers and agents. The SNMP system consists of an SNMP manager, an SNMP agent, and a management information

base (MIB). The SNMP manager can be part of a network management system (NMS) such as Cisco Prime Infrastructure. The agent and MIB reside on the device. To configure SNMP on the device, you define the relationship between the manager and the agent.

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

An agent can send unsolicited traps to the manager. Traps are messages alerting the SNMP manager to a condition on the network. Traps can mean improper user authentication, restarts, link status (up or down), MAC address tracking, closing of a TCP connection, loss of connection to a neighbor, or other significant events.

SNMP Manager Functions

The SNMP manager uses information in the MIB to perform the operations described in the following table:

Table 4: SNMP Operations

Operation	Description
get-request	Retrieves a value from a specific variable.
get-next-request	Retrieves a value from a variable within a table.[1]
get-bulk-request[2]	Retrieves large blocks of data, such as multiple rows in a table, that would otherwise require the transmission of many small blocks of data.
get-response	Replies to a get-request, get-next-request, and set-request sent by an NMS.
set-request	Stores a value in a specific variable.
trap	An unsolicited message sent by an SNMP agent to an SNMP manager when some event has occurred.

[1] With this operation, an SNMP manager does not need to know the exact variable name. A sequential search is performed to find the needed variable from within a table.
[2] The get-bulk command only works with SNMPv2 or later.

SNMP Agent Functions

The SNMP agent responds to SNMP manager requests as follows:

- Get a MIB variable—The SNMP agent begins this function in response to a request from the NMS. The agent retrieves the value of the requested MIB variable and responds to the NMS with that value.
- Set a MIB variable—The SNMP agent begins this function in response to a message from the NMS. The SNMP agent changes the value of the MIB variable to the value requested by the NMS.

The SNMP agent also sends unsolicited trap messages to notify an NMS that a significant event has occurred on the agent. Examples of trap conditions include, but are not limited to, when a port or module goes up or down, when spanning-tree topology changes occur, and when authentication failures occur.

SNMP Community Strings

SNMP community strings authenticate access to MIB objects and function as embedded passwords. In order for the NMS to access the device, the community string definitions on the NMS must match at least one of the three community string definitions on the device.

A community string can have one of the following attributes:

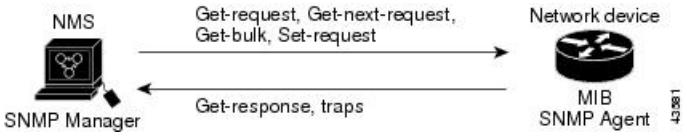
- Read-only (RO)—Gives all objects in the MIB except the community strings read access to authorized management stations, but does not allow write access.
- Read-write (RW)—Gives all objects in the MIB read and write access to authorized management stations, but does not allow access to the community strings.
- When a cluster is created, the command device manages the exchange of messages among member devices and the SNMP application. The Network Assistant software appends the member device number (@esN, where N is the device number) to the first configured RW and RO community strings on the command device and propagates them to the member devices.

SNMP MIB Variables Access

An example of an NMS is the Cisco Prime Infrastructure network management software. Cisco Prime Infrastructure 3.1 software uses the device MIB variables to set device variables and to poll devices on the network for specific information. The results of a poll can be displayed as a graph and analyzed to troubleshoot internetworking problems, increase network performance, verify the configuration of devices, monitor traffic loads, and more.

As shown in the figure, the SNMP agent gathers data from the MIB. The agent can send traps, or notification of certain events, to the SNMP manager, which receives and processes the traps. Traps alert the SNMP manager to a condition on the network such as improper user authentication, restarts, link status (up or down), MAC address tracking, and so forth. The SNMP agent also responds to MIB-related queries sent by the SNMP manager in *get-request*, *get-next-request*, and *set-request* format.

Figure 1: SNMP Network



SNMP Notifications

SNMP allows the device to send notifications to SNMP managers when particular events occur. SNMP notifications can be sent as traps or inform requests. In command syntax, unless there is an option in the command to select either traps or informs, the keyword traps refers to either traps or informs, or both. Use the **snmp-server host** command to specify whether to send SNMP notifications as traps or informs.

Note SNMPv1 does not support informs.

Traps are unreliable because the receiver does not send an acknowledgment when it receives a trap, and the sender cannot determine if the trap was received. When an SNMP manager receives an inform request, it acknowledges the message with an SNMP response protocol data unit (PDU). If the sender does not receive a response, the inform request can be sent again. Because they can be resent, informs are more likely than traps to reach their intended destination.

The characteristics that make informs more reliable than traps also consume more resources in the device and in the network. Unlike a trap, which is discarded as soon as it is sent, an inform request is held in memory until a response is received or the request times out. Traps are sent only once, but an inform might be resent or retried several times. The retries increase traffic and contribute to a higher overhead on the network. Therefore, traps and informs require a trade-off between reliability and resources. If it is important that the SNMP manager receive every notification, use inform requests. If traffic on the network or memory in the device is a concern and notification is not required, use traps.

SNMP ifIndex MIB Object Values

The SNMP agent's IF-MIB module comes up shortly after reboot. As various physical interface drivers are initialized they register with the IF-MIB module, essentially saying "Give me an ifIndex number". The IF-MIB module assigns the next available ifIndex number on a first-come-first-served basis. That is, minor differences in driver initialization order from one reboot to another can result in the same physical interface getting a different ifIndex number than it had before the reboot (unless ifIndex persistency is enabled of course).

SNMP ENTITY-MIB Identifiers

ENTITY-MIB contains information for managing physical entities such as field-replaceable units (FRUs), fans. or power supplies on a device. Each entity is identified by a unique index number-*entPhysicalIndex* that is used to access information about the entity in current and other MIBs. An online insertion and removal (OIR) of the entity results in the entity being assigned the next available *entPhysicalIndex* number, irrespective of whether a new entity is inserted or an existing entity is reinserted.

Default SNMP Configuration

Feature	Default Setting
SNMP agent	Disabled[3].
SNMP trap receiver	None configured.
SNMP traps	None enabled except the trap for TCP connections (tty).
SNMP version	If no version keyword is present, the default is Version 1.
SNMPv3 authentication	If no keyword is entered, the default is the **noauth** (noAuthNoPriv) security level.
SNMP notification type	If no type is specified, all notifications are sent.

[3] This is the default when the device starts and the startup configuration does not have any **snmp-server** global configuration commands.

SNMP Configuration Guidelines

If the device starts and the device startup configuration has at least one **snmp-server** global configuration command, the SNMP agent is enabled.

An SNMP *group* is a table that maps SNMP users to SNMP views. An SNMP *user* is a member of an SNMP group. An SNMP *host* is the recipient of an SNMP trap operation. An SNMP *engine ID* is a name for the local or remote SNMP engine.

When configuring SNMP, follow these guidelines:

- When configuring an SNMP group, do not specify a notify view. The **snmp-server host** global configuration command auto-generates a notify view for the user and then adds it to the group associated with that user. Modifying the group's notify view affects all users associated with that group.
- To configure a remote user, specify the IP address or port number for the remote SNMP agent of the device where the user resides.
- Before you configure remote users for a particular agent, configure the SNMP engine ID, using the **snmp-server engineID** global configuration command with the **remote** option. The remote agent's SNMP engine ID and user password are used to compute the authentication and privacy digests. If you do not configure the remote engine ID first, the configuration command fails.
- When configuring SNMP informs, you need to configure the SNMP engine ID for the remote agent in the SNMP database before you can send proxy requests or informs to it.
- If a local user is not associated with a remote host, the device does not send informs for the **auth** (authNoPriv) and the **priv** (authPriv) authentication levels.
- Changing the value of the SNMP engine ID has significant results. A user's password (entered on the command line) is converted to an MD5 or SHA security digest based on the password and the local engine ID. The command-line password is then destroyed, as required by RFC 2274. Because of this deletion, if the value of the engine ID changes, the security digests of SNMPv3 users become invalid, and you need to reconfigure SNMP users by using the **snmp-server user** *username* global configuration command. Similar restrictions require the reconfiguration of community strings when the engine ID changes.

How to Configure SNMP

Configuring Community Strings

You use the SNMP community string to define the relationship between the SNMP manager and the agent. The community string acts like a password to permit access to the agent on the device. Optionally, you can specify one or more of these characteristics associated with the string:

- An access list of IP addresses of the SNMP managers that are permitted to use the community string to gain access to the agent
- A MIB view, which defines the subset of all MIB objects accessible to the given community
- Read and write or read-only permission for the MIB objects accessible to the community

Follow these steps to configure a community string on the device.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [*access-list-number*] **Example:** `Device(config)# snmp-server community comaccess ro 4`	Configures the community string. **Note** The @ symbol is used for delimiting the context information. Avoid using the @ symbol as part of the SNMP community string when configuring this command. • For *string*, specify a string that acts like a password and permits access to the SNMP protocol. You can configure one or more community strings of any length. • (Optional) For **view**, specify the view record accessible to the community. • (Optional) Specify either read-only (**ro**) if you want authorized management stations to retrieve MIB objects, or specify read-write (**rw**) if you want authorized management stations to retrieve and modify MIB objects. By default, the community string permits read-only access to all objects. • (Optional) For *access-list-number*, enter an IP standard access list numbered from 1 to 99 and 1300 to 1999.
Step 4	**access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] **Example:** `Device(config)# access-list 4 deny any`	(Optional) If you specified an IP standard access list number in Step 3, then create the list, repeating the command as many times as necessary. • For *access-list-number*, enter the access list number specified in Step 3.

	Command or Action	Purpose
		• The **deny** keyword denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched. • For *source*, enter the IP address of the SNMP managers that are permitted to use the community string to gain access to the agent. • (Optional) For *source-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the source. Place ones in the bit positions that you want to ignore. Recall that the access list is always terminated by an implicit deny statement for everything.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

To disable access for an SNMP community, set the community string for that community to the null string (do not enter a value for the community string).

To remove a specific community string, use the **no snmp-server** community string global configuration command.

You can specify an identification name (engine ID) for the local or remote SNMP server engine on the device. You can configure an SNMP server group that maps SNMP users to SNMP views, and you can add new users to the SNMP group.

Configuring SNMP Groups and Users

You can specify an identification name (engine ID) for the local or remote SNMP server engine on the device. You can configure an SNMP server group that maps SNMP users to SNMP views, and you can add new users to the SNMP group.

Follow these steps to configure SNMP groups and users on the device.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server engineID** {**local** *engineid-string* | **remote** *ip-address* [**udp-port** *port-number*] *engineid-string*} **Example:** `Device(config)# snmp-server engineID local 1234`	Configures a name for either the local or remote copy of SNMP. • The *engineid-string* is a 24-character ID string with the name of the copy of SNMP. You need not specify the entire 24-character engine ID if it has trailing zeros. Specify only the portion of the engine ID up to the point where only zeros remain in the value. The Step Example configures an engine ID of 123400000000000000000000. • If you select **remote**, specify the *ip-address* of the device that contains the remote copy of SNMP and the optional User Datagram Protocol (UDP) port on the remote device. The default is 162.
Step 4	**snmp-server group** *group-name* {**v1** | **v2c** | **v3** {**auth** | **noauth** | **priv**}} [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*] **Example:** `Device(config)# snmp-server group public v2c access lmnop`	Configures a new SNMP group on the remote device. For *group-name*, specify the name of the group. Specify one of the following security models: • **v1** is the least secure of the possible security models.

	Command or Action	Purpose
		• **v2c** is the second least secure model. It allows transmission of informs and integers twice the normal width. • **v3**, the most secure, requires you to select one of the following authentication levels: **auth**—Enables the Message Digest 5 (MD5) and the Secure Hash Algorithm (SHA) packet authentication. **noauth**—Enables the noAuthNoPriv security level. This is the default if no keyword is specified. **priv**—Enables Data Encryption Standard (DES) packet encryption (also called privacy). (Optional) Enter **read** *readview* with a string (not to exceed 64 characters) that is the name of the view in which you can only view the contents of the agent. (Optional) Enter **write** *writeview* with a string (not to exceed 64 characters) that is the name of the view in which you enter data and configure the contents of the agent. (Optional) Enter **notify** *notifyview* with a string (not to exceed 64 characters) that is the name of the view in which you specify a notify, inform, or trap. (Optional) Enter **access** *access-list* with a string (not to exceed 64 characters) that is the name of the access list.
Step 5	**snmp-server user** *username group-name* {**remote** *host* [**udp-port** *port*] } {**v1** [**access** *access-list*] | **v2c** [**access** *access-list*] | **v3** [**encrypted**] [**access** *access-list*] [**auth** {**md5** | **sha**} *auth-password*] } [*priv* {**des** | **3des** | **aes** {**128** | **192** | **256**}} *priv-password*] **Example:** `Device(config)# snmp-server user Pat public v2c`	Adds a new user for an SNMP group. The *username* is the name of the user on the host that connects to the agent. The *group-name* is the name of the group to which the user is associated. Enter **remote** to specify a remote SNMP entity to which the user belongs and the hostname or IP address of that entity with the optional UDP port number. The default is 162. Enter the SNMP version number (**v1**, **v2c**, or **v3**). If you enter **v3**, you have these additional options:

	Command or Action	Purpose
		• **encrypted** specifies that the password appears in encrypted format. This keyword is available only when the **v3** keyword is specified. • **auth** is an authentication level setting session that can be either the HMAC-MD5-96 (**md5**) or the HMAC-SHA-96 (**sha**) authentication level and requires a password string *auth-password* (not to exceed 64 characters). If you enter **v3** you can also configure a private (**priv**) encryption algorithm and password string *priv-password* using the following keywords (not to exceed 64 characters): • **priv** specifies the User-based Security Model (USM). • **des** specifies the use of the 56-bit DES algorithm. • **3des** specifies the use of the 168-bit DES algorithm. • **aes** specifies the use of the DES algorithm. You must select either 128-bit, 192-bit, or 256-bit encryption. (Optional) Enter **access** *access-list* with a string (not to exceed 64 characters) that is the name of the access list.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config`	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`startup-config`	

Configuring SNMP Notifications

A trap manager is a management station that receives and processes traps. Traps are system alerts that the device generates when certain events occur. By default, no trap manager is defined, and no traps are sent. Devices running this Cisco IOS release can have an unlimited number of trap managers.

Note Many commands use the word **traps** in the command syntax. Unless there is an option in the command to select either traps or informs, the keyword **traps** refers to traps, informs, or both. Use the **snmp-server host** global configuration command to specify whether to send SNMP notifications as traps or informs.

You can use the **snmp-server enable traps** global configuration command combined with the **snmp-server host** global configuration command for a specific host to receive the notification types listed in the following table. You can enable any or all of these traps and configure a trap manager to receive them.

Note The **snmp-server enable traps** command does not support traps for local-authentication on your device.

Follow these steps to configure the device to send traps or informs to a host.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server engineID remote** *ip-address engineid-string* **Example:** `Device(config)# snmp-server engineID remote 192.180.1.27 00000063000100a1c0b4011b`	Specifies the engine ID for the remote host.

	Command or Action	Purpose
Step 4	**snmp-server user** *username group-name* {**remote** *host* [**udp-port** *port*] } {**v1** [**access** *access-list*] | **v2c** [**access** *access-list*] | **v3** [**encrypted**] [**access** *access-list*] [**auth** {**md5** | **sha**} *auth-password*] } **Example:** `Device(config)# snmp-server user Pat public v2c`	Configures an SNMP user to be associated with the remote host created in Step 3. **Note** You cannot configure a remote user for an address without first configuring the engine ID for the remote host. Otherwise, you receive an error message, and the command is not executed.
Step 5	**snmp-server group** *group-name* {**v1** | **v2c** | **v3** {**auth** | **noauth** | **priv**}} [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*] **Example:** `Device(config)# snmp-server group public v2c access lmnop`	Configures an SNMP group.
Step 6	**snmp-server host** *host-addr* [**informs** | **traps**] [**version** {**1** | **2c** | **3** {**auth** | **noauth** | **priv**}}] *community-string* [*notification-type*] **Example:** `Device(config)# snmp-server host 203.0.113.1 comaccess snmp`	Specifies the recipient of an SNMP trap operation. For *host-addr*, specify the name or Internet address of the host (the targeted recipient). (Optional) Specify **traps** (the default) to send SNMP traps to the host. (Optional) Specify **informs** to send SNMP informs to the host. (Optional) Specify the SNMP **version** (**1**, **2c**, or **3**). SNMPv1 does not support informs. (Optional) For Version 3, select authentication level **auth**, **noauth**, or **priv**. **Note** The **priv** keyword is available only when the cryptographic software image is installed. For *community-string*, when **version 1** or **version 2c** is specified, enter the password-like community string sent with the notification operation. When **version 3** is specified, enter the SNMPv3 username. The @ symbol is used for delimiting the context information. Avoid using the @ symbol as part of the SNMP community string when configuring this command. (Optional) For *notification-type*, use the keywords listed in the table above. If no type is specified, all notifications are sent.

	Command or Action	Purpose
Step 7	**snmp-server enable traps** *notification-types* **Example:** `Device(config)# snmp-server enable traps snmp`	Enables the device to send traps or informs and specifies the type of notifications to be sent. For a list of notification types, see the table above, or enter **snmp-server enable traps ?** To enable multiple types of traps, you must enter a separate **snmp-server enable traps** command for each trap type. **Note** When you configure a trap by using the notification type **port-security**, configure the port security trap first, and then configure the port security trap rate: **a. snmp-server enable traps port-security** **b. snmp-server enable traps port-security trap-rate** *rate*
Step 8	**snmp-server trap-source** *interface-id* **Example:** `Device(config)# snmp-server trap-source gigabitethernet 1/0/1`	(Optional) Specifies the source interface, which provides the IP address for the trap message. This command also sets the source IP address for informs.
Step 9	**snmp-server queue-length** *length* **Example:** `Device(config)# snmp-server queue-length 20`	(Optional) Establishes the message queue length for each trap host. The range is 1 to 5000; the default is 10.
Step 10	**snmp-server trap-timeout** *seconds* **Example:** `Device(config)# snmp-server trap-timeout 60`	(Optional) Defines how often to resend trap messages. The range is 1 to 1000; the default is 30 seconds.
Step 11	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 12	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.

	Command or Action	Purpose
Step 13	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

The **snmp-server host** command specifies which hosts receive the notifications. The **snmp-server enable traps** command globally enables the method for the specified notification (for traps and informs). To enable a host to receive an inform, you must configure an **snmp-server host informs** command for the host and globally enable informs by using the **snmp-server enable traps** command.

To remove the specified host from receiving traps, use the **no snmp-server host** *host* global configuration command. The **no snmp-server host** command with no keywords disables traps, but not informs, to the host. To disable informs, use the **no snmp-server host informs** global configuration command. To disable a specific trap type, use the **no snmp-server enable traps** *notification-types* global configuration command.

Setting the Agent Contact and Location Information

Follow these steps to set the system contact and location of the SNMP agent so that these descriptions can be accessed through the configuration file.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server contact** *text* **Example:** `Device(config)# snmp-server contact Dial System Operator at beeper 21555`	Sets the system contact string.
Step 4	**snmp-server location** *text* **Example:**	Sets the system location string.

	Command or Action	Purpose
	`Device(config)# snmp-server location Building 3/Room 222`	
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Limiting TFTP Servers Used Through SNMP

Follow these steps to limit the TFTP servers used for saving and loading configuration files through SNMP to the servers specified in an access list.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server tftp-server-list** *access-list-number* **Example:**	Limits the TFTP servers used for configuration file copies through SNMP to the servers in the access list.

	Command or Action	Purpose
	`Device(config)# snmp-server tftp-server-list 44`	For *access-list-number*, enter an IP standard access list numbered from 1 to 99 and 1300 to 1999.
Step 4	**access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] **Example:** `Device(config)# access-list 44 permit 10.1.1.2`	Creates a standard access list, repeating the command as many times as necessary. For *access-list-number*, enter the access list number specified in Step 3. The **deny** keyword denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched. For *source*, enter the IP address of the TFTP servers that can access the device. (Optional) For *source-wildcard*, enter the wildcard bits, in dotted decimal notation, to be applied to the source. Place ones in the bit positions that you want to ignore. The access list is always terminated by an implicit deny statement for everything.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Disabling the SNMP Agent

The **no snmp-server** global configuration command disables all running versions (Version 1, Version 2C, and Version 3) of the SNMP agent on the device. You reenable all versions of the SNMP agent by the first **snmp-server** global configuration command that you enter. There is no Cisco IOS command specifically designated for enabling SNMP.

Follow these steps to disable the SNMP agent.

Before you begin

The SNMP Agent must be enabled before it can be disabled. The SNMP agent is enabled by the first **snmp-server** global configuration command entered on the device.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no snmp-server** **Example:** `Device(config)# no snmp-server`	Disables the SNMP agent operation.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring SNMP Status

To display SNMP input and output statistics, including the number of illegal community string entries, errors, and requested variables, use the **show snmp** privileged EXEC command. You also can use the other privileged EXEC commands listed in the table to display SNMP information.

Table 5: Commands for Displaying SNMP Information

Command	Purpose
show snmp	Displays SNMP statistics.
show snmp group	Displays information on each SNMP group on the network.
show snmp pending	Displays information on pending SNMP requests.
show snmp sessions	Displays information on the current SNMP sessions.
show snmp user	Displays information on each SNMP user name in the SNMP users t **Note** You must use this command to display SNMPv3 configuration inform **auth** | **noauth** | **priv** mode. This information is not displayed in the s **running-config** output.

SNMP Examples

This example shows how to enable all versions of SNMP. The configuration permits any SNMP manager to access all objects with read-only permissions using the community string *public*. This configuration does not cause the device to send any traps.

```
Device(config)# snmp-server community public
```

This example shows how to permit any SNMP manager to access all objects with read-only permission using the community string *public*. The device also sends VTP traps to the hosts 192.180.1.111 and 192.180.1.33 using SNMPv1 and to the host 192.180.1.27 using SNMPv2C. The community string *public* is sent with the traps.

```
Device(config)# snmp-server community public
Device(config)# snmp-server enable traps vtp
Device(config)# snmp-server host 192.180.1.27 version 2c public
Device(config)# snmp-server host 192.180.1.111 version 1 public
Device(config)# snmp-server host 192.180.1.33 public
```

This example shows how to allow read-only access for all objects to members of access list 4 that use the *comaccess* community string. No other SNMP managers have access to any objects. SNMP Authentication Failure traps are sent by SNMPv2C to the host *cisco.com* using the community string *public*.

```
Device(config)# snmp-server community comaccess ro 4
Device(config)# snmp-server enable traps snmp authentication
Device(config)# snmp-server host cisco.com version 2c public
```

This example shows how to send Entity MIB traps to the host *cisco.com*. The community string is restricted. The first line enables the device to send Entity MIB traps in addition to any traps previously enabled. The

second line specifies the destination of these traps and overwrites any previous **snmp-server** host commands for the host *cisco.com*.

```
Device(config)# snmp-server enable traps entity
Device(config)# snmp-server host cisco.com restricted entity
```

This example shows how to enable the device to send all traps to the host *myhost.cisco.com* using the community string *public*:

```
Device(config)# snmp-server enable traps
Device(config)# snmp-server host myhost.cisco.com public
```

This example shows how to associate a user with a remote host and to send **auth** (authNoPriv) authentication-level informs when the user enters global configuration mode:

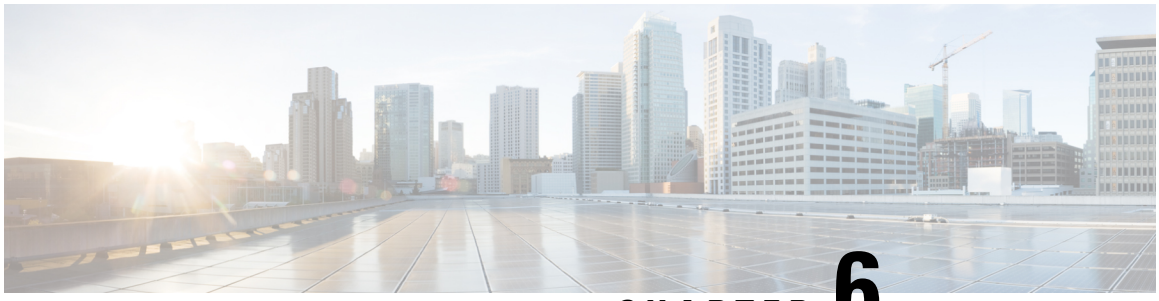
```
Device(config)# snmp-server engineID remote 192.180.1.27 00000063000100a1c0b4011b
Device(config)# snmp-server group authgroup v3 auth
Device(config)# snmp-server user authuser authgroup remote 192.180.1.27 v3 auth md5 mypassword
Device(config)# snmp-server user authuser authgroup v3 auth md5 mypassword
Device(config)# snmp-server host 192.180.1.27 informs version 3 auth authuser config
Device(config)# snmp-server enable traps
Device(config)# snmp-server inform retries 0
```

Feature History for Simple Network Management Protocol

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	Simple Network Management Protocol	This feature was introduced.



CHAPTER 6

Configuring Service Level Agreements

This chapter describes how to use Cisco IOS IP Service Level Agreements (SLAs) on the switch.

Unless otherwise noted, the term *switch* refers to a standalone switch or a switch stack.

- Restrictions on SLAs, on page 55
- Information About SLAs, on page 55
- How to Configure IP SLAs Operations, on page 60
- Monitoring IP SLA Operations, on page 73
- Monitoring IP SLA Operation Examples, on page 74
- Additional References, on page 75

Restrictions on SLAs

This section lists the restrictions on SLAs.

The following are restrictions on IP SLAs network performance measurement:

- The device does not support VoIP service levels using the gatekeeper registration delay operations measurements.
- Only a Cisco IOS device can be a source for a destination IP SLAs responder.
- You cannot configure the IP SLAs responder on non-Cisco devices and Cisco IOS IP SLAs can send operational packets only to services native to those devices.

Information About SLAs

Cisco IOS IP Service Level Agreements (SLAs)

Cisco IOS IP SLAs send data across the network to measure performance between multiple network locations or across multiple network paths. They simulate network data and IP services and collect network performance information in real time. Cisco IOS IP SLAs generate and analyze traffic either between Cisco IOS devices or from a Cisco IOS device to a remote IP device such as a network application server. Measurements provided by the various Cisco IOS IP SLA operations can be used for troubleshooting, for problem analysis, and for designing network topologies.

Depending on the specific Cisco IOS IP SLA operations, various network performance statistics are monitored within the Cisco device and stored in both command-line interface (CLI) and Simple Network Management Protocol (SNMP) MIBs. IP SLA packets have configurable IP and application layer options such as source and destination IP address, User Datagram Protocol (UDP)/TCP port numbers, a type of service (ToS) byte (including Differentiated Services Code Point [DSCP] and IP Prefix bits), Virtual Private Network (VPN) routing/forwarding instance (VRF), and URL web address.

Because Cisco IP SLAs are Layer 2 transport independent, you can configure end-to-end operations over disparate networks to best reflect the metrics that an end user is likely to experience. IP SLAs collect and analyze the following performance metrics:

- Delay (both round-trip and one-way)
- Jitter (directional)
- Packet loss (directional)
- Packet sequencing (packet ordering)
- Path (per hop)
- Connectivity (directional)
- Server or website download time

Because Cisco IOS IP SLAs is SNMP-accessible, it can also be used by performance-monitoring applications like Cisco Prime Internetwork Performance Monitor (IPM) and other third-party Cisco partner performance management products.

Using IP SLAs can provide the following benefits:

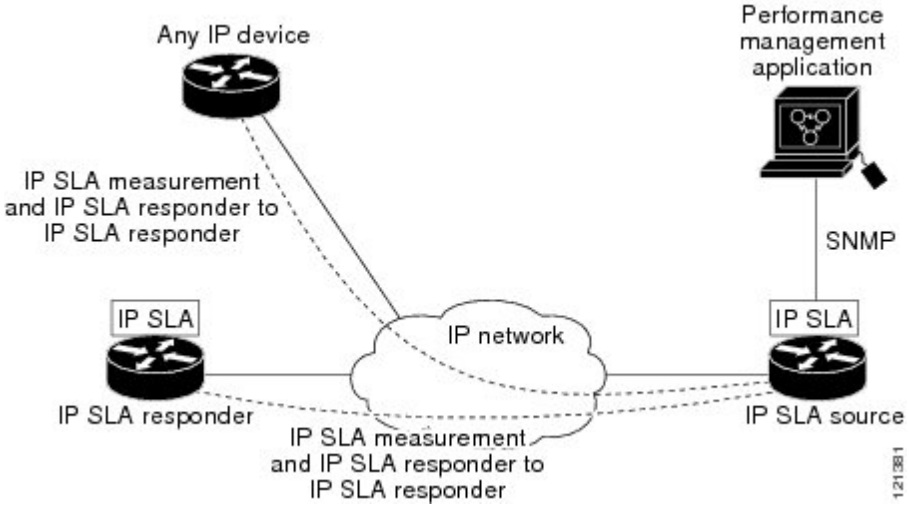
- Service-level agreement monitoring, measurement, and verification.
- Network performance monitoring
    - Measurement of jitter, latency, or packet loss in the network.
    - Continuous, reliable, and predictable measurements.
- IP service network health assessment to verify that the existing QoS is sufficient for new IP services.
- Edge-to-edge network availability monitoring for proactive verification and connectivity testing of network resources (for example, shows the network availability of an NFS server used to store business critical data from a remote site).
- Network operation troubleshooting by providing consistent, reliable measurement that immediately identifies problems and saves troubleshooting time.
- Multiprotocol Label Switching (MPLS) performance monitoring and network verification (if the device supports MPLS).

Network Performance Measurement with Cisco IOS IP SLAs

You can use IP SLAs to monitor the performance between any area in the network—core, distribution, and edge—without deploying a physical probe. It uses generated traffic to measure network performance between two networking devices.

Figure 2: Cisco IOS IP SLAs Operation

The following figure shows how IP SLAs begin when the source device sends a generated packet to the destination device. After the destination device receives the packet, depending on the type of IP SLAs operation, it responds with time-stamp information for the source to make the calculation on performance metrics. An IP SLAs operation performs a network measurement from the source device to a destination in the network using a specific protocol such as UDP.



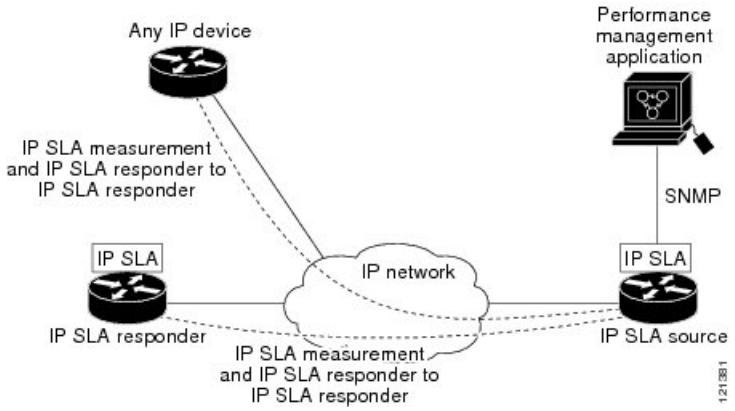
IP SLA Responder and IP SLA Control Protocol

The IP SLA responder is a component embedded in the destination Cisco device that allows the system to anticipate and respond to IP SLA request packets. The responder provides accurate measurements without the need for dedicated probes. The responder uses the Cisco IOS IP SLA Control Protocol to provide a mechanism through which it can be notified on which port it should listen and respond.

Note The IP SLA responder can be a Cisco IOS Layer 2, responder-configurable device. The responder does not need to support full IP SLA functionality.

The following figure shows where the Cisco IOS IP SLA responder fits in the IP network. The responder listens on a specific port for control protocol messages sent by an IP SLA operation. Upon receipt of the control message, it enables the specified UDP or TCP port for the specified duration. During this time, the responder accepts the requests and responds to them. It disables the port after it responds to the IP SLA packet, or when the specified time expires. MD5 authentication for control messages is available for added security.

Figure 3: Cisco IOS IP SLAs Operation



You do not need to enable the responder on the destination device for all IP SLA operations. For example, a responder is not required for services that are already provided by the destination router (such as Telnet or HTTP).

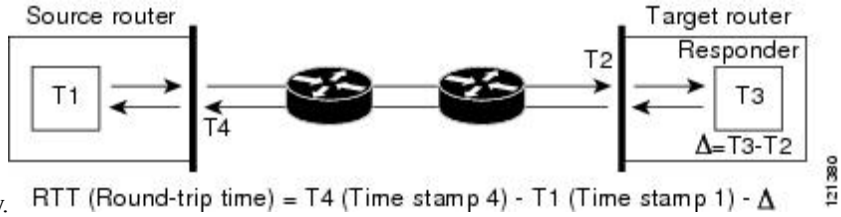
Response Time Computation for IP SLAs

Switches, controllers, and routers can take tens of milliseconds to process incoming packets due to other high priority processes. This delay affects the response times because the test-packet reply might be in a queue while waiting to be processed. In this situation, the response times would not accurately represent true network delays. IP SLAs minimize these processing delays on the source device as well as on the target device (if the responder is being used) to determine true round-trip times. IP SLA test packets use time stamping to minimize the processing delays.

When the IP SLA responder is enabled, it allows the target device to take time stamps when the packet arrives on the interface at interrupt level and again just as it is leaving, eliminating the processing time. This time stamping is made with a granularity of sub-milliseconds (ms).

Figure 4: Cisco IOS IP SLA Responder Time Stamping

The following figure demonstrates how the responder works. Four time stamps are taken to make the calculation for round-trip time. At the target router, with the responder functionality enabled, time stamp 2 (TS2) is subtracted from time stamp 3 (TS3) to produce the time spent processing the test packet as represented by delta. This delta value is then subtracted from the overall round-trip time. Notice that the same principle is applied by IP SLAs on the source router where the incoming time stamp 4 (TS4) is also taken at the interrupt level to allow for greater accuracy.



An additional benefit of the two time stamps at the target device is the ability to track one-way delay, jitter, and directional packet loss. Because much network behavior is asynchronous, it is critical to have these statistics. However, to capture one-way delay measurements, you must configure both the source router and target router with Network Time Protocol (NTP) so that the source and target are synchronized to the same clock source. One-way jitter measurements do not require clock synchronization.

IP SLAs Operation Scheduling

When you configure an IP SLAs operation, you must schedule the operation to begin capturing statistics and collecting error information. You can schedule an operation to start immediately or to start at a certain month, day, and hour. You can use the *pending* option to set the operation to start at a later time. The pending option is an internal state of the operation that is visible through SNMP. The pending state is also used when an operation is a reaction (threshold) operation waiting to be triggered. You can schedule a single IP SLAs operation or a group of operations at one time.

You can schedule several IP SLAs operations by using a single command through the Cisco IOS CLI or the CISCO RTTMON-MIB. Scheduling the operations to run at evenly distributed times allows you to control the amount of IP SLAs monitoring traffic. This distribution of IP SLA operations helps minimize the CPU utilization and thus improves network scalability.

For more details about the IP SLA multi-operations scheduling functionality, see the "IP SLAs—Multiple Operation Scheduling" chapter of the *Cisco IOS IP SLAs Configuration Guide*.

IP SLA Operation Threshold Monitoring

To support successful service level agreement monitoring, you must have mechanisms that notify you immediately of any possible violation. IP SLAs can send SNMP traps that are triggered by events such as the following:

- Connection loss
- Timeout
- Round-trip time threshold
- Average jitter threshold
- One-way packet loss
- One-way jitter
- One-way mean opinion score (MOS)
- One-way latency

An IP SLA threshold violation can also trigger another IP SLA operation for further analysis. For example, the frequency could be increased or an Internet Control Message Protocol (ICMP) path echo or ICMP path jitter operation could be initiated for troubleshooting.

ICMP Echo

The ICMP echo operation measures the end-to-end response time between a Cisco device and any other device that uses IP. The response time is computed by measuring the time it takes to send an ICMP echo request message to a destination and receive an ICMP echo reply. Many customers use IP SLA ICMP-based operations, in-house ping testing, or ping-based dedicated probes to measure this response time. The IP SLA ICMP echo operation conforms to the same specifications as ICMP ping testing, and both methods result in the same response times.

UDP Jitter

Jitter is a simple term that describes interpacket delay variance. When multiple packets are sent consecutively at an interval of 10 ms from source to destination, the destination should receive them 10 ms apart (if the network is behaving correctly). However, if there are delays in the network (such as queuing, arriving through alternate routes, and so on), the time interval between packet arrivals might be more or less than 10 ms. A positive jitter value indicates that the packets arrived more than 10 ms apart. A negative jitter value indicates that the packets arrived less than 10 ms apart. If the packets arrive 12 ms apart, the positive jitter is 2 ms; if the packets arrive 8 ms apart, the negative jitter is 2 ms. For delay-sensitive networks, positive jitter values are undesirable, and a jitter value of 0 is ideal.

In addition to monitoring jitter, the IP SLA UDP jitter operation can be used as a multipurpose data gathering operation. The packets generated by IP SLAs carry sequence information and time stamps from the source and operational target that include packet sending and receiving data. Based on this data, UDP jitter operations measure the following:

- Per-direction jitter (source to destination and destination to source)
- Per-direction packet-loss
- Per-direction delay (one-way delay)
- Round-trip delay (average round-trip time)

Because the paths for the sending and receiving of data can be different (asymmetric), you can use the per-direction data to more readily identify where congestion or other problems are occurring in the network.

The UDP jitter operation generates synthetic (simulated) UDP traffic and sends a number of UDP packets, each of a specified size, sent a specified number of milliseconds apart, from a source router to a target router, at a given frequency. By default, ten packet-frames, each with a payload size of 10 bytes are generated every 10 ms, and the operation is repeated every 60 seconds. You can configure each of these parameters to best simulate the IP service you want to provide.

To provide accurate one-way delay (latency) measurements, time synchronization (as provided by NTP) is required between the source and the target device. Time synchronization is not required for the one-way jitter and packet loss measurements. If the time is not synchronized between the source and target devices, one-way jitter and packet loss data is returned, but values of 0 are returned for the one-way delay measurements provided by the UDP jitter operation.

How to Configure IP SLAs Operations

This section does not include configuration information for all available operations as the configuration information details are included in the *Cisco IOS IP SLAs Configuration Guide*. It does include several operations as examples, including configuring the responder, configuring a UDP jitter operation, which requires a responder, and configuring an ICMP echo operation, which does not require a responder. For details about configuring other operations, see the *Cisco IOS IP SLAs Configuration Guide*.

Default Configuration

No IP SLAs operations are configured.

Configuration Guidelines

For information on the IP SLA commands, see the *Cisco IOS IP SLAs Command Reference, Release 12.4T* command reference.

For detailed descriptions and configuration procedures, see the *Cisco IOS IP SLAs Configuration Guide, Release 12.4TL*.

Not all of the IP SLA commands or operations described in the referenced guide are supported on the device. The device supports IP service level analysis by using UDP jitter, UDP echo, HTTP, TCP connect, ICMP echo, ICMP path echo, ICMP path jitter, FTP, DNS, and DHCP, as well as multiple operation scheduling and proactive threshold monitoring. It does not support VoIP service levels using the gatekeeper registration delay operations measurements.

Before configuring any IP SLAs application, you can use the **show ip sla application** privileged EXEC command to verify that the operation type is supported on your software image. This is an example of the output from the command:

```
Device# show ip sla application

        IP Service Level Agreements
Version: Round Trip Time MIB 2.2.0, Infrastructure Engine-III

Supported Operation Types:
        icmpEcho, path-echo, path-jitter, udpEcho, tcpConnect, http
        dns, udpJitter, dhcp, ftp, udpApp, wspApp

Supported Features:
        IPSLAs Event Publisher

IP SLAs low memory water mark: 33299323
Estimated system max number of entries: 24389

Estimated number of configurable operations: 24389
Number of Entries configured  : 0
Number of active Entries      : 0
Number of pending Entries     : 0
Number of inactive Entries    : 0
Time of last change in whole IP SLAs: *13:04:37.668 UTC Wed Dec 19 2012
```

Configuring the IP SLA Responder

The IP SLA responder is available only on Cisco IOS software-based devices, including some Layer 2 devices that do not support full IP SLA functionality.

Follow these steps to configure the IP SLA responder on the target device (the operational target):

Procedure

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

	Command or Action	Purpose
	`Device> enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip sla responder** {**tcp-connect** | **udp-echo**} **ipaddress** *ip-address* **port** *port-number* **Example:** `Device(config)# ip sla responder udp-echo 172.29.139.134 5000`	Configures the device as an IP SLA responder. The keywords have these meanings: • **tcp-connect**—Enables the responder for TCP connect operations. • **udp-echo**—Enables the responder for User Datagram Protocol (UDP) echo or jitter operations. • **ipaddress** *ip-address*—Enter the destination IP address. • **port** *port-number*—Enter the destination port number. **Note** The IP address and port number must match those configured on the source device for the IP SLA operation.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Implementing IP SLA Network Performance Measurement

Follow these steps to implement IP SLA network performance measurement on your device:

Before you begin

Use the **show ip sla application** privileged EXEC command to verify that the desired operation type is supported on your software image.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip sla** *operation-number* **Example:** `Device(config)# ip sla 10`	Creates an IP SLA operation, and enters IP SLA configuration mode.
Step 4	**udp-jitter** {*destination-ip-address* | *destination-hostname*} *destination-port* [**source-ip** {*ip-address* | *hostname*}] [**source-port** *port-number*] [**control** {**enable** | **disable**}] [**num-packets** *number-of-packets*] [**interval** *interpacket-interval*] **Example:** `Device(config-ip-sla)# udp-jitter 172.29.139.134 5000`	Configures the IP SLA operation as the operation type of your choice (a UDP jitter operation is used in the example), and enters its configuration mode (UDP jitter configuration mode is used in the example). • *destination-ip-address* | *destination-hostname*—Specifies the destination IP address or hostname. • *destination-port*—Specifies the destination port number in the range from 1 to 65535. • (Optional) **source-ip** {*ip-address* | *hostname*}—Specifies the source IP address or hostname. When a source IP address or hostname is not specified, IP SLA chooses the IP address nearest to the destination

	Command or Action	Purpose
		• (Optional) **source-port** *port-number*—Specifies the source port number in the range from 1 to 65535. When a port number is not specified, IP SLA chooses an available port. • (Optional) **control**—Enables or disables sending of IP SLA control messages to the IP SLA responder. By default, IP SLA control messages are sent to the destination device to establish a connection with the IP SLA responder • (Optional) **num-packets** *number-of-packets*—Enters the number of packets to be generated. The range is 1 to 6000; the default is 10. • (Optional) **interval** *inter-packet-interval*—Enters the interval between sending packets in milliseconds. The range is 1 to 6000; the default value is 20 ms.
Step 5	**frequency** *seconds* **Example:** `Device(config-ip-sla-jitter)# frequency 45`	(Optional) Configures options for the SLA operation. This example sets the rate at which a specified IP SLA operation repeats. The range is from 1 to 604800 seconds; the default is 60 seconds.
Step 6	**threshold** *milliseconds* **Example:** `Device(config-ip-sla-jitter)# threshold 200`	(Optional) Configures threshold conditions. This example sets the threshold of the specified IP SLA operation to 200. The range is from 0 to 60000 milliseconds.
Step 7	**exit** **Example:** `Device(config-ip-sla-jitter)# exit`	Exits the SLA operation configuration mode (UDP jitter configuration mode in this example), and returns to global configuration mode.
Step 8	**ip sla schedule** *operation-number* [**life** {**forever** | *seconds*}] [**start-time** {*hh:mm* [*:ss*] [*month day* | *day month*] | **pending** | **now** | **after** *hh:mm:ss*] [**ageout** *seconds*] [**recurring**] **Example:**	Configures the scheduling parameters for an individual IP SLA operation. • *operation-number*—Enter the RTR entry number. • (Optional) **life**—Sets the operation to run indefinitely (**forever**) or for a specific

	Command or Action	Purpose
	`Device(config)# ip sla schedule 10 start-time now life forever`	number of *seconds*. The range is from 0 to 2147483647. The default is 3600 seconds (1 hour). • (Optional) **start-time**—Enters the time for the operation to begin collecting information: To start at a specific time, enter the hour, minute, second (in 24-hour notation), and day of the month. If no month is entered, the default is the current month. Enter **pending** to select no information collection until a start time is selected. Enter **now** to start the operation immediately. Enter **after** *hh:mm:ss* to show that the operation should start after the entered time has elapsed. • (Optional) **ageout** *seconds*—Enter the number of seconds to keep the operation in memory when it is not actively collecting information. The range is 0 to 2073600 seconds, the default is 0 seconds (never ages out). • (Optional) **recurring**—Set the operation to automatically run every day.
Step 9	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 10	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

UDP Jitter Configuration

This example shows how to configure a UDP jitter IP SLA operation:

```
Device(config)# ip sla 10
Device(config-ip-sla)# udp-jitter 172.29.139.134 5000
Device(config-ip-sla-jitter)# frequency 30
Device(config-ip-sla-jitter)# exit
Device(config)# ip sla schedule 5 start-time now life forever
Device(config)# end
Device# show ip sla configuration 10
IP SLAs, Infrastructure Engine-II.

Entry number: 10
Owner:
Tag:
Type of operation to perform: udp-jitter
Target address/Source address: 10.0.0.10/10.0.0.1
Target port/Source port: 2/0
Request size (ARR data portion): 32
Operation timeout (milliseconds): 5000
Packet Interval (milliseconds)/Number of packets: 20/10
Type Of Service parameters: 0x0
Verify data: No
Vrf Name:
Control Packets: enabled
Schedule:
    Operation frequency (seconds): 30
    Next Scheduled Start Time: Pending trigger
    Group Scheduled : FALSE
    Randomly Scheduled : FALSE
    Life (seconds): 3600
    Entry Ageout (seconds): never
    Recurring (Starting Everyday): FALSE
    Status of entry (SNMP RowStatus): notInService
Threshold (milliseconds): 5000
Distribution Statistics:
    Number of statistic hours kept: 2
    Number of statistic distribution buckets kept: 1
    Statistic distribution interval (milliseconds): 20
Enhanced History:
```

Analyzing IP Service Levels by Using the UDP Jitter Operation

Follow these steps to configure a UDP jitter operation on the source device:

Before you begin

You must enable the IP SLA responder on the target device (the operational target) to configure a UDP jitter operation on the source device.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip sla** *operation-number* **Example:** `Device(config)# ip sla 10`	Creates an IP SLA operation, and enters IP SLA configuration mode.
Step 4	**udp-jitter** {*destination-ip-address* | *destination-hostname*} *destination-port* [**source-ip** {*ip-address* | *hostname*}] [**source-port** *port-number*] [**control** {**enable** | **disable**}] [**num-packets** *number-of-packets*] [**interval** *interpacket-interval*] **Example:** `Device(config-ip-sla)# udp-jitter 172.29.139.134 5000 source-ip 172.29.139.140 source-port 4000`	Configures the IP SLA operation as a UDP jitter operation, and enters UDP jitter configuration mode. • *destination-ip-address* | *destination-hostname*: Specifies the destination IP address or hostname. • *destination-port*: Specifies the destination port number in the range from 1 to 65535. • (Optional) **source-ip** {*ip-address* | *hostname*}: Specifies the source IP address or hostname. When a source IP address or hostname is not specified, IP SLA chooses the IP address nearest to the destination. • (Optional) **source-port** *port-number*: Specifies the source port number in the range from 1 to 65535. When a port number is not specified, IP SLA chooses an available port. **Note** If the **udp-jitter** command does not have the source port configured, UDP chooses any random port for control packets. In case UDP chooses the reserved port 1967, it may result in high CPU utilisation by the IP SLA responder.

	Command or Action	Purpose
		• (Optional) **control**: Enables or disables sending of IP SLA control messages to the IP SLA responder. By default, IP SLA control messages are sent to the destination device to establish a connection with the IP SLA responder. • (Optional) **num-packets** *number-of-packets*: Enters the number of packets to be generated. The range is 1 to 6000; the default is 10. • (Optional) **interval** *inter-packet-interval*: Enters the interval between sending packets in milliseconds. The range is 1 to 6000; the default value is 20 ms.
Step 5	**frequency** *seconds* **Example:** `Device(config-ip-sla-jitter)# frequency 45`	(Optional) Sets the rate at which a specified IP SLA operation repeats. The range is from 1 to 604800 seconds; the default is 60 seconds.
Step 6	**exit** **Example:** `Device(config-ip-sla-jitter)# exit`	Exits UDP jitter configuration mode, and returns to global configuration mode.
Step 7	**ip sla schedule** *operation-number* [**life** {**forever** | *seconds*}] [**start-time** {*hh:mm* [*:ss*] [*month day* | *day month*] | **pending** | **now** | **after** *hh:mm:ss*] [**ageout** *seconds*] [**recurring**] **Example:** `Device(config)# ip sla schedule 10 start-time now life forever`	Configures the scheduling parameters for an individual IP SLA operation. • *operation-number*: Enter the RTR entry number. • (Optional) **life**: Sets the operation to run indefinitely (**forever**) or for a specific number of *seconds*. The range is from 0 to 2147483647. The default is 3600 seconds (1 hour). • (Optional) **start-time**: Enters the time for the operation to begin collecting information: To start at a specific time, enter the hour, minute, second (in 24-hour notation), and day of the month. If no month is entered, the default is the current month.

	Command or Action	Purpose
		Enter **pending** to select no information collection until a start time is selected. Enter **now** to start the operation immediately. Enter **after** *hh:mm:ss* to show that the operation should start after the entered time has elapsed. • (Optional) **ageout** *seconds*: Enter the number of seconds to keep the operation in memory when it is not actively collecting information. The range is 0 to 2073600 seconds, the default is 0 seconds (never ages out). • (Optional) **recurring**: Set the operation to automatically run every day.
Step 8	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 9	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 10	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring a UDP Jitter IP SLA Operation

This example shows how to configure a UDP jitter IP SLA operation:

```
Device(config)# ip sla 10
Device(config-ip-sla)# udp-jitter 172.29.139.134 5000 source-ip 172.29.139.140 source-port
 4000
Device(config-ip-sla-jitter)# frequency 30
Device(config-ip-sla-jitter)# exit
Device(config)# ip sla schedule 10 start-time now life forever
Device(config)# end
Device# show ip sla configuration 10
```

```
IP SLAs, Infrastructure Engine-II.

Entry number: 10
Owner:
Tag:
Type of operation to perform: udp-jitter
Target address/Source address: 10.0.0.10/10.0.0.1
Target port/Source port: 2/0
Request size (ARR data portion): 32
Operation timeout (milliseconds): 5000
Packet Interval (milliseconds)/Number of packets: 20/10
Type Of Service parameters: 0x0
Verify data: No
Vrf Name:
Control Packets: enabled
Schedule:
    Operation frequency (seconds): 30
    Next Scheduled Start Time: Pending trigger
    Group Scheduled : FALSE
    Randomly Scheduled : FALSE
    Life (seconds): 3600
    Entry Ageout (seconds): never
    Recurring (Starting Everyday): FALSE
    Status of entry (SNMP RowStatus): notInService
Threshold (milliseconds): 5000
Distribution Statistics:
    Number of statistic hours kept: 2
    Number of statistic distribution buckets kept: 1
    Statistic distribution interval (milliseconds): 20
Enhanced History:
```

Analyzing IP Service Levels by Using the ICMP Echo Operation

Follow these steps to configure an ICMP echo operation on the source device:

Before you begin

This operation does not require the IP SLA responder to be enabled.

Procedure

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

	Command or Action	Purpose
Step 3	**ip sla** *operation-number* **Example:** `Device(config)# ip sla 10`	Creates an IP SLA operation and enters IP SLA configuration mode.
Step 4	**icmp-echo** {*destination-ip-address* | *destination-hostname*} [**source-ip** {*ip-address* | *hostname*} | **source-interface** *interface-id*] **Example:** `Device(config-ip-sla)# icmp-echo 172.29.139.134`	Configures the IP SLA operation as an ICMP Echo operation and enters ICMP echo configuration mode. • *destination-ip-address* | *destination-hostname*—Specifies the destination IP address or hostname. • (Optional) **source-ip** {*ip-address* | *hostname*}—Specifies the source IP address or hostname. When a source IP address or hostname is not specified, IP SLA chooses the IP address nearest to the destination. • (Optional) **source-interface** *interface-id*—Specifies the source interface for the operation.
Step 5	**frequency** *seconds* **Example:** `Device(config-ip-sla-echo)# frequency 30`	(Optional) Sets the rate at which a specified IP SLA operation repeats. The range is from 1 to 604800 seconds; the default is 60 seconds.
Step 6	**exit** **Example:** `Device(config-ip-sla-echo)# exit`	Exits UDP echo configuration mode, and returns to global configuration mode.
Step 7	**ip sla schedule** *operation-number* [**life** {**forever** | *seconds*}] [**start-time** {*hh:mm* [*:ss*] [*month day* | *day month*] | **pending** | **now** | **after** *hh:mm:ss*] [**ageout** *seconds*] [**recurring**] **Example:** `Device(config)# ip sla schedule 5 start-time now life forever`	Configures the scheduling parameters for an individual IP SLA operation. • *operation-number*—Enter the RTR entry number. • (Optional) **life**—Sets the operation to run indefinitely (**forever**) or for a specific number of *seconds*. The range is from 0 to 2147483647. The default is 3600 seconds (1 hour)

	Command or Action	Purpose
		• (Optional) **start-time**—Enter the time for the operation to begin collecting information: To start at a specific time, enter the hour, minute, second (in 24-hour notation), and day of the month. If no month is entered, the default is the current month. Enter **pending** to select no information collection until a start time is selected. Enter **now** to start the operation immediately. Enter **after** *hh:mm:ss* to indicate that the operation should start after the entered time has elapsed. • (Optional) **ageout** *seconds*—Enter the number of seconds to keep the operation in memory when it is not actively collecting information. The range is 0 to 2073600 seconds; the default is 0 seconds (never ages out). • (Optional) **recurring**—Sets the operation to automatically run every day.
Step 8	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 9	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 10	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring an ICMP Echo IP SLA Operation

This example shows how to configure an ICMP echo IP SLA operation:

```
Device(config)# ip sla 12
Device(config-ip-sla)# icmp-echo 172.29.139.134
Device(config-ip-sla-echo)# frequency 30
Device(config-ip-sla-echo)# exit
Device(config)# ip sla schedule 5 start-time now life forever
Device(config)# end
Device# show ip sla configuration 22
IP SLAs, Infrastructure Engine-II.

Entry number: 12
Owner:
Tag:
Type of operation to perform: echo
Target address: 2.2.2.2
Source address: 0.0.0.0
Request size (ARR data portion): 28
Operation timeout (milliseconds): 5000
Type Of Service parameters: 0x0
Verify data: No
Vrf Name:
Schedule:
    Operation frequency (seconds): 60
    Next Scheduled Start Time: Pending trigger
    Group Scheduled : FALSE
    Randomly Scheduled : FALSE
    Life (seconds): 3600
    Entry Ageout (seconds): never
    Recurring (Starting Everyday): FALSE
    Status of entry (SNMP RowStatus): notInService
Threshold (milliseconds): 5000
Distribution Statistics:
    Number of statistic hours kept: 2
    Number of statistic distribution buckets kept: 1
    Statistic distribution interval (milliseconds): 20
History Statistics:
    Number of history Lives kept: 0
    Number of history Buckets kept: 15
    History Filter Type: None
Enhanced History:
```

Monitoring IP SLA Operations

The following table describes the commands used to display IP SLA operation configurations and results:

Table 6: Monitoring IP SLA Operations

show ip sla application	Displays global informatio
show ip sla authentication	Displays IP SLA authentic
show ip sla configuration [*entry-number*]	Displays configuration val operations or a specific op
show ip sla enhanced-history {**collection-statistics** | **distribution statistics**} [*entry-number*]	Displays enhanced history or distribution statistics fo operation.

show ip sla ethernet-monitor configuration [*entry-number*]	Displays IP SLA automatic E
show ip sla group schedule [*schedule-entry-number*]	Displays IP SLA group sched
show ip sla history [*entry-number* | **full** | **tabular**]	Displays history collected for
show ip sla mpls-lsp-monitor {**collection-statistics** | **configuration** | **ldp operational-state** | **scan-queue** | **summary** [*entry-number*] | **neighbors**}	Displays MPLS label switche operations.
show ip sla reaction-configuration [*entry-number*]	Displays the configured proact all IP SLA operations or a spe
show ip sla reaction-trigger [*entry-number*]	Displays the reaction trigger i or a specific operation.
show ip sla responder	Displays information about th
show ip sla statistics [*entry-number* | **aggregated** | **details**]	Displays current or aggregate

Monitoring IP SLA Operation Examples

The following example shows all IP SLAs by application:

```
Device# show ip sla application

        IP Service Level Agreements
Version: Round Trip Time MIB 2.2.0, Infrastructure Engine-III

Supported Operation Types:
        icmpEcho, path-echo, path-jitter, udpEcho, tcpConnect, http
        dns, udpJitter, dhcp, ftp, udpApp, wspApp

Supported Features:
        IPSLAs Event Publisher

IP SLAs low memory water mark: 33299323
Estimated system max number of entries: 24389

Estimated number of configurable operations: 24389
Number of Entries configured  : 0
Number of active Entries      : 0
Number of pending Entries     : 0
Number of inactive Entries    : 0
Time of last change in whole IP SLAs: *13:04:37.668 UTC Wed Dec 19 2012
```

The following example shows all IP SLA distribution statistics:

```
Device# show ip sla enhanced-history distribution-statistics

Point by point Enhanced History
Entry    = Entry Number
Int      = Aggregation Interval
BucI     = Bucket Index
StartT   = Aggregation Start Time
Pth      = Path index
Hop      = Hop in path index
Comps    = Operations completed
OvrTh    = Operations completed over thresholds
```

```
SumCmp   = Sum of RTT (milliseconds)
SumCmp2L = Sum of RTT squared low 32 bits (milliseconds)
SumCmp2H = Sum of RTT squared high 32 bits (milliseconds)
TMax     = RTT maximum (milliseconds)
TMin     = RTT minimum (milliseconds)

Entry Int BucI StartT     Pth Hop Comps OvrTh SumCmp     SumCmp2L   SumCmp2H   T
Max    TMin
```

Additional References

Related Documents

Related Topic	Document Title
Cisco Medianet Metadata Guide	http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mdata/configuration/15-sy/mdata-15sy-book/metadata-framework.pdf
Cisco Media Services Proxy Configuration Guide	http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/msp/configuration/15-mt/msp-15-mt-book.pdf
Cisco Mediatrace and Cisco Performance Monitor Configuration Guide	http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/media_monitoring/configuration/15-mt/mm-15-mt-book/mm-mediatrace.html

Error Message Decoder

Description	Link
To help you research and resolve system error messages in this release, use the Error Message Decoder tool.	https://www.cisco.com/cgi-bin/Support/Errordecoder/index.cgi

Standards and RFCs

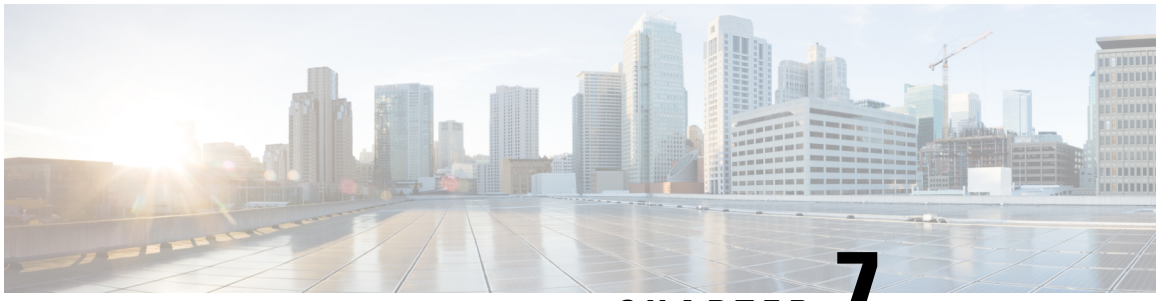
Standard/RFC	Title
None	-

MIBs

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

Technical Assistance

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



CHAPTER 7

Configuring SPAN and RSPAN

- Prerequisites for SPAN and RSPAN, on page 77
- Restrictions for SPAN and RSPAN, on page 77
- Information About SPAN and RSPAN, on page 79
- How to Configure SPAN and RSPAN, on page 89
- Monitoring SPAN and RSPAN Operations, on page 111
- SPAN and RSPAN Configuration Examples, on page 111
- Feature History for SPAN and RSPAN, on page 113

Prerequisites for SPAN and RSPAN

SPAN

- You can limit SPAN traffic to specific VLANs by using the **filter vlan** keyword. If a trunk port is being monitored, only traffic on the VLANs specified with this keyword is monitored. By default, all VLANs are monitored on a trunk port.

RSPAN

- We recommend that you configure an RSPAN VLAN before you configure an RSPAN source or a destination session.

Restrictions for SPAN and RSPAN

SPAN

The restrictions for SPAN are as follows:

- On each device, you can configure 66 sessions. A maximum of 8 source sessions can be configured and the remaining sessions can be configured as RSPAN destinations sessions. A source session is either a local SPAN session or an RSPAN source session.
- For SPAN sources, you can monitor traffic for a single port or VLAN or a series or range of ports or VLANs for each session. You cannot mix source ports and source VLANs within a single SPAN session.

- The destination port cannot be a source port; a source port cannot be a destination port.
- You cannot have two SPAN sessions using the same destination port.
- When you configure a device port as a SPAN destination port, it is no longer a normal device port; only monitored traffic passes through the SPAN destination port.
- Entering SPAN configuration commands does not remove previously configured SPAN parameters. You must enter the **no monitor session** {*session_number* | **all** | **local** | **remote**} global configuration command to delete configured SPAN parameters.
- For local SPAN, outgoing packets through the SPAN destination port carry the original encapsulation headers—untagged, ISL, or IEEE 802.1Q—if the **encapsulation replicate** keywords are specified. If the keywords are not specified, the packets are sent in native form.
- You can configure a disabled port to be a source or destination port, but the SPAN function does not start until the destination port and at least one source port or source VLAN are enabled.
- You cannot mix source VLANs and filter VLANs within a single SPAN session.

Traffic monitoring in a SPAN session has the following restrictions:

- Sources can be ports or VLANs, but you cannot mix source ports and source VLANs in the same session.
- Wireshark does not capture egress packets when egress span is active.
- You can run both a local SPAN and an RSPAN source session in the same device or device stack. The device or device stack supports a total of 66 source and RSPAN destination sessions.
- You can configure two separate SPAN or RSPAN source sessions with separate or overlapping sets of SPAN source ports and VLANs. Both switched and routed ports can be configured as SPAN sources and destinations.
- You can have multiple destination ports in a SPAN session, but no more than 64 destination ports per device stack.
- SPAN sessions do not interfere with the normal operation of the device. However, an oversubscribed SPAN destination, for example, a 10-Mb/s port monitoring a 100-Mb/s port, can result in dropped or lost packets.
- When SPAN or RSPAN is enabled, each packet being monitored is sent twice, once as normal traffic and once as a monitored packet. Monitoring a large number of ports or VLANs could potentially generate large amounts of network traffic.
- You can configure SPAN sessions on disabled ports; however, a SPAN session does not become active unless you enable the destination port and at least one source port or VLAN for that session.
- The device does not support a combination of local SPAN and RSPAN in a single session.
  - An RSPAN source session cannot have a local destination port.
  - An RSPAN destination session cannot have a local source port.
  - An RSPAN destination session and an RSPAN source session that are using the same RSPAN VLAN cannot run on the same device or device stack.
- SPAN sessions capture only Dynamic Host Configuration Protocol (DHCP) ingress packets when DHCP snooping is enabled on the device.

RSPAN

The restrictions for RSPAN are as follows:

- RSPAN does not support BPDU packet monitoring or other Layer 2 device protocols.
- The RSPAN VLAN is configured only on trunk ports and not on access ports. To avoid unwanted traffic in RSPAN VLANs, make sure that the VLAN remote-span feature is supported in all the participating devices.
- RSPAN VLANs are included as sources for port-based RSPAN sessions when source trunk ports have active RSPAN VLANs. RSPAN VLANs can also be sources in SPAN sessions. However, since the device does not monitor spanned traffic, it does not support egress spanning of packets on any RSPAN VLAN identified as the destination of an RSPAN source session on the device.
- If you enable VTP and VTP pruning, RSPAN traffic is pruned in the trunks to prevent the unwanted flooding of RSPAN traffic across the network for VLAN IDs that are lower than 1005.
- It is recommended not to configure RSPAN VLAN as Native VLAN.
- RSPAN sessions do not capture DHCP-inject packets. DHCP-inject packets refer to DHCP packets (DISCOVER, OFFER, REQUEST, and ACK packets) which are modified by the CPU and inserted back into the network.

Information About SPAN and RSPAN

SPAN and RSPAN

You can analyze network traffic passing through ports or VLANs by using SPAN or RSPAN to send a copy of the traffic to another port on the device or on another device that has been connected to a network analyzer or other monitoring or security device. SPAN copies (or mirrors) traffic received or sent (or both) on source ports or source VLANs to a destination port for analysis. SPAN does not affect the switching of network traffic on the source ports or VLANs. You must dedicate the destination port for SPAN use. Except for traffic that is required for the SPAN or RSPAN session, destination ports do not receive or forward traffic.

Only traffic that enters or leaves source ports or traffic that enters or leaves source VLANs can be monitored by using SPAN; traffic routed to a source VLAN cannot be monitored. For example, if incoming traffic is being monitored, traffic that gets routed from another VLAN to the source VLAN cannot be monitored; however, traffic that is received on the source VLAN and routed to another VLAN can be monitored.

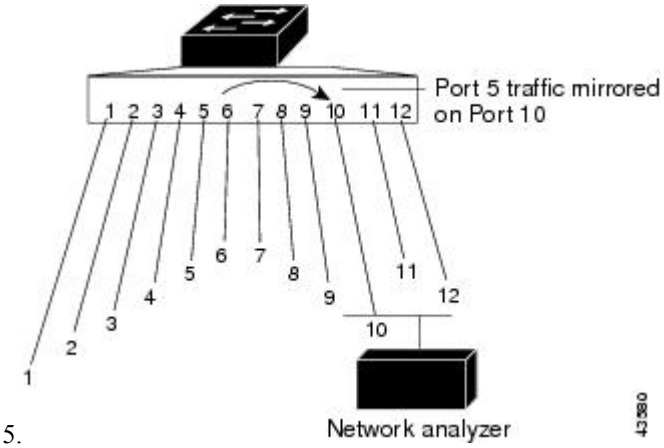
You can use the SPAN or RSPAN destination port to inject traffic from a network security device. For example, if you connect a Cisco Intrusion Detection System (IDS) sensor appliance to a destination port, the IDS device can send TCP reset packets to close down the TCP session of a suspected attacker.

Local SPAN

Local SPAN supports a SPAN session entirely within one device; all source ports or source VLANs and destination ports are in the same device or device stack. Local SPAN copies traffic from one or more source ports in any VLAN or from one or more VLANs to a destination port for analysis.

Figure 5: Example of Local SPAN Configuration on a Single Device

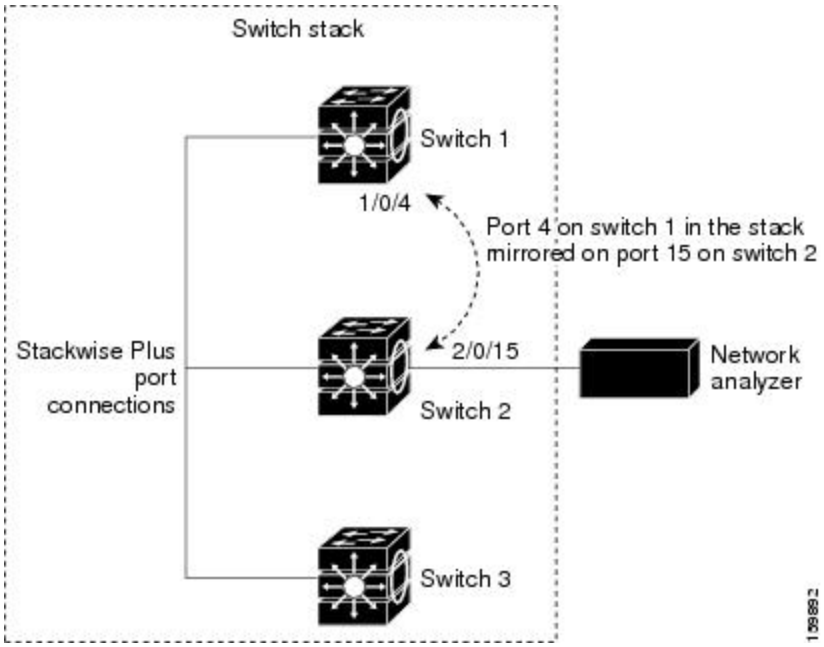
All traffic on port 5 (the source port) is mirrored to port 10 (the destination port). A network analyzer on port 10 receives all network traffic from port 5 without being physically attached to port



5.

Figure 6: Example of Local SPAN Configuration on a Device Stack

This is an example of a local SPAN in a device stack, where the source and destination ports reside on different stack members.



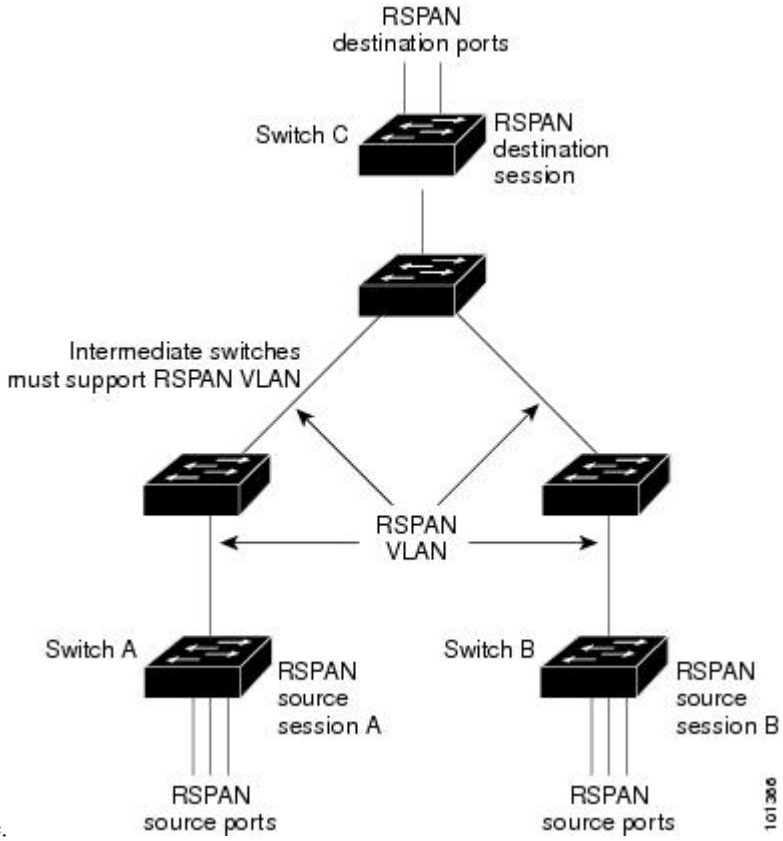
Remote SPAN

RSPAN supports source ports, source VLANs, and destination ports on different devices (or different device stacks), enabling remote monitoring of multiple devices across your network.

Figure 7: Example of RSPAN Configuration

The figure below shows source ports on Device A and Device B. The traffic for each RSPAN session is carried over a user-specified RSPAN VLAN that is dedicated for that RSPAN session in all participating devices.

The RSPAN traffic from the source ports or VLANs is copied into the RSPAN VLAN and forwarded over trunk ports carrying the RSPAN VLAN to a destination session monitoring the RSPAN VLAN. Each RSPAN source device must have either ports or VLANs as RSPAN sources. The destination is always a physical port, as shown on Device C in the figure.



SPAN and RSPAN Concepts and Terminology

SPAN Sessions

SPAN sessions (local or remote) allow you to monitor traffic on one or more ports, or one or more VLANs, and send the monitored traffic to one or more destination ports.

A local SPAN session is an association of a destination port with source ports or source VLANs, all on a single network device. Local SPAN does not have separate source and destination sessions. Local SPAN sessions gather a set of ingress and egress packets specified by the user and form them into a stream of SPAN data, which is directed to the destination port.

RSPAN consists of at least one RSPAN source session, an RSPAN VLAN, and at least one RSPAN destination session. You separately configure RSPAN source sessions and RSPAN destination sessions on different network devices. To configure an RSPAN source session on a device, you associate a set of source ports or source VLANs with an RSPAN VLAN. The output of this session is the stream of SPAN packets that are sent to the RSPAN VLAN. To configure an RSPAN destination session on another device, you associate the destination port with the RSPAN VLAN. The destination session collects all RSPAN VLAN traffic and sends it out the RSPAN destination port.

An RSPAN source session is very similar to a local SPAN session, except for where the packet stream is directed. In an RSPAN source session, SPAN packets are relabeled with the RSPAN VLAN ID and directed over normal trunk ports to the destination device.

An RSPAN destination session takes all packets received on the RSPAN VLAN, strips off the VLAN tagging, and presents them on the destination port. The session presents a copy of all RSPAN VLAN packets (except Layer 2 control packets) to the user for analysis.

Traffic monitoring in a SPAN session has these restrictions:

- Sources can be ports or VLANs, but you cannot mix source ports and source VLANs in the same session.
- You can run both a local SPAN and an RSPAN source session in the same device or device stack. The device or device stack supports a total of 66 source and RSPAN destination sessions.
- You can configure two separate SPAN or RSPAN source sessions with separate or overlapping sets of SPAN source ports and VLANs. Both switched and routed ports can be configured as SPAN sources and destinations.
- You can have multiple destination ports in a SPAN session, but no more than 64 destination ports per device stack.
- SPAN sessions do not interfere with the normal operation of the device. However, an oversubscribed SPAN destination, for example, a 10-Mb/s port monitoring a 100-Mb/s port, can result in dropped or lost packets.
- When SPAN or RSPAN is enabled, each packet being monitored is sent twice, once as normal traffic and once as a monitored packet. Therefore monitoring a large number of ports or VLANs could potentially generate large amounts of network traffic.
- You can configure SPAN sessions on disabled ports; however, a SPAN session does not become active unless you enable the destination port and at least one source port or VLAN for that session.
- The device does not support a combination of local SPAN and RSPAN in a single session.
  - An RSPAN source session cannot have a local destination port.
  - An RSPAN destination session cannot have a local source port.
  - An RSPAN destination session and an RSPAN source session that are using the same RSPAN VLAN cannot run on the same device or device stack.

Monitored Traffic

SPAN sessions can monitor these traffic types:

- Receive (Rx) SPAN—Receive (or ingress) SPAN monitors as much as possible all of the packets received by the source interface or VLAN before any modification or processing is performed by the device. A copy of each packet received by the source is sent to the destination port for that SPAN session.

  Packets that are modified because of routing or Quality of Service (QoS)—for example, modified Differentiated Services Code Point (DSCP)—are copied before modification.

  Features that can cause a packet to be dropped during receive processing have no effect on ingress SPAN; the destination port receives a copy of the packet even if the actual incoming packet is dropped. These features include IP standard and extended input Access Control Lists (ACLs), ingress QoS policing, VLAN ACLs, and egress QoS policing.

- Transmit (Tx) SPAN—Transmit (or egress) SPAN monitors as much as possible all of the packets sent by the source interface after all modification and processing is performed by the device. A copy of each packet sent by the source is sent to the destination port for that SPAN session. The copy is provided after the packet is modified.

Packets that are modified because of routing (for example, with modified time-to-live (TTL), MAC address, or QoS values) are duplicated (with the modifications) at the destination port.

Features that can cause a packet to be dropped during transmit processing also affect the duplicated copy for SPAN. These features include IP standard and extended output ACLs and egress QoS policing.

- Both—In a SPAN session, you can also monitor a port or VLAN for both received and sent packets. This is the default.

The default configuration for local SPAN session ports is to send all packets untagged. However, when you enter the **encapsulation replicate** keywords while configuring a destination port, these changes occur:

- Packets are sent on the destination port with the same encapsulation (untagged or IEEE 802.1Q) that they had on the source port.
- Packets of all types, including BPDU and Layer 2 protocol packets, are monitored.

Therefore, a local SPAN session with encapsulation replicate enabled can have a mixture of untagged and IEEE 802.1Q tagged packets appear on the destination port.

Device congestion can cause packets to be dropped at ingress source ports, egress source ports, or SPAN destination ports. In general, these characteristics are independent of one another. For example:

- A packet might be forwarded normally but dropped from monitoring due to an oversubscribed SPAN destination port.
- An ingress packet might be dropped from normal forwarding, but still appear on the SPAN destination port.
- An egress packet dropped because of device congestion is also dropped from egress SPAN.

In some SPAN configurations, multiple copies of the same source packet are sent to the SPAN destination port. For example, a bidirectional (both Rx and Tx) SPAN session is configured for the Rx monitor on port A and Tx monitor on port B. If a packet enters the device through port A and is switched to port B, both incoming and outgoing packets are sent to the destination port. Both packets are the same unless a Layer 3 rewrite occurs, in which case the packets are different because of the packet modification.

Source Ports

A source port (also called a monitored port) is a switched or routed port that you monitor for network traffic analysis.

In a local SPAN session or RSPAN source session, you can monitor source ports or VLANs for traffic in one or both directions.

The device supports any number of source ports (up to the maximum number of available ports on the device) and any number of source VLANs (up to the maximum number of VLANs supported).

You cannot mix ports and VLANs in a single session.

A source port has these characteristics:

- It can be monitored in multiple SPAN sessions.
- Each source port can be configured with a direction (ingress, egress, or both) to monitor.
- It can be any port type (for example, EtherChannel, Gigabit Ethernet, and so forth).

- For EtherChannel sources, you can monitor traffic for the entire EtherChannel or individually on a physical port as it participates in the port channel.
- It can be an access port, trunk port, routed port, or voice VLAN port.
- It cannot be a destination port.
- Source ports can be in the same or different VLANs.
- You can monitor multiple source ports in a single session.

Source VLANs

VLAN-based SPAN (VSPAN) is the monitoring of the network traffic in one or more VLANs. The SPAN or RSPAN source interface in VSPAN is a VLAN ID, and traffic is monitored on all the ports for that VLAN.

VSPAN has these characteristics:

- All active ports in the source VLAN are included as source ports and can be monitored in either or both directions.
- On a given port, only traffic on the monitored VLAN is sent to the destination port.
- If a destination port belongs to a source VLAN, it is excluded from the source list and is not monitored.
- If ports are added to or removed from the source VLANs, the traffic on the source VLAN received by those ports is added to or removed from the sources being monitored.
- You cannot use filter VLANs in the same session with VLAN sources.
- You can monitor only Ethernet VLANs.

VLAN Filtering

When you monitor a trunk port as a source port, by default, all VLANs active on the trunk are monitored. You can limit SPAN traffic monitoring on trunk source ports to specific VLANs by using VLAN filtering.

- VLAN filtering applies only to trunk ports or to voice VLAN ports.
- VLAN filtering applies only to port-based sessions and is not allowed in sessions with VLAN sources.
- When a VLAN filter list is specified, only those VLANs in the list are monitored on trunk ports or on voice VLAN access ports.
- SPAN traffic coming from other port types is not affected by VLAN filtering; that is, all VLANs are allowed on other ports.
- VLAN filtering affects only traffic forwarded to the destination SPAN port and does not affect the switching of normal traffic.

Destination Port

Each local SPAN session or RSPAN destination session must have a destination port (also called a monitoring port) that receives a copy of traffic from the source ports or VLANs and sends the SPAN packets to the user, usually a network analyzer.

A destination port has these characteristics:

- For a local SPAN session, the destination port must reside on the same device or device stack as the source port. For an RSPAN session, it is located on the device containing the RSPAN destination session. There is no destination port on a device or device stack running only an RSPAN source session.
- When a port is configured as a SPAN destination port, the configuration overwrites the original port configuration. When the SPAN destination configuration is removed, the port reverts to its previous configuration. If a configuration change is made to the port while it is acting as a SPAN destination port, the change does not take effect until the SPAN destination configuration had been removed.
- If the port was in an EtherChannel group, it is removed from the group while it is a destination port. If it was a routed port, it is no longer a routed port.
- It can be any Ethernet physical port.
- It cannot be a secure port.
- It cannot be a source port.
- It can be an EtherChannel group (**ON** mode only), on Cisco Catalyst 9500 Series Switches - High Performance.
- It can participate in only one SPAN session at a time (a destination port in one SPAN session cannot be a destination port for a second SPAN session).
- When it is active, incoming traffic is disabled. The port does not transmit any traffic except that required for the SPAN session. Incoming traffic is never learned or forwarded on a destination port.
- If ingress traffic forwarding is enabled for a network security device, the destination port forwards traffic at Layer 2.
- It does not participate in any of the Layer 2 protocols (STP, VTP, CDP, DTP, PagP).
- A destination port that belongs to a source VLAN of any SPAN session is excluded from the source list and is not monitored.
- The maximum number of destination ports in a device or device stack is 64.

Local SPAN and RSPAN destination ports function differently with VLAN tagging and encapsulation:

- For local SPAN, if the **encapsulation replicate** keywords are specified for the destination port, these packets appear with the original encapsulation (untagged, ISL, or IEEE 802.1Q). If these keywords are not specified, packets appear in the untagged format. Therefore, the output of a local SPAN session with **encapsulation replicate** enabled can contain a mixture of untagged, ISL, or IEEE 802.1Q-tagged packets.
- For RSPAN, the original VLAN ID is lost because it is overwritten by the RSPAN VLAN identification. Therefore, all packets appear on the destination port as untagged.

RSPAN VLAN

The RSPAN VLAN carries SPAN traffic between RSPAN source and destination sessions. RSPAN VLAN has these special characteristics:

- All traffic in the RSPAN VLAN is always flooded.
- No MAC address learning occurs on the RSPAN VLAN.
- RSPAN VLAN traffic only flows on trunk ports.

- RSPAN VLANs must be configured in VLAN configuration mode by using the **remote-span** VLAN configuration mode command.
- STP can run on RSPAN VLAN trunks but not on SPAN destination ports.
- An RSPAN VLAN cannot be a private-VLAN primary or secondary VLAN.

For VLANs 1 to 1005 that are visible to VLAN Trunking Protocol (VTP), the VLAN ID and its associated RSPAN characteristic are propagated by VTP. If you assign an RSPAN VLAN ID in the extended VLAN range (1006 to 4094), you must manually configure all intermediate devices.

It is normal to have multiple RSPAN VLANs in a network at the same time with each RSPAN VLAN defining a network-wide RSPAN session. That is, multiple RSPAN source sessions anywhere in the network can contribute packets to the RSPAN session. It is also possible to have multiple RSPAN destination sessions throughout the network, monitoring the same RSPAN VLAN and presenting traffic to the user. The RSPAN VLAN ID separates the sessions.

SPAN and RSPAN Interaction with Other Features

SPAN interacts with these features:

- Routing—SPAN does not monitor routed traffic. VSPAN only monitors traffic that enters or exits the device, not traffic that is routed between VLANs. For example, if a VLAN is being Rx-monitored and the device routes traffic from another VLAN to the monitored VLAN, that traffic is not monitored and not received on the SPAN destination port.
- STP—A destination port does not participate in STP while its SPAN or RSPAN session is active. The destination port can participate in STP after the SPAN or RSPAN session is disabled. On a source port, SPAN does not affect the STP status. STP can be active on trunk ports carrying an RSPAN VLAN.
- CDP—A SPAN destination port does not participate in CDP while the SPAN session is active. After the SPAN session is disabled, the port again participates in CDP.
- VTP—You can use VTP to prune an RSPAN VLAN between devices.
- VLAN and trunking—You can modify VLAN membership or trunk settings for source or destination ports at any time. However, changes in VLAN membership or trunk settings for a destination port do not take effect until you remove the SPAN destination configuration. Changes in VLAN membership or trunk settings for a source port immediately take effect, and the respective SPAN sessions automatically adjust accordingly.
- EtherChannel—You can configure an EtherChannel group as a source port . When a group is configured as a SPAN source, the entire group is monitored.

  If a physical port is added to a monitored EtherChannel group, the new port is added to the SPAN source port list. If a port is removed from a monitored EtherChannel group, it is automatically removed from the source port list.

  A physical port that belongs to an EtherChannel group can be configured as a SPAN source port and still be a part of the EtherChannel. In this case, data from the physical port is monitored as it participates in the EtherChannel. However, if a physical port that belongs to an EtherChannel group is configured as a SPAN destination, it is removed from the group. After the port is removed from the SPAN session, it rejoins the EtherChannel group. Ports removed from an EtherChannel group remain members of the group, but they are in the inactive or suspended state.

  If a physical port that belongs to an EtherChannel group is a destination port and the EtherChannel group is a source, the port is removed from the EtherChannel group and from the list of monitored ports.

- Multicast traffic can be monitored. For egress and ingress port monitoring, only a single unedited packet is sent to the SPAN destination port. It does not reflect the number of times the multicast packet is sent.
- A private-VLAN port cannot be a SPAN destination port.
- A secure port cannot be a SPAN destination port.

  For SPAN sessions, do not enable port security on ports with monitored egress when ingress forwarding is enabled on the destination port. For RSPAN source sessions, do not enable port security on any ports with monitored egress.
- An IEEE 802.1x port can be a SPAN source port. You can enable IEEE 802.1x on a port that is a SPAN destination port; however, IEEE 802.1x is disabled until the port is removed as a SPAN destination.

  For SPAN sessions, do not enable IEEE 802.1x on ports with monitored egress when ingress forwarding is enabled on the destination port. For RSPAN source sessions, do not enable IEEE 802.1x on any ports that are egress monitored.

SPAN and RSPAN and Device Stacks

Because the stack of devices represents one logical device, local SPAN source ports and destination ports can be in different devices in the stack. Therefore, the addition or deletion of devices in the stack can affect a local SPAN session, as well as an RSPAN source or destination session. An active session can become inactive when a device is removed from the stack or an inactive session can become active when a device is added to the stack.

Flow-Based SPAN

You can control the type of network traffic to be monitored in SPAN or RSPAN sessions by using flow-based SPAN (FSPAN) or flow-based RSPAN (FRSPAN), which apply access control lists (ACLs) to the monitored traffic on the source ports. The FSPAN ACLs can be configured to filter IPv4, IPv6, and non-IP monitored traffic.

You apply an ACL to a SPAN session through the interface. It is applied to all the traffic that is monitored on all interfaces in the SPAN session.The packets that are permitted by this ACL are copied to the SPAN destination port. No other packets are copied to the SPAN destination port.

The original traffic continues to be forwarded, and any port, VLAN, and router ACLs attached are applied. The FSPAN ACL does not have any effect on the forwarding decisions. Similarly, the port, VLAN, and router ACLs do not have any effect on the traffic monitoring. If a security input ACL denies a packet and it is not forwarded, the packet is still copied to the SPAN destination ports if the FSPAN ACL permits it. But if the security output ACL denies a packet and it is not sent, it is not copied to the SPAN destination ports. However, if the security output ACL permits the packet to go out, it is only copied to the SPAN destination ports if the FSPAN ACL permits it. This is also true for an RSPAN session.

You can attach three types of FSPAN ACLs to the SPAN session:

- IPv4 FSPAN ACL— Filters only IPv4 packets.
- IPv6 FSPAN ACL— Filters only IPv6 packets.
- MAC FSPAN ACL— Filters only non-IP packets.

If a VLAN-based FSPAN session configured on a stack cannot fit in the hardware memory on one or more devices, it is treated as unloaded on those devices, and traffic meant for the FSPAN ACL and sourcing on

that device is not copied to the SPAN destination ports. The FSPAN ACL continues to be correctly applied, and traffic is copied to the SPAN destination ports on the devices where the FSPAN ACL fits in the hardware memory.

When an empty FSPAN ACL is attached, some hardware functions copy all traffic to the SPAN destination ports for that ACL. If sufficient hardware resources are not available, even an empty FSPAN ACL can be unloaded.

Default SPAN and RSPAN Configuration

Table 7: Default SPAN and RSPAN Configuration

Feature	Default Setting
SPAN state (SPAN and RSPAN)	Disabled.
Source port traffic to monitor	Both received and sent traffic (**both**).
Encapsulation type (destination port)	Native form (untagged packets).
Ingress forwarding (destination port)	Disabled.
VLAN filtering	On a trunk interface used as a source port, all VLANs are monitored.
RSPAN VLANs	None configured.

Configuration Guidelines

SPAN Configuration Guidelines

- To remove a source or destination port or VLAN from the SPAN session, use the **no monitor session** *session_number* **source** {**interface** *interface-id* | **vlan** *vlan-id*} global configuration command or the **no monitor session** *session_number* **destination interface** *interface-id* global configuration command. For destination interfaces, the **encapsulation** options are ignored with the **no** form of the command.
- To monitor all VLANs on the trunk port, use the **no monitor session** *session_number* **filter** global configuration command.

RSPAN Configuration Guidelines

- All the SPAN configuration guidelines apply to RSPAN.
- As RSPAN VLANs have special properties, you should reserve a few VLANs across your network for use as RSPAN VLANs; do not assign access ports to these VLANs.
- You can apply an output ACL to RSPAN traffic to selectively filter or monitor specific packets. Specify these ACLs on the RSPAN VLAN in the RSPAN source devices.
- For RSPAN configuration, you can distribute the source ports and the destination ports across multiple devices in your network.

- Access ports (including voice VLAN ports) on the RSPAN VLAN are put in the inactive state.
- You can configure any VLAN as an RSPAN VLAN as long as these conditions are met:
  - The same RSPAN VLAN is used for an RSPAN session in all the devices.
  - All participating devices support RSPAN.

FSPAN and FRSPAN Configuration Guidelines

- When at least one FSPAN ACL is attached, FSPAN is enabled.
- When you attach at least one FSPAN ACL that is not empty to a SPAN session, and you have not attached one or more of the other FSPAN ACLs (for instance, you have attached an IPv4 ACL that is not empty, and have not attached IPv6 and MAC ACLs), FSPAN blocks the traffic that would have been filtered by the unattached ACLs. Therefore, this traffic is not monitored.

How to Configure SPAN and RSPAN

Creating a Local SPAN Session

Follow these steps to create a SPAN session and specify the source (monitored) ports or VLANs and the destination (monitoring) ports.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no monitor session** {*session_number* | **all** | **local** | **remote**} **Example:** `Device(config)# no monitor session all`	Removes any existing SPAN configuration for the session. • For *session_number*, the range is 1 to 66. • **all**—Removes all SPAN sessions. • **local**—Removes all local sessions. • **remote**—Removes all remote SPAN sessions.

	Command or Action	Purpose
Step 4	**monitor session** *session_number* **source** {**interface** *interface-id* | **vlan** *vlan-id*} [**,** | **-**] [**both** | **rx** | **tx**] **Example:** `Device(config)# monitor session 1 source interface gigabitethernet1/0/1`	Specifies the SPAN session and the source port (monitored port). • For *session_number*, the range is 1 to 66. • For *interface-id*, specify the source port to monitor. Valid interfaces include physical interfaces and port-channel logical interfaces (**port-channel** *port-channel-number*). Valid port-channel numbers are 1 to 48. • For *vlan-id*, specify the source VLAN to monitor. The range is 1 to 4094 (excluding the RSPAN VLAN). **Note** A single session can include multiple sources (ports or VLANs) defined in a series of commands, but you cannot combine source ports and source VLANs in one session. • (Optional) [**,** | **-**] Specifies a series or range of interfaces. Enter a space before and after the comma; enter a space before and after the hyphen. • (Optional) **both** | **rx** | **tx**—Specifies the direction of traffic to monitor. If you do not specify a traffic direction, the source interface sends both sent and received traffic. • **both**—Monitors both received and sent traffic. • **rx**—Monitors received traffic. • **tx**—Monitors sent traffic. **Note** You can use the **monitor session** *session_number* **source** command multiple times to configure multiple source ports.
Step 5	**monitor session** *session_number* **destination** {**interface** *interface-id* [**,** | **-**] [**encapsulation {replicate | dot1q}**]} **Example:**	Specifies the SPAN session and the destination port (monitoring port). The port LED remains green when the configuration changes take effect, and even after removing the configuration.

	Command or Action	Purpose
	`Device(config)# monitor session 1 destination interface gigabitethernet1/0/2 encapsulation replicate`	**Note** For local SPAN, you must use the same session number for the source and destination interfaces. • For *session_number*, specify the session number entered in step 4. • For *interface-id*, specify the destination port. The destination interface must be a physical port; it cannot be an EtherChannel, and it cannot be a VLAN. • (Optional) [**,** | -] Specifies a series or range of interfaces. Enter a space before and after the comma; enter a space before and after the hyphen. (Optional) **encapsulation replicate** specifies that the destination interface replicates the source interface encapsulation method. If not selected, the default is to send packets in native form (untagged). (Optional) **encapsulation dot1q**specifies that the destination interface accepts the source interface incoming packets with IEEE 802.1Q encapsulation. **Note** You can use **monitor session** *session_number* **destination** command multiple times to configure multiple destination ports.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config`	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`startup-config`	

Creating a Local SPAN Session and Configuring Incoming Traffic

Follow these steps to create a SPAN session, to specify the source ports or VLANs and the destination ports, and to enable incoming traffic on the destination port for a network security device (such as a Cisco IDS Sensor Appliance).

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no monitor session** {*session_number* | **all** | **local** | **remote**} **Example:** `Device(config)# no monitor session all`	Removes any existing SPAN configuration for the session. • For *session_number*, the range is 1 to 66. • **all**—Removes all SPAN sessions. • **local**—Removes all local sessions. • **remote**—Removes all remote SPAN sessions.
Step 4	**monitor session** *session_number* **source** {**interface** *interface-id* | **vlan** *vlan-id*} [**,** | **-**] [**both** | **rx** | **tx**] **Example:** `Device(config)# monitor session 2 source gigabitethernet1/0/1 rx`	Specifies the SPAN session and the source port (monitored port).
Step 5	**monitor session** *session_number* **destination** {**interface** *interface-id* [**,** | **-**] [**encapsulation replicate**] [**ingress** {**dot1q vlan** *vlan-id* | **untagged vlan** *vlan-id* | **vlan** *vlan-id*}]} **Example:**	Specifies the SPAN session, the destination port, the packet encapsulation, and the ingress VLAN and encapsulation. • For *session_number*, specify the session number entered in Step 4.

	Command or Action	Purpose

	`Device(config)# monitor session 2 destination interface gigabitethernet1/0/2 encapsulation replicate ingress dot1q vlan 6`	• For *interface-id*, specify the destination port. The destination interface must be a physical port; it cannot be an EtherChannel, and it cannot be a VLAN.
		• (Optional) [**,** | -]—Specifies a series or range of interfaces. Enter a space before and after the comma or hyphen.
		• (Optional) **encapsulation replicate** specifies that the destination interface replicates the source interface encapsulation method. If not selected, the default is to send packets in native form (untagged).
		• (Optional) **encapsulation dot1q**specifies that the destination interface accepts the source interface incoming packets with IEEE 802.1Q encapsulation.
		• **ingress** enables forwarding of incoming traffic on the destination port and to specify the encapsulation type:
		  • **dot1q vlan** *vlan-id*—Accepts incoming packets with IEEE 802.1Q encapsulation with the specified VLAN as the default VLAN.
		  • **untagged vlan** *vlan-id* or **vlan** *vlan-id*—Accepts incoming packets with untagged encapsulation type with the specified VLAN as the default VLAN.

Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device# copy running-config startup-config`	

Specifying VLANs to Filter

Follow these steps to limit SPAN source traffic to specific VLANs.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no monitor session** {*session_number* | **all** | **local** | **remote**} **Example:** `Device(config)# no monitor session all`	Removes any existing SPAN configuration for the session. • For *session_number*, the range is 1 to 66. • **all**—Removes all SPAN sessions. • **local**—Removes all local sessions. • **remote**—Removes all remote SPAN sessions.
Step 4	**monitor session** *session_number* **source interface** *interface-id* **Example:** `Device(config)# monitor session 2 source interface gigabitethernet1/0/2 rx`	Specifies the characteristics of the source port (monitored port) and SPAN session. • For *session_number*, the range is 1 to 66. • For *interface-id*, specify the source port to monitor. The interface specified must already be configured as a trunk port.
Step 5	**monitor session** *session_number* **filter vlan** *vlan-id* [**,** | **-**] **Example:**	Limits the SPAN source traffic to specific VLANs. • For *session_number*, enter the session number specified in Step 4.

	Command or Action	Purpose
	`Device(config)# monitor session 2 filter vlan 1 - 5 , 9`	• For *vlan-id*, the range is 1 to 4094. • (Optional) Use a comma (**,**) to specify a series of VLANs, or use a hyphen (-) to specify a range of VLANs. Enter a space before and after the comma; enter a space before and after the hyphen.
Step 6	**monitor session** *session_number* **destination** {**interface** *interface-id* [**,** | -] [**encapsulation replicate]**} **Example:** `Device(config)# monitor session 2 destination interface gigabitethernet1/0/1`	Specifies the SPAN session and the destination port (monitoring port). • For *session_number*, specify the session number entered in Step 4. • For *interface-id*, specify the destination port. The destination interface must be a physical port; it cannot be an EtherChannel, and it cannot be a VLAN. • (Optional) [**,** | -] Specifies a series or range of interfaces. Enter a space before and after the comma; enter a space before and after the hyphen. • (Optional) **encapsulation replicate** specifies that the destination interface replicates the source interface encapsulation method. If not selected, the default is to send packets in native form (untagged).
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 8	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 9	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring a VLAN as an RSPAN VLAN

Follow these steps to create a new VLAN, then configure it to be the RSPAN VLAN for the RSPAN session.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vlan** *vlan-id* **Example:** `Device(config)# vlan 100`	Enters a VLAN ID to create a VLAN, or enters the VLAN ID of an existing VLAN, and enters VLAN configuration mode. The range is 2 to 1001 and 1006 to 4094. The RSPAN VLAN cannot be VLAN 1 (the default VLAN) or VLAN IDs 1002 through 1005 (reserved for Token Ring and FDDI VLANs).
Step 4	**remote-span** **Example:** `Device(config-vlan)# remote-span`	Configures the VLAN as an RSPAN VLAN.
Step 5	**end** **Example:** `Device(config-vlan)# end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config`	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`startup-config`	

What to do next

You must create the RSPAN VLAN in all devices that will participate in RSPAN. If the RSPAN VLAN-ID is in the normal range (lower than 1005) and VTP is enabled in the network, you can create the RSPAN VLAN in one device, and VTP propagates it to the other devices in the VTP domain. For extended-range VLANs (greater than 1005), you must configure RSPAN VLAN on both source and destination devices and any intermediate devices.

Use VTP pruning to get an efficient flow of RSPAN traffic, or manually delete the RSPAN VLAN from all trunks that do not need to carry the RSPAN traffic.

To remove the remote SPAN characteristic from a VLAN and convert it back to a normal VLAN, use the **no remote-span** VLAN configuration command.

To remove a source port or VLAN from the SPAN session, use the **no monitor session** *session_number* **source** {**interface** *interface-id* | **vlan** *vlan-id*} global configuration command. To remove the RSPAN VLAN from the session, use the **no monitor session** *session_number* **destination remote vlan** *vlan-id*.

Creating an RSPAN Source Session

Follow these steps to create and start an RSPAN source session and to specify the monitored source and the destination RSPAN VLAN.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no monitor session** {*session_number* | **all** | **local** | **remote**} **Example:** `Device(config)# no monitor session 1`	Removes any existing SPAN configuration for the session. • For *session_number*, the range is 1 to 66. • **all**—Removes all SPAN sessions. • **local**—Removes all local sessions.

	Command or Action	Purpose
		• **remote**—Removes all remote SPAN sessions.
Step 4	**monitor session** *session_number* **source** {**interface** *interface-id* | **vlan** *vlan-id*} [**,** | **-**] [**both** | **rx** | **tx**] **Example:** `Device(config)# monitor session 1 source interface gigabitethernet1/0/1 tx`	Specifies the RSPAN session and the source port (monitored port). • For *session_number*, the range is 1 to 66. • Enter a source port or source VLAN for the RSPAN session: • For *interface-id*, specifies the source port to monitor. Valid interfaces include physical interfaces and port-channel logical interfaces (**port-channel** *port-channel-number*). Valid port-channel numbers are 1 to 48. • For *vlan-id*, specifies the source VLAN to monitor. The range is 1 to 4094 (excluding the RSPAN VLAN). A single session can include multiple sources (ports or VLANs), defined in a series of commands, but you cannot combine source ports and source VLANs in one session. • (Optional) [**,** | **-**]—Specifies a series or range of interfaces. Enter a space before and after the comma; enter a space before and after the hyphen. • (Optional) **both** | **rx** | **tx**—Specifies the direction of traffic to monitor. If you do not specify a traffic direction, the source interface sends both sent and received traffic. • **both**—Monitors both received and sent traffic. • **rx**—Monitors received traffic. • **tx**—Monitors sent traffic.
Step 5	**monitor session** *session_number* **destination remote vlan** *vlan-id* **Example:**	Specifies the RSPAN session, the destination RSPAN VLAN, and the destination-port group. • For *session_number*, enter the number defined in Step 4.

	Command or Action	Purpose
	`Device(config)# monitor session 1 destination remote vlan 100`	• For *vlan-id*, specify the source RSPAN VLAN to monitor.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Specifying VLANs to Filter

Follow these steps to configure the RSPAN source session to limit RSPAN source traffic to specific VLANs.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no monitor session** {*session_number* | **all** | **local** | **remote**} **Example:** `Device(config)# no monitor session 2`	Removes any existing SPAN configuration for the session. • For *session_number*, the range is 1 to 66. • **all**—Removes all SPAN sessions.

	Command or Action	Purpose
		• **local**—Removes all local sessions. • **remote**—Removes all remote SPAN sessions.
Step 4	**monitor session** *session_number* **source interface** *interface-id* **Example:** `Device(config)# monitor session 2 source interface gigabitethernet1/0/2 rx`	Specifies the characteristics of the source port (monitored port) and SPAN session. • For *session_number*, the range is 1 to 66. • For *interface-id*, specify the source port to monitor. The interface specified must already be configured as a trunk port.
Step 5	**monitor session** *session_number* **filter vlan** *vlan-id* [**,** | **-**] **Example:** `Device(config)# monitor session 2 filter vlan 1 - 5 , 9`	Limits the SPAN source traffic to specific VLANs. • For *session_number*, enter the session number specified in step 4. • For *vlan-id*, the range is 1 to 4094. • (Optional) **,** | **-** Use a comma (**,**) to specify a series of VLANs or use a hyphen (-) to specify a range of VLANs. Enter a space before and after the comma; enter a space before and after the hyphen.
Step 6	**monitor session** *session_number* **destination remote vlan** *vlan-id* **Example:** `Device(config)# monitor session 2 destination remote vlan 902`	Specifies the RSPAN session and the destination remote VLAN (RSPAN VLAN). • For *session_number*, enter the session number specified in Step 4. • For *vlan-id*, specify the RSPAN VLAN to carry the monitored traffic to the destination port.
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 8	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.

	Command or Action	Purpose
Step 9	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Creating an RSPAN Destination Session

You configure an RSPAN destination session on a different device or device stack; that is, not the device or device stack on which the source session was configured.

Follow these steps to define the RSPAN VLAN on that device, to create an RSPAN destination session, and to specify the source RSPAN VLAN and the destination port.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vlan** *vlan-id* **Example:** `Device(config)# vlan 901`	Specifies the VLAN ID of the RSPAN VLAN created from the source device, and enters VLAN configuration mode. If both devices are participating in VTP and the RSPAN VLAN ID is from 2 to 1005, Steps 3 through 5 are not required because the RSPAN VLAN ID is propagated through the VTP network.
Step 4	**remote-span** **Example:** `Device(config-vlan)# remote-span`	Identifies the VLAN as the RSPAN VLAN.
Step 5	**exit** **Example:**	Returns to global configuration mode.

	Command or Action	Purpose
	`Device(config-vlan)# exit`	
Step 6	**no monitor session** {*session_number* | **all** | **local** | **remote**} **Example:** `Device(config)# no monitor session 1`	Removes any existing SPAN configuration for the session. • For *session_number*, the range is 1 to 66. • **all**—Removes all SPAN sessions. • **local**—Removes all local sessions. • **remote**—Removes all remote SPAN sessions.
Step 7	**monitor session** *session_number* **source remote vlan** *vlan-id* **Example:** `Device(config)# monitor session 1 source remote vlan 901`	Specifies the RSPAN session and the source RSPAN VLAN. • For *session_number*, the range is 1 to 66. • For *vlan-id*, specify the source RSPAN VLAN to monitor.
Step 8	**monitor session** *session_number* **destination interface** *interface-id* **Example:** `Device(config)# monitor session 1 destination interface gigabitethernet2/0/1`	Specifies the RSPAN session and the destination interface. • For *session_number*, enter the number defined in Step 7. In an RSPAN destination session, you must use the same session number for the source RSPAN VLAN and the destination port. • For *interface-id*, specify the destination interface. The destination interface must be a physical interface. • Though visible in the command-line help string, **encapsulation replicate** is not supported for RSPAN. The original VLAN ID is overwritten by the RSPAN VLAN ID, and all packets appear on the destination port as untagged.
Step 9	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 10	**show running-config** **Example:**	Verifies your entries.

	Command or Action	Purpose
	`Device# show running-config`	
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Creating an RSPAN Destination Session and Configuring Incoming Traffic

Follow these steps to create an RSPAN destination session, to specify the source RSPAN VLAN and the destination port, and to enable incoming traffic on the destination port for a network security device (such as a Cisco IDS Sensor Appliance).

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no monitor session** {*session_number* | **all** | **local** | **remote**} **Example:** `Device(config)# no monitor session 2`	Removes any existing SPAN configuration for the session. • For *session_number*, the range is 1 to 66. • **all**—Removes all SPAN sessions. • **local**—Removes all local sessions. • **remote**—Removes all remote SPAN sessions.
Step 4	**monitor session** *session_number* **source remote vlan** *vlan-id* **Example:**	Specifies the RSPAN session and the source RSPAN VLAN. • For *session_number*, the range is 1 to 66.

	Command or Action	Purpose
	`Device(config)# monitor session 2 source remote vlan 901`	• For *vlan-id*, specify the source RSPAN VLAN to monitor.

Step 5	**monitor session** *session_number* **destination** {**interface** *interface-id* [**,** | **-**] [**ingress** {**dot1q vlan** *vlan-id* | **untagged vlan** *vlan-id* | **vlan** *vlan-id*}]}
	Example:
	`Device(config)# monitor session 2 destination interface gigabitethernet1/0/2 ingress vlan 6`
		Specifies the SPAN session, the destination port, the packet encapsulation, and the incoming VLAN and encapsulation.
		• For *session_number*, enter the number defined in Step 5.
		In an RSPAN destination session, you must use the same session number for the source RSPAN VLAN and the destination port.
		• For *interface-id*, specify the destination interface. The destination interface must be a physical interface.
		• Though visible in the command-line help string, **encapsulation replicate** is not supported for RSPAN. The original VLAN ID is overwritten by the RSPAN VLAN ID, and all packets appear on the destination port as untagged.
		• (Optional) [**,** | **-**] Specifies a series or range of interfaces. Enter a space before and after the comma; enter a space before and after the hyphen.
		• Enter **ingress** with additional keywords to enable forwarding of incoming traffic on the destination port and to specify the encapsulation type:
		  • **dot1q vlan** *vlan-id*—Forwards incoming packets with IEEE 802.1Q encapsulation with the specified VLAN as the default VLAN.
		  • **untagged vlan** *vlan-id* or **vlan** *vlan-id*—Forwards incoming packets with untagged encapsulation type with the specified VLAN as the default VLAN.

Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 7	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring an FSPAN Session

Follow these steps to create a SPAN session, specify the source (monitored) ports or VLANs and the destination (monitoring) ports, and configure FSPAN for the session.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no monitor session** {*session_number* | **all** | **local** | **remote**} **Example:** `Device(config)# no monitor session 2`	Removes any existing SPAN configuration for the session. • For *session_number*, the range is 1 to 66. • **all**—Removes all SPAN sessions. • **local**—Removes all local sessions. • **remote**—Removes all remote SPAN sessions.
Step 4	**monitor session** *session_number* **source** {**interface** *interface-id* | **vlan** *vlan-id*} [**,** | **-**] [**both** | **rx** | **tx**]	Specifies the SPAN session and the source port (monitored port). • For *session_number*, the range is 1 to 66.

	Command or Action	Purpose
	Example: `Device(config)# monitor session 2 source interface gigabitethernet1/0/1`	• For *interface-id*, specifies the source port to monitor. Valid interfaces include physical interfaces and port-channel logical interfaces (**port-channel** *port-channel-number*). Valid port-channel numbers are 1 to 48. • For *vlan-id*, specify the source VLAN to monitor. The range is 1 to 4094 (excluding the RSPAN VLAN). **Note** A single session can include multiple sources (ports or VLANs) defined in a series of commands, but you cannot combine source ports and source VLANs in one session. • (Optional) [**,** | **-**]—Specifies a series or range of interfaces. Enter a space before and after the comma; enter a space before and after the hyphen. • (Optional) [**both** | **rx** | **tx**]—Specifies the direction of traffic to monitor. If you do not specify a traffic direction, the SPAN monitors both sent and received traffic. • **both**—Monitors both sent and received traffic. This is the default. • **rx**—Monitors received traffic. • **tx**—Monitors sent traffic. **Note** You can use the **monitor session** *session_number* **source** command multiple times to configure multiple source ports.
Step 5	**monitor session** *session_number* **destination** {**interface** *interface-id* [**,** | **-**] [**encapsulation replicate**]} **Example:** `Device(config)# monitor session 2 destination interface gigabitethernet1/0/2 encapsulation replicate`	Specifies the SPAN session and the destination port (monitoring port). • For *session_number*, specify the session number entered in Step 4. • For **destination**, specify the following parameters: • For *interface-id*, specify the destination port. The destination interface must be a physical port; it

	Command or Action	Purpose
		cannot be an EtherChannel, and it cannot be a VLAN. • (Optional) [**,** | -] Specifies a series or range of interfaces. Enter a space before and after the comma; enter a space before and after the hyphen. • (Optional) **encapsulation replicate** specifies that the destination interface replicates the source interface encapsulation method. If not selected, the default is to send packets in native form (untagged). **Note** For local SPAN, you must use the same session number for the source and destination interfaces. You can use **monitor session** *session_number* **destination** command multiple times to configure multiple destination ports.
Step 6	**monitor session** *session_number* **filter** {**ip** | **ipv6** | **mac**} **access-group** {*access-list-number* | *name*} **Example:** `Device(config)# monitor session 2 filter ipv6 access-group 4`	Specifies the SPAN session, the types of packets to filter, and the ACLs to use in an FSPAN session. • For *session_number*, specify the session number entered in Step 4. • For *access-list-number*, specify the ACL number that you want to use to filter traffic. • For *name*, specify the ACL name that you want to use to filter traffic.
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 8	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.

	Command or Action	Purpose
Step 9	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring an FRSPAN Session

Follow these steps to start an RSPAN source session, specify the monitored source and the destination RSPAN VLAN, and configure FRSPAN for the session.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**no monitor session** {*session_number* | **all** | **local** | **remote**} **Example:** `Device(config)# no monitor session 2`	Removes any existing SPAN configuration for the session. • For *session_number*, the range is 1 to 66. • **all**—Removes all SPAN sessions. • **local**—Removes all local sessions. • **remote**—Removes all remote SPAN sessions.
Step 4	**monitor session** *session_number* **source** {**interface** *interface-id* | **vlan** *vlan-id*} [**,** | **-**] [**both** | **rx** | **tx**] **Example:** `Device(config)# monitor session 2 source interface gigabitethernet1/0/1`	Specifies the SPAN session and the source port (monitored port). • For *session_number*, the range is 1 to 66. • For *interface-id*, specifies the source port to monitor. Valid interfaces include physical interfaces and port-channel logical interfaces (**port-channel** *port-channel-number*). Valid port-channel numbers are 1 to 48.

	Command or Action	Purpose
		• For *vlan-id*, specify the source VLAN to monitor. The range is 1 to 4094 (excluding the RSPAN VLAN). **Note** A single session can include multiple sources (ports or VLANs) defined in a series of commands, but you cannot combine source ports and source VLANs in one session. • (Optional) [**,** | **-**]—Specifies a series or range of interfaces. Enter a space before and after the comma; enter a space before and after the hyphen. • (Optional) [**both** | **rx** | **tx**]—Specifies the direction of traffic to monitor. If you do not specify a traffic direction, the SPAN monitors both sent and received traffic. • **both**—Monitors both sent and received traffic. This is the default. • **rx**—Monitors received traffic. • **tx**—Monitors sent traffic. **Note** You can use the **monitor session** *session_number* **source** command multiple times to configure multiple source ports.
Step 5	**monitor session** *session_number* **destination remote vlan** *vlan-id* **Example:** `Device(config)# monitor session 2 destination remote vlan 5`	Specifies the RSPAN session and the destination RSPAN VLAN. • For *session_number*, enter the number defined in Step 4. • For *vlan-id*, specify the destination RSPAN VLAN to monitor.
Step 6	**vlan** *vlan-id* **Example:** `Device(config)# vlan 10`	Enters the VLAN configuration mode. For *vlan-id*, specify the source RSPAN VLAN to monitor.
Step 7	**remote-span** **Example:**	Specifies that the VLAN you specified in Step 5 is part of the RSPAN VLAN.

	Command or Action	Purpose
	`Device(config-vlan)# remote-span`	
Step 8	**exit** **Example:** `Device(config-vlan)# exit`	Returns to global configuration mode.
Step 9	**monitor session** *session_number* **filter** {**ip** | **ipv6** | **mac**} **access-group** {*access-list-number* | *name*} **Example:** `Device(config)# monitor session 2 filter ip access-group 7`	Specifies the RSPAN session, the types of packets to filter, and the ACLs to use in an FRSPAN session. • For *session_number*, specify the session number entered in Step 4. • For *access-list-number*, specify the ACL number that you want to use to filter traffic. • For *name*, specify the ACL name that you want to use to filter traffic.
Step 10	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 11	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 12	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring SPAN and RSPAN Operations

The following table describes the command used to display SPAN and RSPAN operations configuration and results to monitor operations:

Table 8: Monitoring SPAN and RSPAN Operations

Command	Purpose
show monitor	Displays the current SPAN configuration.

SPAN and RSPAN Configuration Examples

Example: Configuring Local SPAN

This example shows how to set up SPAN session 1 for monitoring source port traffic to a destination port. First, any existing SPAN configuration for session 1 is deleted, and then bidirectional traffic is mirrored from source Gigabit Ethernet port 1 to destination Gigabit Ethernet port 2, retaining the encapsulation method.

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 1
Device(config)# monitor session 1 source interface gigabitethernet1/0/1
Device(config)# monitor session 1 destination interface gigabitethernet1/0/2
encapsulation replicate
Device(config)# end
```

This example shows how to remove port 1 as a SPAN source for SPAN session 1:

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 1 source interface gigabitethernet1/0/1
Device(config)# end
```

This example shows how to disable received traffic monitoring on port 1, which was configured for bidirectional monitoring:

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 1 source interface gigabitethernet1/0/1 rx
```

The monitoring of traffic received on port 1 is disabled, but traffic sent from this port continues to be monitored.

This example shows how to remove any existing configuration on SPAN session 2, configure SPAN session 2 to monitor received traffic on all ports belonging to VLANs 1 through 3, and send it to destination Gigabit Ethernet port 2. The configuration is then modified to also monitor all traffic on all ports belonging to VLAN 10.

```
Device> enable
Device# configure terminal
```

```
Device(config)# no monitor session 2
Device(config)# monitor session 2 source vlan 1 - 3 rx
Device(config)# monitor session 2 destination interface gigabitethernet1/0/2
Device(config)# monitor session 2 source vlan 10
Device(config)# end
```

This example shows how to remove any existing configuration on SPAN session 2, configure SPAN session 2 to monitor received traffic on Gigabit Ethernet source port 1, and send it to destination Gigabit Ethernet port 2 with the same egress encapsulation type as the source port, and to enable ingress forwarding with VLAN 6 as the default ingress VLAN:

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 2
Device(config)# monitor session 2 source gigabitethernet0/1 rx
Device(config)# monitor session 2 destination interface gigabitethernet0/2 encapsulation
replicate ingress vlan 6
Device(config)# end
```

This example shows how to remove any existing configuration on SPAN session 2, configure SPAN session 2 to monitor traffic received on Gigabit Ethernet trunk port 2, and send traffic for only VLANs 1 through 5 and VLAN 9 to destination Gigabit Ethernet port 1:

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 2
Device(config)# monitor session 2 source interface gigabitethernet1/0/2 rx
Device(config)# monitor session 2 filter vlan 1 - 5 , 9
Device(config)# monitor session 2 destination interface gigabitethernet1/0/1
Device(config)# end
```

Examples: Creating an RSPAN VLAN

This example shows how to create the RSPAN VLAN 901:

```
Device> enable
Device# configure terminal
Device(config)# vlan 901
Device(config-vlan)# remote span
Device(config-vlan)# end
```

This example shows how to remove any existing RSPAN configuration for session 1, configure RSPAN session 1 to monitor multiple source interfaces, and configure the destination as RSPAN VLAN 901:

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 1
Device(config)# monitor session 1 source interface gigabitethernet1/0/1 tx
Device(config)# monitor session 1 source interface gigabitethernet1/0/2 rx
Device(config)# monitor session 1 source interface port-channel 2
Device(config)# monitor session 1 destination remote vlan 901
Device(config)# end
```

This example shows how to remove any existing configuration on RSPAN session 2, configure RSPAN session 2 to monitor traffic received on trunk port 2, and send traffic for only VLANs 1 through 5 and 9 to destination RSPAN VLAN 902:

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 2
Device(config)# monitor session 2 source interface gigabitethernet1/0/2 rx
Device(config)# monitor session 2 filter vlan 1 - 5 , 9
Device(config)# monitor session 2 destination remote vlan 902
Device(config)# end
```

This example shows how to configure VLAN 901 as the source remote VLAN and port 1 as the destination interface:

```
Device> enable
Device# configure terminal
Device(config)# monitor session 1 source remote vlan 901
Device(config)# monitor session 1 destination interface gigabitethernet2/0/1
Device(config)# end
```

This example shows how to configure VLAN 901 as the source remote VLAN in RSPAN session 2, to configure Gigabit Ethernet source port 2 as the destination interface, and to enable forwarding of incoming traffic on the interface with VLAN 6 as the default receiving VLAN:

```
Device> enable
Device# configure terminal
Device(config)# monitor session 2 source remote vlan 901
Device(config)# monitor session 2 destination interface gigabitethernet1/0/2 ingress vlan 6
Device(config)# end
```

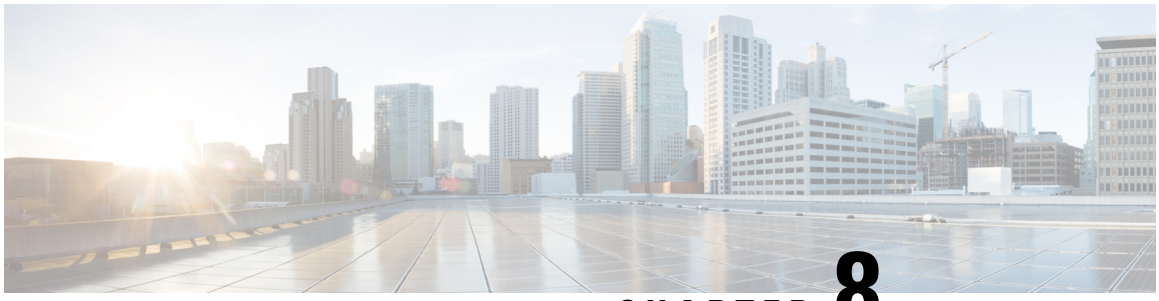
Feature History for SPAN and RSPAN

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	Switch Port Analyzer (SPAN)	Allows monitoring of device traffic on a port or VLAN using a sniffer/analyzer or RMON probe. This feature was introduced.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	Flow-based Switch Port Analyzer (SPAN)	Provides a method to capture only required data between end hosts by using specified filters. The filters are defined in terms of access lists that limit IPv4, IPv6 or IPv4 + IPv6, or non-IP traffic (MAC) between specified source and destination addresses. This feature was introduced.
Cisco IOS XE Everest 16.5.1a	Switch Port Analyzer (SPAN) - distributed egress SPAN	Provides distributed egress SPAN functionality onto line cards in conjunction with ingress SPAN already been distributed to line cards. By distributing egress SPAN functionalities onto line cards, the performance of the system is improved. This feature was introduced.



CHAPTER 8

Configuring ERSPAN

• Configuring ERSPAN, on page 115

Configuring ERSPAN

This module describes how to configure Encapsulated Remote Switched Port Analyzer (ERSPAN). The Cisco ERSPAN feature allows you to monitor traffic on ports or VLANs and send the monitored traffic to destination ports.

Prerequisites for Configuring ERSPAN

- ERSPAN is supported only with the **Network Advantage** license level.
- Access control list (ACL) filter is applied before sending the monitored traffic on to the tunnel.
- Only supports Type-II ERSPAN header.

Restrictions for Configuring ERSPAN

The following restrictions apply for this feature:

- Destination sessions are not supported.
- A device supports up to 66 sessions. A maximum of 8 source sessions can be configured and the remaining sessions can be configured as RSPAN destinations sessions. A source session can be a local SPAN source session or an RSPAN source session or an ERSPAN source session.
- You can configure either a list of ports or a list of VLANs as a source, but cannot configure both for a given session.
- When a session is configured through the ERSPAN CLI, the session ID and the session type cannot be changed. To change them, you must use the no form of the configuration commands to remove the session and then reconfigure the session.
- ERSPAN source sessions do not copy locally-sourced Remote SPAN (RSPAN) VLAN traffic from source trunk ports that carry RSPAN VLANs.
- ERSPAN source sessions do not copy locally-sourced ERSPAN GRE-encapsulated traffic from source ports.

Information for Configuring ERSPAN

ERSPAN Overview

The Cisco ERSPAN feature allows you to monitor traffic on ports or VLANs, and send the monitored traffic to destination ports. ERSPAN sends traffic to a network analyzer, such as a Switch Probe device or a Remote Monitoring (RMON) probe. ERSPAN supports source ports, source VLANs, and destination ports on different devices, which helps remote monitoring of multiple devices across a network.

ERSPAN supports encapsulated packets of up to 9180 bytes. ERSPAN consists of an ERSPAN source session, routable ERSPAN GRE-encapsulated traffic, and an ERSPAN destination session.

ERSPAN consists of an ERSPAN source session, routable ERSPAN GRE-encapsulated traffic, and an ERSPAN destination session. You can configure an ERSPAN source session, an ERSPAN destination session, or both on a device. A device on which only an ERSPAN source session is configured is called an ERSPAN source device, and a device on which only an ERSPAN destination session is configured is called an ERSPAN termination device. A device can act as both; an ERSPAN source device and a termination device.

For a source port or a source VLAN, the ERSPAN can monitor the ingress, egress, or both ingress and egress traffic. By default, ERSPAN monitors all traffic, including multicast, and Bridge Protocol Data Unit (BPDU) frames.

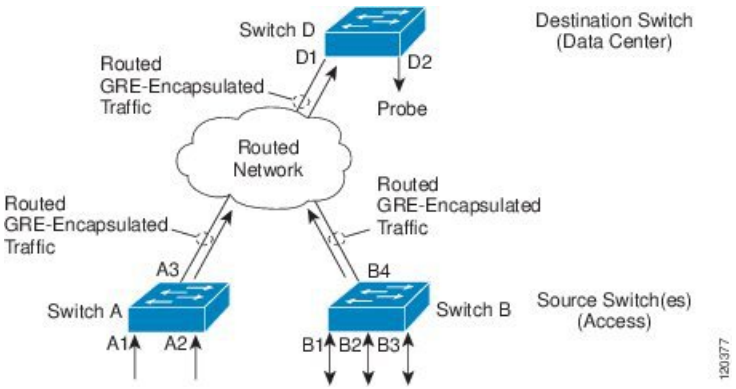
An ERSPAN source session is defined by the following parameters:

- A session ID
- List of source ports or source VLANs to be monitored by the session
- The destination and origin IP addresses, which are used as the destination and source IP addresses of the generic routing encapsulation (GRE) envelope for the captured traffic, respectively
- ERSPAN flow ID
- Optional attributes, such as, IP Time to Live (TTL), related to the GRE envelope

Note ERSPAN source sessions do not copy ERSPAN GRE-encapsulated traffic from source ports. Each ERSPAN source session can have either ports or VLANs as sources, but not both.

Note Because encapsulation is performed in the hardware, the CPU performance is not impacted.

Figure 8: ERSPAN Configuration



ERSPAN Sources

The Cisco ERSPAN feature supports the following sources:

- Source ports—A source port that is monitored for traffic analysis. Source ports in any VLAN can be configured and trunk ports can be configured as source ports along with nontrunk source ports.
- Source VLANs—A VLAN that is monitored for traffic analysis.

The following interfaces are supported as source ports:

- GigabitEthernet
- PortChannel
- TenGigabitEthernet

How to Configure ERSPAN

Configuring an ERSPAN Source Session

The ERSPAN source session defines the session configuration parameters and the ports or VLANs to be monitored.

Procedure

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

	Command or Action	Purpose
Step 3	**monitor session** *span-session-number* **type erspan-source** **Example:** `Switch(config)# monitor session span-session-number type erspan-source`	Defines an ERSPAN source session using the session ID and the session type, and enters ERSPAN monitor source session configuration mode. • Session IDs for source sessions or destination sessions are in the same global ID space, so each session ID is globally unique for both session types. • The *span-session-number* and the session type (configured by the **erspan-source** keyword) cannot be changed once configured. Use the no form of this command to remove the session and then re-create the session with a new session ID or a new session type.
Step 4	**description** *description* **Example:** `Switch(config-mon-erspan-src)# description source1`	Describes the ERSPAN source session.
Step 5	**source** {**interface** *type number* | **vlan** *vlan-ID*} [**,** | **-**| **both** | **rx** | **tx**] **Example:** `Switch(config-mon-erspan-src)# source interface fastethernet 0/1 rx`	Configures the source interface or the VLAN, and the traffic direction to be monitored.
Step 6	**filter** {**ip access-group** {*standard-access-list* | *expanded-access-list* | *acl-name* } | **ipv6 access-group** *acl-name* | **mac access-group** *acl-name* | **vlan** *vlan-ID* [**,** **-**]} **Example:** `Switch(config-mon-erspan-src)# filter vlan 3`	(Optional) Configures source VLAN filtering when the ERSPAN source is a trunk port. • **Note** You cannot include source VLANs and filter VLANs in the same session.
Step 7	**no shutdown** **Example:** `Switch(config-mon-erspan-src)# no shutdown`	Disables the shutting down of the configured session.
Step 8	**destination** **Example:** `Switch(config-mon-erspan-src)# destination`	Defines an ERSPAN destination session and enters ERSPAN monitor destination session configuration mode.

	Command or Action	Purpose
Step 9	**ip address** *ip-address* **Example:** `Switch(config-mon-erspan-src-dst)# ip address 192.0.2.9`	Configures an IP address for the ERSPAN destination session.
Step 10	**erspan-id** *erspan-ID* **Example:** `Switch(config-mon-erspan-src-dst)# erspan-id 2`	Configures the ID used by the destination session to identify the ERSPAN traffic.
Step 11	**origin** *ip-address* **Example:** `Switch(config-mon-erspan-src-dst)# origin ip-address 203.0.113.2`	Configures the IP address used as the destination for the ERSPAN traffic.
Step 12	**ip ttl** *ttl-value* **Example:** `Switch(config-mon-erspan-src-dst)# erspan ttl 32`	Configures Time to Live (TTL) values for packets in the ERSPAN traffic.
Step 13	**end** **Example:** `Switch(config-mon-erspan-src-dst)# end`	Exits ERSPAN monitor destination session configuration mode and returns to privileged EXEC mode.

Configuration Examples for ERSPAN

Example: Configuring an ERSPAN Source Session

```
Switch> enable
Switch# configure terminal
Switch(config)# monitor session 1 type erspan-source
Switch(config-mon-erspan-src)# description source1
Switch(config-mon-erspan-src)# source interface fastethernet 0/1 rx
Switch(config-mon-erspan-src)# filter vlan 3
Switch(config-mon-erspan-src)# no shutdown
Switch(config-mon-erspan-src)# destination
Switch(config-mon-erspan-src-dst)# ip address 192.0.2.9
Switch(config-mon-erspan-src-dst)# erspan-id 2
Switch(config-mon-erspan-src-dst)# origin ip-address 203.0.113.2
Switch(config-mon-erspan-src-dst)# ip ttl 32
Switch(config-mon-erspan-src-dst)# end
```

Verifying ERSPAN

To verify the ERSPAN configuration, use the following commands:

The following is sample output from the **show monitor session erspan-source** command:

```
Switch# show monitor session erspan-source session

Type : ERSPAN Source Session
Status : Admin Enabled
Source Ports :
RX Only : Gi1/4/33
Destination IP Address : 192.0.2.1
Destination ERSPAN ID : 110
Origin IP Address : 10.10.10.216
IPv6 Flow Label : None
```

The following is sample output from the **show monitor session erspan-source detail** command:

```
Switch# show monitor session erspan-source detail

Type : ERSPAN Source Session
Status : Admin Enabled
Description : -
Source Ports :
RX Only : Gi1/4/33
TX Only : None
Both : None
Source VLANs :
RX Only : None
TX Only : None
Both : None
Source RSPAN VLAN : None
Destination Ports : None
Filter VLANs : None
Filter Addr Type :
RX Only : None
TX Only : None
Both : None
Filter Pkt Type :
RX Only : None
Dest RSPAN VLAN : None
IP Access-group : None
IPv6 Access-group : None
Destination IP Address : 192.0.2.1
Destination IPv6 Address : None
Destination IP VRF : None
Destination ERSPAN ID : 110
Origin IP Address : 10.10.10.216
IP QOS PREC : 0
IP TTL : 255
```

The following output from the **show capability feature monitor erspan-source** command displays information about the configured ERSPAN source sessions:

```
Switch# show capability feature monitor erspan-source

ERSPAN Source Session Supported: true
No of Rx ERSPAN source session: 8
No of Tx ERSPAN source session: 8
ERSPAN Header Type supported: II
ACL filter Supported: true
Fragmentation Supported: true
Truncation Supported: false
Sequence number Supported: false
QOS Supported: true
```

The following output from the **show capability feature monitor erspan-destination** command displays all the configured global built-in templates:

```
Switch# show capability feature monitor erspan-destination

ERSPAN Destination Session Supported: false
```

Additional References

RFCs

Standard/RFC	Title
RFC 2784	Generic Routing Encapsulation (GRE)

Technical Assistance

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

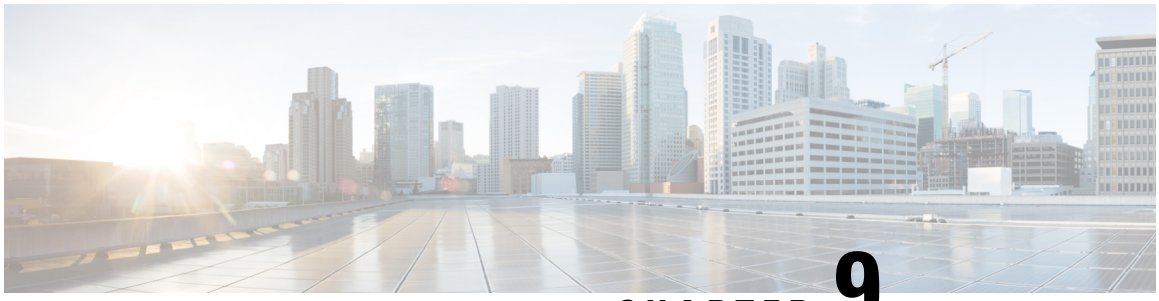
Feature Information for Configuring ERSPAN

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

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

Table 9: Feature Information for Configuring ERSPAN

Feature Name	Releases	Feature Information
ERSPAN	Cisco IOS XE Everest 16.5.1a	This feature was introduced



CHAPTER 9

Configuring Packet Capture

- Prerequisites for Configuring Packet Capture, on page 123
- Restrictions for Configuring Packet Capture, on page 124
- Information About Packet Capture, on page 126
- How to Configure Packet Capture, on page 135
- Configuration Examples for Packet Capture, on page 151
- Additional References, on page 168
- Feature History for Configuring Packet Capture, on page 168

Prerequisites for Configuring Packet Capture

Packet capture is supported on C9500X-28C8D

The following sections provide information about the prerequisites for configuring packet capture.

Prerequisites for Configuring Wireshark

- Wireshark is supported only on switches running DNA Advantage
- Before starting a Wireshark capture process, ensure that CPU usage is moderate and that sufficient memory (at least 200 MB) is available. The CPU usage during Wireshark capture depends on how many packets match the specified conditions and on the intended actions for the matched packets (store, decode and display, or both).

Prerequisites for Configuring Embedded Packet Capture

The Embedded Packet Capture (EPC) software subsystem consumes CPU and memory resources during its operation. You must have adequate system resources for different types of operations. Some guidelines for using the system resources are provided in the table below.

Table 10: System Requirements for the EPC Subsystem

System Resources	Requirements
Hardware	CPU utilization requirements are platform dependent.

System Resources	Requirements
Memory	The packet buffer is stored in DRAM. The size of the packet buffer is user specified.
Diskspace	Packets can be exported to external devices. No intermediate storage on flash disk is required.

Restrictions for Configuring Packet Capture

The following sections provide information about the restrictions for configuring packet capture.

Restrictions for Configuring Wireshark

- Global packet capture on Wireshark is not supported.
- Limiting circular file storage by file size is not supported.
- File limit is limited to the size of the flash in DNA Advantage.
- Wireshark cannot capture packets on a destination SPAN port.
- Wireshark stops capturing when one of the attachment points (interfaces) attached to a capture point stops working. For example, if the device that is associated with an attachment point is unplugged from the device. To resume capturing, the capture must be restarted manually.
- The streaming capture mode supports approximately 1000 pps; lock-step mode supports approximately 2 Mbps (measured with 256-byte packets). When the matching traffic rate exceeds this number, you may experience packet loss.
- You cannot make changes to a capture point when the capture is active.
- Wireshark does not capture packets dropped by floodblock.
- Only one ACL (IPv4, IPv6 or MAC) is allowed in a Wireshark class map.
- ACL logging and Wireshark are incompatible. Once Wireshark is activated, it takes priority. All traffic, including that being captured by ACL logging on any ports, will be redirected to Wireshark. We recommended that you deactivate ACL logging before starting Wireshark. Otherwise, Wireshark traffic will be contaminated by ACL logging traffic.
- If you capture both PACL and RACL on the same port, only one copy is sent to the CPU. If you capture a DTLS-encrypted CAPWAP interface, two copies are sent to Wireshark, one encrypted and the other decrypted. The same behavior will occur if we capture a Layer 2 interface carrying DTLS-encrypted CAPWAP traffic. The core filter is based on the outer CAPWAP header.
- The CLI for configuring Wireshark requires that the feature be executed only from EXEC mode. Actions that usually occur in configuration submode (such as defining capture points), are handled at the EXEC mode instead. All key commands are not NVGEN'd and are not synchronized to the standby supervisor in NSF and SSO scenarios.

Embedded Wireshark is supported with the following limitations:

- Capture filters and display filters are not supported.
- Active capture decoding is not available.
- The output format is different from previous releases.
- A Wireshark session with either a longer duration limit or no capture duration (using a terminal with no auto-more support using the **term len 0** command) may make the console or terminal unusable.

Restrictions for Embedded Packet Capture

- Layer 2 EtherChannels are not supported.
- Neither VRFs, management ports, nor private VLANs can be used as attachment points.
- Embedded Packet Capture (EPC) is not supported on logical ports, which includes port channels, switch virtual interfaces (SVIs), and subinterfaces. It is supported only on physical ports.
- If the user changes interface from switch port to routed port (Layer 2 to Layer 3) or vice versa, they must delete the capture point and create a new one, once the interface comes back up. Stop/start the capture point will not work.
- Packets captured in the output direction of an interface might not reflect the changes made by the device rewrite (includes TTL, VLAN tag, CoS, checksum, MAC addresses, DSCP, precedent, UP, etc.).
- Even though the minimum configurable duration for packet capture is 1 second, packet capture works for a minimum of 2 seconds.
- It is not possible to modify a capture point parameter when a capture is already active or has started.
- EPC captures multicast packets only on ingress and does not capture the replicated packets on egress.
- The Rewrite information of both ingress and egress packets are not captured.
- CPU-injected packets are considered control plane packets. Therefore, these types of packets will not be captured on an interface egress capture.
- Control plane packets are not rate limited and performance impacting. Please use filters to limit control plane packet capture.
- Decoding of protocols such as Control and Provisioning of Wireless Access Points (CAPWAP) is supported in DNA Advantage.
- Up to 8 capture points can be defined, but only one can be active at a time. You need to stop one before you can start the other.
- MAC filter will not capture IP packets even if it matches the MAC address. This applies to all interfaces (Layer 2 switch port, Layer 3 routed port)
- MAC ACL is only used for non-IP packets such as ARP. It will not be supported on a Layer 3 port or SVI.
- MAC filter cannot capture Layer 2 packets (ARP) on Layer 3 interfaces.
- IPv6-based ACLs are not supported in VACL.
- EPC cannot capture based on the underlying routing protocols in MPLS packets.

Information About Packet Capture

The Packet Capture feature is an onboard packet capture facility that allows network administrators to capture packets flowing to, through, and from the device and to analyze them locally or save and export them for offline analysis by using tools such as Wireshark and Embedded Packet Capture (EPC). This feature simplifies network operations by allowing devices to become active participants in the management and operation of the network. This feature facilitates troubleshooting by gathering information about the packet format. This feature also facilitates application analysis and security.

Embedded Packet Capture with Wireshark is supported on DNA Advantage.

About Wireshark

Wireshark is a packet analyzer program that supports multiple protocols and presents information in a text-based user interface.

Wireshark dumps packets to a file using a well known format called .pcap, and is applied or enabled on individual interfaces. You specify an interface in EXEC mode along with the filter and other parameters. The Wireshark application is applied only when you enter a **start** command, and is removed only when Wireshark stops capturing packets either automatically or manually.

Capture Points

A capture point is the central policy definition of the Wireshark feature. The capture point describes all of the characteristics associated with a given instance of Wireshark: which packets to capture, where to capture them from, what to do with the captured packets, and when to stop. Capture points can be modified after creation, and do not become active until explicitly activated with a **start** command. This process is termed activating the capture point or starting the capture point. Capture points are identified by name and can also be manually or automatically deactivated or stopped.

Multiple capture points can be defined, but only one can be active at a time. You need to stop one before you can start the other.

In case of stacked systems, the capture point is activated on the active member. A switchover will terminate any active packet capture session and it will have to be restarted.

Attachment Points

An attachment point is a point in the logical packet process path associated with a capture point. An attachment point is an attribute of the capture point. Packets that impact an attachment point are tested against capture point filters; packets that match are copied and sent to the associated Wireshark instance of the capture point. A specific capture point can be associated with multiple attachment points, with limits on mixing attachment points of different types. Some restrictions apply when you specify attachment points of different types. Attachment points are directional (input or output or both) with the exception of the Layer 2 VLAN attachment point, which is always bidirectional.

In case of stacked systems, the attachment points on all stack members are valid. EPC captures the packets from all the defined attachment points. However these packets are processed only on the active member.

Filters

Filters are attributes of a capture point that identify and limit the subset of traffic traveling through the attachment point of a capture point, which is copied and passed to Wireshark. To be displayed by Wireshark, a packet must pass through an attachment point, as well as all of the filters associated with the capture point.

A capture point has the following types of filters:

- Core system filter—The core system filter is applied by hardware, and its match criteria is limited by hardware. This filter determines whether hardware-forwarded traffic is copied to software for Wireshark purposes.
- Capture filter—The capture filter is applied by Wireshark. The match criteria are more granular than those supported by the core system filter. Packets that pass the core filter but fail the capture filter are still copied and sent to the CPU/software, but are discarded by the Wireshark process. The capture filter syntax matches that of the display filter.

Note Wireshark on the C9500X-28C8D does not use the syntax of the capture filter.

- Display filter—The display filter is applied by Wireshark, and its match criteria are similar to those of the capture filter. Packets that fail the display filter are not displayed.

Core System Filter

You can specify core system filter match criteria by using the class map or ACL, or explicitly by using the CLI.

Note When specifying CAPWAP as an attachment point, the core system filter is not used.

In some installations, you need to obtain authorization to modify the device configuration, which can lead to extended delays if the approval process is lengthy. This can limit the ability of network administrators to monitor and analyze traffic. To address this situation, Wireshark supports explicit specification of core system filter match criteria from the EXEC mode CLI. The disadvantage is that the match criteria that you can specify is a limited subset of what class map supports, such as MAC, IP source and destination addresses, ether-type, IP protocol, and TCP/UDP source and destination ports.

If you prefer to use configuration mode, you can define ACLs or have class maps refer capture points to them. Explicit and ACL-based match criteria are used internally to construct class maps and policy maps.

Note that the ACL and class map configuration are part of the system and not aspects of the Wireshark feature.

Display Filter

With the display filter, you can direct Wireshark to further narrow the set of packets to display when decoding and displaying from a .pcap file.

Actions

Wireshark can be invoked on live traffic or on a previously existing .pcap file. When invoked on live traffic, it can perform four types of actions on packets that pass its display filters:

- Captures to buffer in memory to decode and analyze and store
- Stores to a .pcap file
- Decodes and displays
- Stores and displays

When invoked on a .pcap file only, only the decode and display action is applicable.

Storage of Captured Packets to Buffer in Memory

Packets can be stored in the capture buffer in memory for subsequent decoding, analysis, or storage to a .pcap file.

The capture buffer can be in linear or circular mode. In linear mode, new packets are discarded when the buffer is full. In circular mode, if the buffer is full, the oldest packets are discarded to accommodate the new packets. Although the buffer can also be cleared when needed, this mode is mainly used for debugging network traffic. However, it is not possible to only clear the contents of the buffer alone without deleting it. Stop the current captures and restart the capture again for this to take effect.

Note If you have more than one capture that is storing packets in a buffer, clear the buffer before starting a new capture to avoid memory loss.

Storage of Captured Packets to a .pcap File

Note When Wireshark is used on switches in a stack, packet captures can be stored only on flash or USB flash devices connected to the active switch.

For example, if flash1 is connected to the active switch, and flash2 is connected to the secondary switch, only flash1 can be used to store packet captures.

Attempts to store packet captures on devices other than flash or USB flash devices connected to the active switch will probably result in errors.

Wireshark can store captured packets to a .pcap file. The capture file can be located on the following storage devices:

- Device on-board flash storage (flash:)
- USB drive (usbflash0:)

Note Attempts to store packet captures on unsupported devices or devices not connected to the active switch will probably result in errors.

When configuring a Wireshark capture point, you can associate a filename. When the capture point is activated, Wireshark creates a file with the specified name and writes packets to it. If the file already exists at the time of creation of the capture point, Wireshark queries you as to whether the file can be overwritten. If the file

already exists at the time of activating the capture point, Wireshark will overwrite the existing file. Only one capture point may be associated with a given filename.

If the destination of the Wireshark writing process is full, Wireshark fails with partial data in the file. You must ensure that there is sufficient space in the file system before you start the capture session.

You can reduce the required storage space by retaining only a segment, instead of the entire packet. Typically, you do not require details beyond the first 64 or 128 bytes. The default behavior is to store the entire packet.

To avoid possible packet drops when processing and writing to the file system, Wireshark can optionally use a memory buffer to temporarily hold packets as they arrive. Memory buffer size can be specified when the capture point is associated with a .pcap file.

Packet Decoding and Display

Wireshark can decode and display packets to the console. This functionality is possible for capture points applied to live traffic and for capture points applied to a previously existing .pcap file.

Note Decoding and displaying packets may be CPU intensive.

Wireshark can decode and display packet details for a wide variety of packet formats. The details are displayed by entering the **monitor capture name start** command with one of the following keyword options, which place you into a display and decode mode:

- brief—Displays one line per packet (the default).
- detailed—Decodes and displays all the fields of all the packets whose protocols are supported. Detailed modes require more CPU than the other two modes.
- (hexadecimal) dump—Displays one line per packet as a hexadecimal dump of the packet data and the printable characters of each packet.

When you enter the **capture** command with the decode and display option, the Wireshark output is returned to Cisco IOS and displayed on the console unchanged.

Live Traffic Display

Wireshark receives copies of packets from the core system. Wireshark applies its display filters to discard uninteresting packets, and then decodes and displays the remaining packets.

.pcap File Display

Wireshark can decode and display packets from a previously stored .pcap file and direct the display filter to selectively displayed packets.

Packet Storage and Display

Functionally, this mode is a combination of the previous two modes. Wireshark stores packets in the specified .pcap file and decodes and displays them to the console. Only the core filters are applicable here.

Wireshark Capture Point Activation and Deactivation

After a Wireshark capture point has been defined with its attachment points, filters, actions, and other options, it must be activated. Until the capture point is activated, it does not actually capture packets.

Before a capture point is activated, some functional checks are performed. A capture point cannot be activated if it has neither a core system filter nor attachment points defined. Attempting to activate a capture point that does not meet these requirements generates an error.

The display filters are specified as needed.

After Wireshark capture points are activated, they can be deactivated in multiple ways. A capture point that is storing only packets to a .pcap file can be halted manually or configured with time or packet limits, after which the capture point halts automatically.

When a Wireshark capture point is activated, a fixed rate policer is applied automatically in the hardware so that the CPU is not flooded with Wireshark-directed packets. The disadvantage of the rate policer is that you cannot capture contiguous packets beyond the established rate even if more resources are available.

The set packet capture rate is 1000 packets per sec (pps). The 1000 pps limit is applied to the sum of all attachment points. For example, if we have a capture session with 3 attachment points, the rates of all 3 attachment points added together is policed to 1000 pps.

Note Policer is not supported for control-plane packet capture. When activating control-plane capture points, you need to be extra cautious, so that it does not flood the CPU.

Wireshark Features

This section describes how Wireshark features function in the device environment:

- If port security and Wireshark are applied on an ingress capture, a packet that is dropped by port security will still be captured by Wireshark. If port security is applied on an ingress capture, and Wireshark is applied on an egress capture, a packet that is dropped by port security will not be captured by Wireshark.
- Packets dropped by Dynamic ARP Inspection (DAI) are not captured by Wireshark.
- If a port that is in STP blocked state is used as an attachment point and the core filter is matched, Wireshark will capture the packets that come into the port, even though the packets will be dropped by the switch.
- Classification-based security features—Packets that are dropped by input classification-based security features (such as ACLs and IPSG) are not caught by Wireshark capture points that are connected to attachment points at the same layer. In contrast, packets that are dropped by output classification-based security features are caught by Wireshark capture points that are connected to attachment points at the same layer. The logical model is that the Wireshark attachment point occurs after the security feature lookup on the input side, and symmetrically before the security feature lookup on the output side.

  On ingress, a packet goes through a Layer 2 port, a VLAN, and a Layer 3 port/SVI. On egress, the packet goes through a Layer 3 port/SVI, a VLAN, and a Layer 2 port. If the attachment point is before the point where the packet is dropped, Wireshark will capture the packet. Otherwise, Wireshark will not capture the packet. For example, Wireshark capture policies connected to Layer 2 attachment points in the input direction capture packets dropped by Layer 3 classification-based security features. Symmetrically, Wireshark capture policies attached to Layer 3 attachment points in the output direction capture packets dropped by Layer 2 classification-based security features.

- Routed ports and switch virtual interfaces (SVIs)—Wireshark cannot capture the output of an SVI because the packets that go out of an SVI's output are generated by CPU. To capture these packets, include the control plane as an attachment point.
- VLANs—Starting with Cisco IOS Release 16.1, when a VLAN is used as a Wireshark attachment point, packet capture is supported on L2 and L3 in both input and output directions.
- Redirection features—In the input direction, features traffic redirected by Layer 3 (such as PBR and WCCP) are logically later than Layer 3 Wireshark attachment points. Wireshark captures these packets even though they might later be redirected out another Layer 3 interface. Symmetrically, output features redirected by Layer 3 (such as egress WCCP) are logically prior to Layer 3 Wireshark attachment points, and Wireshark will not capture them.
- SPAN—Wireshark cannot capture packets on interface configured as a SPAN destination.
- SPAN—Wireshark is able to capture packets on interfaces configured as a SPAN source in the ingress direction, and may be available for egress direction too.
- You can capture packets from a maximum of 1000 VLANs at a time, if no ACLs are applied. If ACLs are applied, the hardware will have less space for Wireshark to use. As a result, the maximum number of VLANs than can be used for packet capture at a time will be lower. Using more than 1000 VLANs tunnels at a time or extensive ACLs might have unpredictable results. For example, mobility may go down.

Note Capturing an excessive number of attachment points at the same time is strongly discouraged because it may cause excessive CPU utilization and unpredictable hardware behavior.

Guidelines for Configuring Wireshark

- During Wireshark packet capture, hardware forwarding happens concurrently.
- Because packet forwarding typically occurs in hardware, packets are not copied to the CPU for software processing. For Wireshark packet capture, packets are copied and delivered to the CPU, which causes an increase in CPU usage.
- You might experience high CPU (or memory) usage if:
  - You leave a capture session enabled and unattended for a long period of time, resulting in unanticipated bursts of traffic.
  - You launch a capture session with ring files or capture buffer and leave it unattended for a long time, resulting in performance or system health issues.
- To avoid high CPU usage, do the following:
  - Attach only relevant ports.
  - Use a class map, and secondarily, an access list to express match conditions. If neither is viable, use an explicit, in-line filter.
  - Adhere closely to the filter rules. Restrict the traffic type (such as, IPv4 only) with a restrictive, rather than relaxed ACL, which elicits unwanted traffic.

    - When using Wireshark to capture live traffic, consider applying a QoS policy temporarily to limit the actual traffic until the capture process concludes.

- Always limit packet capture to either a shorter duration or a smaller packet number. The parameters of the capture command enable you to specify the following:
    - Capture duration
    - Number of packets captured
    - File size
    - Packet segment size
- During a capture session, watch for high CPU usage and memory consumption due to Wireshark that may impact device performance or health. If these situations arise, stop the Wireshark session immediately.
- Run a capture session without limits if you know that very little traffic matches the core filter.
- You can define up to eight Wireshark instances. An active **show** command that decodes and displays packets from a .pcap file or capture buffer counts as one instance. However, only one of the instances can be active.
- Whenever an ACL that is associated with a running capture is modified, you must restart the capture for the ACL modifications to take effect. If you do not restart the capture, it will continue to use the original ACL as if it had not been modified.
- Writing to flash disk is a CPU-intensive operation, so if the capture rate is insufficient, you may want to use a buffer capture.
- Avoid decoding and displaying packets from a .pcap file for a large file. Instead, transfer the .pcap file to a PC and run Wireshark on the PC.
- If you plan to store packets to a storage file, ensure that sufficient space is available before beginning a Wireshark capture process.
- To avoid packet loss, consider the following:
    - Use store-only (when you do not specify the display option) while capturing live packets rather than decode and display, which is an CPU-intensive operation (especially in detailed mode).
    - If you have more than one capture that is storing packets in a buffer, clear the buffer before starting a new capture to avoid memory loss.
    - If you use the default buffer size and see that you are losing packets, you can increase the buffer size to avoid losing packets.
- If you want to decode and display live packets in the console window, ensure that the Wireshark session is bounded by a short capture duration.
- The core filter can be an explicit filter, access list, or class map. Specifying a newer filter of these types replaces the existing one.

Note A core filter is required except when using a CAPWAP tunnel interface as a capture point attachment point.

- No specific order applies when defining a capture point; you can define capture point parameters in any order, provided that CLI allows this. The Wireshark CLI allows as many parameters as possible on a single line. This limits the number of commands required to define a capture point.
- All parameters except attachment points take a single value. Generally, you can replace the value with a new one by reentering the command. After user confirmation, the system accepts the new value and overrides the older one. A **no** form of the command is unnecessary to provide a new value, but it is necessary to remove a parameter.
- Wireshark allows you to specify one or more attachment points. To add more than one attachment point, reenter the command with the new attachment point. To remove an attachment point, use the **no** form of the command. You can specify an interface range as an attachment point.

  For example, enter **monitor capture mycap interface GigabitEthernet1/0/1 in** where GigabitEthernet1/0/1 is an attachment point. If you also need to attach interface GigabitEthernet1/0/2, enter it as **monitor capture mycap interface GigabitEthernet1/0/2 in**
- The action you want to perform determines which parameters are mandatory. The Wireshark CLI allows you to specify or modify any parameter prior to entering the **start** command. When you enter the **start** command, Wireshark will start only after determining that all mandatory parameters have been provided.
- If the file already exists at the time of creation of the capture point, Wireshark queries you as to whether the file can be overwritten. If the file already exists at the time of activating the capture point, Wireshark will overwrite the existing file.
- You can terminate a Wireshark session with an explicit **stop** command or by entering **q** in automore mode. The session could terminate itself automatically when a stop condition such as duration or packet capture limit is met, or if an internal error occurs, or resource is full (specifically if disk is full in file mode).
- Dropped packets will not be shown at the end of the capture. However, only the count of dropped and oversized packets will be displayed.

Default Wireshark Configuration

The table below shows the default Wireshark configuration.

Feature	Default Setting
Duration	No limit
Packets	No limit
Packet-length	No limit (full packet)
File size	No limit
Ring file storage	No
Buffer storage mode	Linear

Feature Enhancements for Wireshark

Starting from Cisco IOS XE Everest 16.5.1a:

- Display filters are supported on Wireshark.

- Additional warning message is displayed for control plane capturing.
- When the display is from a cap file, display details of the selected packet can be viewed using packet-number.
- Display filter can be used in file mode.
- The system can query statistics on a pcap cap file's contents, as supported by Wireshark.

About Embedded Packet Capture

EPC provides an embedded systems management facility that helps in tracing and troubleshooting packets. This feature allows network administrators to capture data packets flowing through, to, and from a Cisco device. The network administrator may define the capture buffer size and type (circular, or linear) and the maximum number of bytes of each packet to capture. The packet capture rate can be throttled using further administrative controls. For example, options allow for filtering the packets to be captured using an Access Control List and, optionally, further defined by specifying a maximum packet capture rate or by specifying a sampling interval.

Benefits of Embedded Packet Capture

- Ability to capture IPv4 and IPv6 packets in the device, and also capture non-IP packets with MAC filter or match any MAC address.
- Extensible infrastructure for enabling packet capture points. A capture point is a traffic transit point where a packet is captured and associated with a buffer.
- Facility to export the packet capture in packet capture file (PCAP) format suitable for analysis using any external tool.
- Methods to decode data packets captured with varying degrees of detail.

Packet Data Capture

Packet data capture is the capture of data packets that are then stored in a buffer. You can define packet data captures by providing unique names and parameters.

You can perform the following actions on the capture:

- Activate captures at any interface.
- Apply access control lists (ACLs) or class maps to capture points.

Note Network Based Application Recognition (NBAR) and MAC-style class map is not supported.

- Destroy captures.
- Specify buffer storage parameters such as size and type. The size ranges from 1 MB to 100 MB. The default option for the buffer is linear and the other option for the buffer is circular.
- Specify match criteria that includes information about the protocol, IP address or port address.

Feature Enhancements for Embedded Packet Capture

Starting from Cisco IOS XE Everest 16.5.1a:

- Layer 3 PortChannel is supported.
- Minor changes have been made in the display format.
- Ability to display the number of packets in a cap file
- Clearing the captured buffer deletes the buffer along with its contents. It cannot be run when the packet capture is active.
- In buffer mode, the packet display is allowed only after stop.
- Packet statistics displayed at stop, in DNA Advantage.
- Ability to query the number of packets captured in a pcap file.
- Statistics of packet capture (packets and bytes received or dropped) are displayed after capture stop.
- The packet capture session is always in streaming mode irrespective of the size of the buffer. There is no lock-step mode anymore.

How to Configure Packet Capture

The following sections provide information on configuring packet capture.

How to Configure Wireshark

To configure Wireshark, perform these basic steps.

1. Define a capture point.
2. Add or modify the capture point's parameters.
3. Activate or deactivate a capture point.
4. Delete the capture point when you are no longer using it.

Defining a Capture Point

The example in this procedure defines a very simple capture point. If you choose, you can define a capture point and all of its parameters with one instance of the **monitor capture** command.

Note You must define an attachment point, direction of capture, and core filter to have a functional capture point.

An exception to needing to define a core filter is when you are defining a wireless capture point using a CAPWAP tunneling interface. In this case, you do not define your core filter. It cannot be used.

Follow these steps to define a capture point.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**monitor capture** {*capture-name*} {**interface** *interface-type interface-id* | **control-plane**} {**in** | **out** | **both**} **Example:** `Device# monitor capture mycap interface GigabitEthernet1/0/1 in`	Defines the capture point, specifies the attachment point with which the capture point is associated, and specifies the direction of the capture. The keywords have these meanings: • *capture-name*—Specifies the name of the capture point to be defined (mycap is used in the example). Capture Name should be less than or equal to 8 characters. Only alphanumeric characters and underscore (_) is permitted • (Optional) **interface** *interface-type interface-id*—Specifies the attachment point with which the capture point is associated (GigabitEthernet1/0/1 is used in the example). **Note** Optionally, you can define multiple attachment points and all of the parameters for this capture point with this one command instance. These parameters are discussed in the instructions for modifying capture point parameters. Range support is also available both for adding and removing attachment points. Use one of the following for *interface-type*: • **GigabitEthernet**—Specifies the attachment point as GigabitEthernet. • **vlan**—Specifies the attachment point as a VLAN. **Note** Only ingress capture (in) is allowed when using this interface as an attachment point.

	Command or Action	Purpose
		• **capwap**—Specifies the attachment point as a CAPWAP tunnel. **Note** When using this interface as an attachment point, a core filter cannot be used. • (Optional) **control-plane**—Specifies the control plane as an attachment point. • **in** | **out** | **both**—Specifies the direction of capture.
Step 3	**monitor capture** {*capture-name*} [**match** {**any** | **ipv4 any any** | **ipv6**} **any any**}] **Example:** `Device# monitor capture mycap interface GigabitEthernet1/0/1 in match any`	Defines the core system filter. **Note** When using the CAPWAP tunneling interface as an attachment point, do not perform this step because a core filter cannot be used. The keywords have these meanings: • *capture-name*—Specifies the name of the capture point to be defined (mycap is used in the example). • **match**—Specifies a filter. The first filter defined is the core filter. **Note** A capture point cannot be activated if it has neither a core system filter nor attachment points defined. Attempting to activate a capture point that does not meet these requirements generates an error. • **ipv4**—Specifies an IP version 4 filter. • **ipv6**—Specifies an IP version 6 filter.

Step 4	**show monitor capture** {*capture-name*} [**parameter**]
	Example:
	`Device# show monitor capture mycap parameter`
	`monitor capture mycap interface GigabitEthernet1/0/1 in`
	`monitor capture mycap match any`
		Displays the capture point parameters that you defined in Step 2 and confirms that you defined a capture point.

	Command or Action	Purpose
Step 5	**show capwap summary** **Example:** `Device# show capwap summary`	Displays the CAPWAP tunnels available as attachment points for a wireless capture. **Note** Use this command only if you are using a CAPWAP tunnel as an attachment point to perform a wireless capture. See the CAPWAP example in the examples section.
Step 6	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Example

To define a capture point with a CAPWAP attachment point:

```
Device# show capwap summary

CAPWAP Tunnels General Statistics:
  Number of Capwap Data Tunnels       = 1
  Number of Capwap Mobility Tunnels   = 0
  Number of Capwap Multicast Tunnels  = 0


Name   APName                           Type PhyPortIf Mode      McastIf
------ -------------------------------- ---- --------- --------- -------
Ca0    AP442b.03a9.6715                 data Gi3/0/6   unicast   -


Name   SrcIP           SrcPort DestIP          DstPort DtlsEn MTU   Xact
------ --------------- ------- --------------- ------- ------ ----- ----
Ca0    10.10.14.32     5247    10.10.14.2      38514   No     1449  0

Device# monitor capture mycap interface capwap 0 both
Device# monitor capture mycap file location flash:mycap.pcap
Device# monitor capture mycap file buffer-size 1
Device# monitor capture mycap start

*Aug 20 11:02:21.983: %BUFCAP-6-ENABLE: Capture Point mycap enabled.on

Device# show monitor capture mycap parameter
   monitor capture mycap interface capwap  0 in
   monitor capture mycap interface capwap  0 out
   monitor capture mycap file location flash:mycap.pcap buffer-size 1
Device#
```

```
Device# show monitor capture mycap

Status Information for Capture mycap
  Target Type:
  Interface: CAPWAP,
    Ingress:
 0
    Egress:
 0
   Status : Active
  Filter Details:
    Capture all packets
  Buffer Details:
   Buffer Type: LINEAR (default)
  File Details:
   Associated file name: flash:mycap.pcap
   Size of buffer(in MB): 1
  Limit Details:
   Number of Packets to capture: 0 (no limit)
   Packet Capture duration: 0 (no limit)
   Packet Size to capture: 0 (no limit)
   Packets per second: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
Device#
Device# show monitor capture file flash:mycap.pcap
  1   0.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  2   0.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  3   2.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  4   2.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  5   3.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  6   4.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  7   4.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  8   5.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  9   5.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 10   6.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 11   8.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 12   9.225986   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 13   9.225986   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 14   9.225986   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 15   9.231998   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 16   9.231998   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 17   9.231998   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 18   9.236987   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 19  10.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 20  10.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 21  12.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 22  12.239993   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 23  12.244997   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 24  12.244997   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 25  12.250994   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
```

```
26  12.256990   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
27  12.262987   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
28  12.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
Flags=........
29  12.802012   10.10.14.3 -> 10.10.14.255 NBNS Name query NB WPAD.<00>
30  13.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
Flags=........
```

What to do next

You can add additional attachment points, modify the parameters of your capture point, then activate it, or if you want to use your capture point just as it is, you can now activate it.

Note You cannot change a capture point's parameters using the methods presented in this topic.

If the user enters an incorrect capture name, or an invalid/non existing attachment point, the switch will show errors like "*Capture Name should be less than or equal to 8 characters. Only alphanumeric characters and underscore (_) is permitted*" and "*% Invalid input detected at '^' marker*" respectively.

Adding or Modifying Capture Point Parameters

Although listed in sequence, the steps to specify values for the parameters can be executed in any order. You can also specify them in one, two, or several lines. Except for attachment points, which can be multiple, you can replace any value with a more recent value by redefining the same option. You will need to confirm interactively when certain parameters already specified are being modified.

Follow these steps to modify a capture point's parameters.

Before you begin

A capture point must be defined before you can use these instructions.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**monitor capture** {*capture-name*} **match** {**any** | **mac** *mac-match-string* | **ipv4** {**any** | **host** | **protocol**}{**any** | **host**} | **ipv6** {**any** | **host** | **protocol**}{**any** | **host**}} **Example:** `Device# monitor capture mycap match ipv4 any any`	Defines the core system filter (**ipv4 any any**), defined either explicitly, through ACL or through a class map.

	Command or Action	Purpose
Step 3	**monitor capture** {*capture-name*} **limit** {[**duration** *seconds*] [**packet-length** *size*] [**packets** *num*]} **Example:** `Device# monitor capture mycap limit duration 60 packet-len 400`	Specifies the session limit in seconds (60), packets captured, or the packet segment length to be retained by Wireshark (400).
Step 4	**monitor capture** {*capture-name*} **file** {**location** *filename*} **Example:** `Device# monitor capture mycap file location flash:mycap.pcap`	Specifies the file association, if the capture point intends to capture packets rather than only display them. **Note** If the file already exists, you have to confirm if it can be overwritten.
Step 5	**monitor capture** {*capture-name*} **file** {**buffer-size** *size*} **Example:** `Device# monitor capture mycap file buffer-size 100`	Specifies the size of the memory buffer used by Wireshark to handle traffic bursts.

Step 6	**show monitor capture** {*capture-name*} [**parameter**]
	Example:
	`Device# show monitor capture mycap parameter`
	`monitor capture mycap interface GigabitEthernet1/0/1 in`
	`monitor capture mycap match ipv4 any any`
	`monitor capture mycap limit duration 60 packet-len 400`
	`monitor capture point mycap file location bootdisk:mycap.pcap`
	`monitor capture mycap file buffer-size 100`
		Displays the capture point parameters that you defined previously.

Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Modifying Parameters

Associating or Disassociating a Capture File

```
Device# monitor capture point mycap file location flash:mycap.pcap
Device# no monitor capture mycap file
```

Specifying a Memory Buffer Size for Packet Burst Handling

```
Device# monitor capture mycap buffer size 100
```

Defining an Explicit Core System Filter to Match Both IPv4 and IPv6

```
Device# monitor capture mycap match any
```

What to do next

If your capture point contains all of the parameters you want, activate it.

Deleting Capture Point Parameters

Although listed in sequence, you can execute the steps to delete parameters in any order. You can also delete them in one, two, or several lines. Except for attachment points, which can be multiple, you can delete any parameter.

Follow these steps to delete the parameters of a capture point.

Before you begin

Define parameters of a capture point before you can use these instructions to delete it.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**no monitor capture** {*capture-name*} **match** **Example:** `Device# no monitor capture mycap match`	Deletes all filters defined on capture point (mycap).

Step 3	**no monitor capture** {*capture-name*} **limit** [**duration**] [**packet-length**] [**packets**]
	Example:
	`Device# no monitor capture mycap limit duration packet-len`
	`Device# no monitor capture mycap limit`
		Deletes the session time limit and the packet segment length retained by Wireshark. It leaves other specified limits in place.
		Deletes all limits on Wireshark.

Step 4	**no monitor capture** {*capture-name*} **file** [**location**] [**buffer-size**]
	Example:
	`Device# no monitor capture mycap file`
	`Device# no monitor capture mycap file location`
		Deletes the file association. The capture point will no longer capture packets. It only displays them.
		Deletes the file location association. The file location is no longer associated with the capture point. However, other defined file association is unaffected by this action.

Step 5	**show monitor capture** {*capture-name*} [**parameter**]	Displays the capture point parameters that remain defined after your parameter deletion operations. You can run this command at any

	Command or Action	Purpose

	Example:
	`Device# show monitor capture mycap parameter`
	`monitor capture mycap interface GigabitEthernet1/0/1 in`
		point in the procedure to see what parameters are associated with a capture point.

Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

What to do next

If your capture point contains all the parameters you want, activate it.

Note If you delete the parameters when the capture point is active, the switch shows an error "*Capture is active*".

Deleting a Capture Point

Follow these steps to delete a capture point.

Before you begin

Define a capture point before you can use these instructions to delete it. Stop the capture point before you can delete it.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**no monitor capture** {*capture-name*} **Example:** `Device# no monitor capture mycap`	Deletes the specified capture point (mycap).

Step 3	**show monitor capture** {*capture-name*}[**parameter**]
	Example:
	`Device# show monitor capture mycap parameter`
	`Capture mycap does not exist`
		Displays a message indicating that the specified capture point doesn't exist because it was deleted.

	Command or Action	Purpose
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

You can define a new capture point with the same name as the one you deleted. You can perform these instructions when you want to start over with defining a capture point.

Activating and Deactivating a Capture Point

Follow these steps to activate or deactivate a capture point.

Before you begin

A capture point can be activated even if an attachment point and a core system filter have been defined and the associated filename already exists. In such an instance, the existing file will be overwritten.

A capture point with no associated filename can only be activated to display. When the filename is not specified, the packets are captured into the buffer. Live display (display during capture) is available in both file and buffer modes.

If no display filters are specified, packets are not displayed live, and all the packets captured by the core system filter are displayed. The default display mode is brief.

Note When using a CAPWAP tunneling interface as an attachment point, core filters are not used, so there is no requirement to define them in this case.

Procedure

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

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**monitor capture** {*capture-name*} **start** [**display** [**display-filter** *filter-string*]] [**brief** | **detailed** | **dump**] **Example:** `Device# monitor capture mycap start display display-filter "stp"`	Activates a capture point and filters the display, so only packets containing "stp" are displayed.
Step 3	**monitor capture** {*capture-name*} **stop** **Example:** `Device# monitor capture name stop`	Deactivates a capture point.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

While activating and deactivating a capture point, you could encounter a few errors. Here are examples of some of the possible errors.

Missing attachment point on activation

```
Switch#monitor capture mycap match any
Switch#monitor capture mycap start
No Target is attached to capture failed to disable provision featurefailed to remove
policyfailed to disable provision featurefailed to remove policyfailed to disable provision
 featurefailed to remove policy
Capture statistics collected at software (Buffer):
 Capture duration - 0 seconds
 Packets received - 0
 Packets dropped - 0
 Packets oversized - 0

Unable to activate Capture.
```

```
Switch# unable to get action unable to get action unable to get action
Switch#monitor capture mycap interface g1/0/1 both
Switch#monitor capture mycap start
Switch#
*Nov 5 12:33:43.906: %BUFCAP-6-ENABLE: Capture Point mycap enabled.
```

Missing filter on activation

```
Switch#monitor capture mycap int g1/0/1 both
Switch#monitor capture mycap start
Filter not attached to capture
Capture statistics collected at software (Buffer):
 Capture duration - 0 seconds
 Packets received - 0
 Packets dropped - 0
 Packets oversized - 0

Unable to activate Capture.
Switch#monitor capture mycap match any
Switch#monitor capture mycap start
Switch#
*Nov 5 12:35:37.200: %BUFCAP-6-ENABLE: Capture Point mycap enabled.
```

Attempting to activate a capture point while another one is already active

```
Switch#monitor capture mycap start
PD start invoked while previous run is active Failed to start capture : Wireshark operation
 failure
Unable to activate Capture.
Switch#show monitor capture

Status Information for Capture test
 Target Type:
 Interface: GigabitEthernet1/0/13, Direction: both
 Interface: GigabitEthernet1/0/14, Direction: both
 Status : Active
 Filter Details:
 Capture all packets
 Buffer Details:
 Buffer Type: LINEAR (default)
 Buffer Size (in MB): 10
 File Details:
 Associated file name: flash:cchh.pcap
 Limit Details:
 Number of Packets to capture: 0 (no limit)
 Packet Capture duration: 0 (no limit)
 Packet Size to capture: 0 (no limit)
 Maximum number of packets to capture per second: 1000
 Packet sampling rate: 0 (no sampling)

Status Information for Capture mycap
 Target Type:
 Interface: GigabitEthernet1/0/1, Direction: both
 Status : Inactive
 Filter Details:
 Capture all packets
 Buffer Details:
 Buffer Type: LINEAR (default)
 Buffer Size (in MB): 10
 File Details:
 File not associated
 Limit Details:
 Number of Packets to capture: 0 (no limit)
 Packet Capture duration: 0 (no limit)
 Packet Size to capture: 0 (no limit)
```

```
 Maximum number of packets to capture per second: 1000
 Packet sampling rate: 0 (no sampling)
Switch#monitor capture test stop
Capture statistics collected at software (Buffer & Wireshark):
 Capture duration - 157 seconds
 Packets received - 0
 Packets dropped - 0
 Packets oversized - 0

Switch#
*Nov 5 13:18:17.406: %BUFCAP-6-DISABLE: Capture Point test disabled.
Switch#monitor capture mycap start
Switch#
*Nov 5 13:18:22.664: %BUFCAP-6-ENABLE: Capture Point mycap enabled.
Switch#
```

Clearing the Capture Point Buffer

Follow these steps to clear the buffer contents or save them to an external file for storage.

Note If you have more than one capture that is storing packets in a buffer, clear the buffer before starting a new capture to avoid memory loss. Don't try to clear buffer on an active capture point.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**monitor capture** {*capture-name*} [**clear** | **export** *filename*] **Example:** `Device# monitor capture mycap clear`	Clear - Completely deletes the buffer. **Note** When you run the clear command, • On DNA Advantage license - the command clears the buffer contents without deleting the buffer. • On all other licenses - the command deletes the buffer itself. Export - Saves the captured packets in the buffer and deletes the buffer.
Step 3	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 4	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 5	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Examples: Capture Point Buffer Handling

Exporting Capture to a File

```
Device# monitor capture mycap export flash:mycap.pcap

Storage configured as File for this capture
```

Clearing Capture Point Buffer

```
Device# monitor capture mycap clear

 Capture configured with file options
```

What to do next

Note If you try to clear the capture point buffer on licenses other than DNA Advantage, the switch shows an error "*Failed to clear capture buffer: Capture Buffer BUSY*".

How to Implement Embedded Packet Capture

Managing Packet Data Capture

Note Export of an active capture point is only supported on DNA Advantage. For all other type of licenses you need to stop the capture first and only then export.

To manage Packet Data Capture in the buffer mode, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**monitor capture** *capture-name* **access-list** *access-list-name* **Example:** `Device# monitor capture mycap access-list v4acl`	Configures a monitor capture specifying an access list as the core filter for the packet capture.
Step 3	**monitor capture** *capture-name* **limit duration** *seconds* **Example:** `Device# monitor capture mycap limit duration 1000`	Configures monitor capture limits.
Step 4	**monitor capture** *capture-name* **interface** *interface-name* **both** **Example:** `Device# monitor capture mycap interface GigabitEthernet 0/0/1 both`	Configures monitor capture specifying an attachment point and the packet flow direction.
Step 5	**monitor capture** *capture-name* **buffer circular size** *bytes* **Example:** `Device# monitor capture mycap buffer circular size 10`	Configures a buffer to capture packet data.
Step 6	**monitor capture** *capture-name* **start** **Example:** `Device# monitor capture mycap start`	Starts the capture of packet data at a traffic trace point into a buffer.
Step 7	**monitor capture** *capture-name* **stop** **Example:** `Device# monitor capture mycap stop`	Stops the capture of packet data at a traffic trace point.
Step 8	**monitor capture** *capture-name* **export** *file--location/file-name* **Example:**	Exports captured data for analysis.

	Command or Action	Purpose
	`Device# monitor capture mycap export tftp://10.1.88.9/mycap.pcap`	
Step 9	**end** **Example:** `Device# end`	Returns to privileged EXEC mode.

Monitoring and Maintaining Captured Data

Perform this task to monitor and maintain the packet data captured. Capture buffer details and capture point details are displayed.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**show monitor capture** *capture-buffer-name* **buffer dump** **Example:** `Device# show monitor capture mycap buffer dump`	(Optional) Displays a hexadecimal dump of captured packet and its metadata.
Step 3	**show monitor capture** *capture-buffer-name* **parameter** **Example:** `Device# show monitor capture mycap parameter`	(Optional) Displays a list of commands that were used to specify the capture.
Step 4	**debug epc capture-point** **Example:** `Device# debug epc capture-point`	(Optional) Enables packet capture point debugging.
Step 5	**debug epc provision** **Example:** `Device# debug epc provision`	(Optional) Enables packet capture provisioning debugging.

	Command or Action	Purpose
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Configuration Examples for Packet Capture

The following sections provide configuration examples for packet capture.

Configuration Examples for Wireshark

The following sections provide configuration examples for Wireshark.

Example: Displaying a Brief Output from a .pcap File

You can display the output from a .pcap file by entering:

```
Device# show monitor capture file flash:mycap.pcap brief
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1 0.000000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=0/0, ttl=254
  2 0.000051000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=0/0, ttl=255 (request in 1)
  3 0.000908000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=1/256, ttl=254
  4 0.001782000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=1/256, ttl=255 (request in 3)
  5 0.002961000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=2/512, ttl=254
  6 0.003676000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=2/512, ttl=255 (request in 5)
  7 0.004835000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=3/768, ttl=254
  8 0.005579000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=3/768, ttl=255 (request in 7)
  9 0.006850000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=4/1024, ttl=254
 10 0.007586000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=4/1024, ttl=255 (request in 9)
 11 0.008768000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=5/1280, ttl=254
 12 0.009497000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=5/1280, ttl=255 (request in 11)
 13 0.010695000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=6/1536, ttl=254
 14 0.011427000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=6/1536, ttl=255 (request in 13)
 15 0.012728000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=7/1792, ttl=254
 16 0.013458000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=7/1792, ttl=255 (request in 15)
```

```
 17 0.014652000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=8/2048, ttl=254
 18 0.015394000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=8/2048, ttl=255 (request in 17)
 19 0.016682000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=9/2304, ttl=254
 20 0.017439000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=9/2304, ttl=255 (request in 19)
 21 0.018655000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=10/2560, ttl=254
 22 0.019385000   10.10.10.1 -> 10.10.10.2   ICMP 114 Echo (ping) reply    id=0x002e,
seq=10/2560, ttl=255 (request in 21)
 23 0.020575000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=11/2816, ttl=254
 --More‹
```

Example: Displaying Detailed Output from a .pcap File

You can display the detailed .pcap file output by entering:

```
Device# show monitor capture file flash:mycap.pcap detailed
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

Frame 1: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
    Interface id: 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov  6, 2015 11:44:48.322497000 UTC
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1446810288.322497000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 114 bytes (912 bits)
    Capture Length: 114 bytes (912 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:icmp:data]
Ethernet II, Src: Cisco_f3:63:46 (00:e1:6d:f3:63:46), Dst: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)

    Destination: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)
        Address: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: Cisco_f3:63:46 (00:e1:6d:f3:63:46)
        Address: Cisco_f3:63:46 (00:e1:6d:f3:63:46)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.2 (10.10.10.2), Dst: 10.10.10.1 (10.10.10.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not
ECN-Capable Transport))
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport)
(0x00)
    Total Length: 100
    Identification: 0x04ba (1210)
    Flags: 0x00
        0... .... = Reserved bit: Not set
```

```
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 254
    Protocol: ICMP (1)
    Header checksum: 0x8fc8 [validation disabled]
        [Good: False]
        [Bad: False]
    Source: 10.10.10.2 (10.10.10.2)
    Destination: 10.10.10.1 (10.10.10.1)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xe4db [correct]
    Identifier (BE): 46 (0x002e)
    Identifier (LE): 11776 (0x2e00)
    Sequence number (BE): 0 (0x0000)
    Sequence number (LE): 0 (0x0000)
    Data (72 bytes)

0000  00 00 00 00 09 c9 8f 77 ab cd ab cd ab cd ab cd   .......w........
0010  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0020  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0030  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0040  ab cd ab cd ab cd ab cd                           ........
        Data: 0000000009c98f77abcdabcdabcdabcdabcdabcdabcdabcd...
        [Length: 72]

Frame 2: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
    Interface id: 0
```

Example: Displaying a Packet Dump Output from a .pcap File.

You can display the packet dump output by entering:

```
Device# show monitor capture file flash:mycap.pcap dump
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

0000  00 e1 6d 31 f1 c6 00 e1 6d f3 63 46 08 00 45 00   ..m1....m.cF..E.
0010  00 64 04 ba 00 00 fe 01 8f c8 0a 0a 0a 02 0a 0a   .d..............
0020  0a 01 08 00 e4 db 00 2e 00 00 00 00 00 00 09 c9   ................
0030  8f 77 ab cd ab cd ab cd ab cd ab cd ab cd ab cd   .w..............
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0050  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0060  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0070  ab cd                                            ..

0000  00 e1 6d 31 f1 80 00 e1 6d 31 f1 80 08 00 45 00   ..m1....m1....E.
0010  00 64 04 ba 00 00 ff 01 8e c8 0a 0a 0a 01 0a 0a   .d..............
0020  0a 02 00 00 ec db 00 2e 00 00 00 00 00 00 09 c9   ................
0030  8f 77 ab cd ab cd ab cd ab cd ab cd ab cd ab cd   .w..............
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0050  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0060  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0070  ab cd                                            ..

0000  00 e1 6d 31 f1 c6 00 e1 6d f3 63 46 08 00 45 00   ..m1....m.cF..E.
0010  00 64 04 bb 00 00 fe 01 8f c7 0a 0a 0a 02 0a 0a   .d..............
0020  0a 01 08 00 e4 d7 00 2e 00 01 00 00 00 00 09 c9   ................
0030  8f 7a ab cd ab cd ab cd ab cd ab cd ab cd ab cd   .z..............
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
```

Example: Displaying Packets from a .pcap File using a Display Filter

You can display the .pcap file packets output by entering:

```
Device# show monitor capture file flash:mycap.pcap display-filter "ip.src == 10.10.10.2"
brief
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1 0.000000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=0/0, ttl=254
  3 0.000908000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=1/256, ttl=254
  5 0.002961000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=2/512, ttl=254
  7 0.004835000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=3/768, ttl=254
  9 0.006850000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=4/1024, ttl=254
 11 0.008768000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=5/1280, ttl=254
 13 0.010695000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=6/1536, ttl=254
 15 0.012728000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=7/1792, ttl=254
 17 0.014652000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=8/2048, ttl=254
 19 0.016682000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=9/2304, ttl=254
 21 0.018655000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=10/2560, ttl=254
 23 0.020575000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x002e,
seq=11/2816, ttl=254
```

Example: Displaying the Number of Packets Captured in a .pcap File

You can display the number of packets captured in a .pcap file by entering:

```
Device# show monitor capture file flash:mycap.pcap packet-count
File name:           /flash/mycap.pcap
Number of packets:   50
```

Example: Displaying a Single Packet Dump from a .pcap File

You can display a single packet dump from a .pcap file by entering:

```
Device#  show monitor capture file flash:mycap.pcap packet-number 10 dump
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

0000  00 e1 6d 31 f1 80 00 e1 6d 31 f1 80 08 00 45 00   ..m1....m1....E.
0010  00 64 04 be 00 00 ff 01 8e c4 0a 0a 0a 01 0a 0a   .d..............
0020  0a 02 00 00 ec ce 00 2e 00 04 00 00 00 00 09 c9   ................
0030  8f 80 ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0050  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0060  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0070  ab cd
```

Example: Displaying Statistics of Packets Captured in a .pcap File

You can display the statistics of the packets captured in a .pcap file by entering:

```
Device# show monitor capture file flash:mycap.pcap statistics "h225,counter"
================== H225 Message and Reason Counter ==================
RAS-Messages:
Call Signalling:
=====================================================================
```

Example: Simple Capture and Display

This example shows how to monitor traffic in the Layer 3 interface Gigabit Ethernet 1/0/1:

Step 1: Define a capture point to match on the relevant traffic by entering:

```
Device# monitor capture mycap interface GigabitEthernet1/0/3 in
Device# monitor capture mycap match ipv4 any any
Device# monitor capture mycap limit duration 60 packets 50
Device# monitor capture mycap buffer size 100
```

To avoid high CPU utilization, a low packet count and duration as limits has been set.

Step 2: Confirm that the capture point has been correctly defined by entering:

```
Device# show monitor capture mycap parameter
      monitor capture mycap interface GigabitEthernet1/0/3 in
   monitor capture mycap match ipv4  any any
   monitor capture mycap buffer size 100
   monitor capture mycap limit packets 50 duration 60

Device# show monitor capture mycap
Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/3, Direction: in
   Status : Inactive
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  Buffer Details:
   Buffer Type: LINEAR (default)
   Buffer Size (in MB): 100
  File Details:
   File not associated
  Limit Details:
   Number of Packets to capture: 50
   Packet Capture duration: 60
   Packet Size to capture: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
```

Step 3: Start the capture process and display the results.

```
Device# monitor capture mycap start display
 Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1   0.000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030, seq=0/0,
 ttl=254
  2   0.003682   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=1/256, ttl=254
  3   0.006586   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
```

```
seq=2/512, ttl=254
  4   0.008941   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=3/768, ttl=254
  5   0.011138   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=4/1024, ttl=254
  6   0.014099   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=5/1280, ttl=254
  7   0.016868   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=6/1536, ttl=254
  8   0.019210   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=7/1792, ttl=254
  9   0.024785   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0030,
seq=8/2048, ttl=254
 --More--
```

Step 4: Delete the capture point by entering:

```
Device# no monitor capture mycap
```

Note A **stop** command is not required in this particular case since we have set a limit and the capture will automatically stop once that limit is reached.

For more information on syntax to be used for pcap statistics, refer the "*Additional References*" section.

Example: Simple Capture and Store

This example shows how to capture packets to a filter:

Step 1: Define a capture point to match on the relevant traffic and associate it to a file by entering:

```
Device# monitor capture mycap interface GigabitEthernet1/0/3 in
Device# monitor capture mycap match ipv4 any any
Device# monitor capture mycap limit duration 60 packets 50
Device# monitor capture mycap file location flash:mycap.pcap
```

Step 2: Confirm that the capture point has been correctly defined by entering:

```
Device# show monitor capture mycap parameter
   monitor capture mycap interface GigabitEthernet1/0/3 in
   monitor capture mycap match ipv4  any any
   monitor capture mycap file location flash:mycap.pcap
   monitor capture mycap limit packets 50 duration 60

Device# show monitor capture mycap

Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/3, Direction: in
   Status : Inactive
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  Buffer Details:
   Buffer Type: LINEAR (default)
```

```
  File Details:
   Associated file name: flash:mycap.pcap
  Limit Details:
   Number of Packets to capture: 50
   Packet Capture duration: 60
   Packet Size to capture: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
```

Step 3: Launch packet capture by entering:

```
Device# monitor capture mycap start
```

Step 4: Display extended capture statistics during runtime by entering:

```
Device#  show monitor capture mycap capture-statistics
Capture statistics collected at software:
        Capture duration - 15 seconds
        Packets received - 40
        Packets dropped - 0
        Packets oversized - 0
        Packets errored - 0
        Packets sent - 40
        Bytes received - 7280
        Bytes dropped - 0
        Bytes oversized - 0
        Bytes errored - 0
        Bytes sent ¬ 4560
```

Step 5: After sufficient time has passed, stop the capture by entering:

```
Device# monitor capture mycap stop
Capture statistics collected at software (Buffer & Wireshark):
        Capture duration - 20 seconds
        Packets received - 50
        Packets dropped - 0
        Packets oversized - 0
```

Note Alternatively, you could allow the capture operation stop automatically after the time has elapsed or the packet count has been met.

The mycap.pcap file now contains the captured packets.

Step 6: Display extended capture statistics after stop by entering:

```
Device#  show monitor capture mycap capture-statistics
Capture statistics collected at software:
        Capture duration - 20 seconds
        Packets received - 50
        Packets dropped - 0
        Packets oversized - 0
        Packets errored - 0
        Packets sent - 50
        Bytes received - 8190
        Bytes dropped - 0
        Bytes oversized - 0
        Bytes errored - 0
        Bytes sent ¬ 5130
```

Step 7: Display the packets by entering:

```
Device# show monitor capture file flash:mycap.pcap
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1 0.000000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=0/0, ttl=254
  2 0.002555000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=1/256, ttl=254
  3 0.006199000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=2/512, ttl=254
  4 0.009199000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=3/768, ttl=254
  5 0.011647000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=4/1024, ttl=254
  6 0.014168000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=5/1280, ttl=254
  7 0.016737000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=6/1536, ttl=254
  8 0.019403000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=7/1792, ttl=254
  9 0.022151000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=8/2048, ttl=254
 10 0.024722000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=9/2304, ttl=254
 11 0.026890000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=10/2560, ttl=254
 12 0.028862000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0031,
seq=11/2816, ttl=254
 --More--
```

For more information on syntax to be used for pcap statistics, refer the "*Additional References*" section.

Step 8: Delete the capture point by entering:

```
Device# no monitor capture mycap
```

Example: Using Buffer Capture

This example shows how to use buffer capture:

Step 1: Launch a capture session with the buffer capture option by entering:

```
Device# monitor capture mycap interface GigabitEthernet1/0/3 in
Device# monitor capture mycap match ipv4 any any
Device# monitor capture mycap buffer circular size 1
Device# monitor capture mycap start
```

Step 2: Determine whether the capture is active by entering:

```
Device# show monitor capture mycap
Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/3, Direction: in
   Status : Active
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
```

```
  Buffer Details:
   Buffer Type: CIRCULAR
   Buffer Size (in MB): 1
  File Details:
   File not associated
  Limit Details:
   Number of Packets to capture: 0 (no limit)
   Packet Capture duration: 0 (no limit)
   Packet Size to capture: 0 (no limit)
   Maximum number of packets to capture per second: 1000
   Packet sampling rate: 0 (no sampling)
```

Step 3: Display extended capture statistics during runtime by entering:

```
Device# show monitor capture mycap capture-statistics
Capture statistics collected at software:
        Capture duration - 88 seconds
        Packets received - 1000
        Packets dropped - 0
        Packets oversized - 0
        Packets errored - 0
        Packets sent - 1000
        Bytes received - 182000
        Bytes dropped - 0
        Bytes oversized - 0
        Bytes errored - 0
        Bytes sent - 114000
```

Step 4: Stop the capture by entering:

```
Device# monitor capture mycap stop
Capture statistics collected at software (Buffer):
        Capture duration - 2185 seconds
        Packets received - 51500
        Packets dropped - 0
        Packets oversized - 0
```

Step 5: Display extended capture statistics after stop by entering:

```
Device# show monitor capture mycap capture-statistics
Capture statistics collected at software:
        Capture duration - 156 seconds
        Packets received - 2000
        Packets dropped - 0
        Packets oversized - 0
        Packets errored - 0
        Packets sent - 2000
        Bytes received - 364000
        Bytes dropped - 0
        Bytes oversized - 0
        Bytes errored - 0
        Bytes sent - 228000
```

Step 6: Determine whether the capture is active by entering:

```
Device# show monitor capture mycap
Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/3, Direction: in
   Status : Inactive
  Filter Details:
   IPv4
    Source IP:  any
```

```
    Destination IP:  any
   Protocol: any
  Buffer Details:
   Buffer Type: CIRCULAR
   Buffer Size (in MB): 1
  File Details:
   File not associated
  Limit Details:
   Number of Packets to capture: 0 (no limit)
   Packet Capture duration: 0 (no limit)
   Packet Size to capture: 0 (no limit)
   Maximum number of packets to capture per second: 1000
   Packet sampling rate: 0 (no sampling)
```

Step 7: Display the packets in the buffer by entering:

```
Device# show monitor capture mycap buffer brief
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1   0.000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40057/31132, ttl=254
  2   0.000030   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40058/31388, ttl=254
  3   0.000052   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40059/31644, ttl=254
  4   0.000073   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40060/31900, ttl=254
  5   0.000094   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40061/32156, ttl=254
  6   0.000115   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40062/32412, ttl=254
  7   0.000137   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40063/32668, ttl=254
  8   0.000158   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40064/32924, ttl=254
  9   0.000179   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40065/33180, ttl=254
 10   0.000200   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40066/33436, ttl=254
 11   0.000221   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40067/33692, ttl=254
 12   0.000243   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0038,
seq=40068/33948, ttl=254
 --More--
```

Notice that the packets have been buffered.

Step 8: Display the packets in other display modes.

```
Device# show monitor capture mycap buffer detailed
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

Frame 1: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
    Interface id: 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov  6, 2015 18:10:06.297972000 UTC
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1446833406.297972000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 114 bytes (912 bits)
```

```
    Capture Length: 114 bytes (912 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:icmp:data]
Ethernet II, Src: Cisco_f3:63:46 (00:e1:6d:f3:63:46), Dst: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)

    Destination: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)
        Address: Cisco_31:f1:c6 (00:e1:6d:31:f1:c6)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: Cisco_f3:63:46 (00:e1:6d:f3:63:46)
        Address: Cisco_f3:63:46 (00:e1:6d:f3:63:46)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.2 (10.10.10.2), Dst: 10.10.10.1 (10.10.10.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not
ECN-Capable Transport))
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport)
(0x00)
    Total Length: 100
    Identification: 0xabdd (43997)
    Flags: 0x00
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 254
    Protocol: ICMP (1)
    Header checksum: 0xe8a4 [validation disabled]
        [Good: False]
        [Bad: False]
    Source: 10.10.10.2 (10.10.10.2)
    Destination: 10.10.10.1 (10.10.10.1)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xa620 [correct]
    Identifier (BE): 56 (0x0038)
    Identifier (LE): 14336 (0x3800)
    Sequence number (BE): 40057 (0x9c79)
    Sequence number (LE): 31132 (0x799c)
    Data (72 bytes)

0000  00 00 00 00 0b 15 30 63 ab cd ab cd ab cd ab cd   ......0c........
0010  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0020  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0030  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0040  ab cd ab cd ab cd ab cd                           ........
        Data: 000000000b153063abcdabcdabcdabcdabcdabcdabcdabcd...
        [Length: 72]

Frame 2: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0


Device# show monitor capture mycap buffer dump
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

0000  00 e1 6d 31 f1 c6 00 e1 6d f3 63 46 08 00 45 00   ..m1....m.cF..E.
0010  00 64 ab dd 00 00 fe 01 e8 a4 0a 0a 0a 02 0a 0a   .d..............
0020  0a 01 08 00 a6 20 00 38 9c 79 00 00 00 00 0b 15   ..... .8.y......
```

```
0030  30 63 ab cd ab cd ab cd ab cd ab cd ab cd ab cd   0c..............
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0050  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0060  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0070  ab cd                                            ..

0000  00 e1 6d 31 f1 c6 00 e1 6d f3 63 46 08 00 45 00   ..m1....m.cF..E.
0010  00 64 ab de 00 00 fe 01 e8 a3 0a 0a 0a 02 0a 0a   .d..............
0020  0a 01 08 00 a6 1d 00 38 9c 7a 00 00 00 00 0b 15   .......8.z......
0030  30 65 ab cd ab cd ab cd ab cd ab cd ab cd ab cd   0e..............
0040  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0050  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0060  ab cd ab cd ab cd ab cd ab cd ab cd ab cd ab cd   ................
0070  ab cd
```

Step 9: Clear the buffer by entering:

```
Device# monitor capture mycap clear
```

Note NOTE - Clearing the buffer deletes the buffer along with the contents.

Note If you require the buffer contents to be displayed, run the clear commands after show commands.

Step 10: Restart the traffic, wait for 10 seconds, then display the buffer contents by entering:

Note We cannot run show from buffer during an active capture. Capture should be stopped before running show from buffer. We can however run a show on a pcap file during an active capture in both file and buffer mode. In file mode, we can display the packets in the current capture session's pcap file as well when the capture is active.

```
Device# monitor capture mycap start
Switch#  show monitor capture mycap

Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/3, Direction: in
   Status : Active
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  Buffer Details:
   Buffer Type: CIRCULAR
   Buffer Size (in MB): 1
  File Details:
   File not associated
  Limit Details:
   Number of Packets to capture: 0 (no limit)
   Packet Capture duration: 0 (no limit)
   Packet Size to capture: 0 (no limit)
   Maximum number of packets to capture per second: 1000
```

```
   Packet sampling rate: 0 (no sampling)
```

Step 11: Stop the packet capture and display the buffer contents by entering:

```
Device# monitor capture mycap stop
Capture statistics collected at software (Buffer):
 Capture duration - 111 seconds
 Packets received - 5000
 Packets dropped - 0
 Packets oversized - 0
```

Step 12: Determine whether the capture is active by entering:

```
Device# show monitor capture mycap
Status Information for Capture mycap
 Target Type:
  Interface: GigabitEthernet1/0/3, Direction: in
 Status : Inactive
Filter Details:
 IPv4
  Source IP: any
  Destination IP: any
 Protocol: any
Buffer Details:
 Buffer Type: CIRCULAR
 Buffer Size (in MB): 1
File Details:
 File not associated
Limit Details:
 Number of Packets to capture: 0 (no limit)
 Packet Capture duration: 0 (no limit)
 Packet Size to capture: 0 (no limit)
 Maximum number of packets to capture per second: 1000
 Packet sampling rate: 0 (no sampling)
```

Step 13: Display the packets in the buffer by entering:

```
Device# show monitor capture mycap buffer brief
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1 0.000000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=0/0, ttl=254
  2 0.000030000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=1/256, ttl=254
  3 0.000051000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=2/512, ttl=254
  4 0.000072000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=3/768, ttl=254
  5 0.000093000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=4/1024, ttl=254
  6 0.000114000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=5/1280, ttl=254
  7 0.000136000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=6/1536, ttl=254
  8 0.000157000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=7/1792, ttl=254
  9 0.000178000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=8/2048, ttl=254
 10 0.000199000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=9/2304, ttl=254
 11 0.000220000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=10/2560, ttl=254
 12 0.000241000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=11/2816, ttl=254
```

```
 --More‹
```

Step 14: Store the buffer contents to the mycap.pcap file in the internal flash: storage device by entering:

```
Device# monitor capture mycap export flash:mycap.pcap
Exported Successfully
```

Note The current implementation of export is such that when the command is run, export is "started" but not complete when it returns the prompt to the user. So we have to wait for a message display on the console from Wireshark before it can run a display of packets in the file.

Step 15: Display capture packets from the file by entering:

```
Device#  show monitor capture file flash:mycap.pcap
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1 0.000000000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=0/0, ttl=254
  2 0.000030000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=1/256, ttl=254
  3 0.000051000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=2/512, ttl=254
  4 0.000072000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=3/768, ttl=254
  5 0.000093000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=4/1024, ttl=254
  6 0.000114000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=5/1280, ttl=254
  7 0.000136000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=6/1536, ttl=254
  8 0.000157000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=7/1792, ttl=254
  9 0.000178000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=8/2048, ttl=254
 10 0.000199000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=9/2304, ttl=254
 11 0.000220000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=10/2560, ttl=254
 12 0.000241000   10.10.10.2 -> 10.10.10.1   ICMP 114 Echo (ping) request  id=0x0039,
seq=11/2816, ttl=254
 --More--
```

Step 16: Delete the capture point by entering:

```
Device# no monitor capture mycap
```

Example: Simple Capture and Store of Packets in Egress Direction

This example shows how to capture packets to a filter:

Step 1: Define a capture point to match on the relevant traffic and associate it to a file by entering:

```
Device# monitor capture mycap interface Gigabit 1/0/1 out match ipv4 any any
Device# monitor capture mycap limit duration 60 packets 100
Device# monitor capture mycap file location flash:mycap.pcap buffer-size 90
```

Step 2: Confirm that the capture point has been correctly defined by entering:

```
Device# show monitor capture mycap parameter
   monitor capture mycap interface GigabitEthernet1/0/1 out
   monitor capture mycap match ipv4  any any
   monitor capture mycap file location flash:mycap.pcap buffer-size 90
   monitor capture mycap limit packets 100 duration 60

Device# show monitor capture mycap

Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/1, Direction: out
   Status : Inactive
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  Buffer Details:
   Buffer Type: LINEAR (default)
  File Details:
   Associated file name: flash:mycap.pcap
   Size of buffer(in MB): 90
  Limit Details:
   Number of Packets to capture: 100
   Packet Capture duration: 60
   Packet Size to capture: 0 (no limit)
   Packets per second: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
```

Step 3: Launch packet capture by entering:

```
Device# monitor capture mycap start
A file by the same capture file name already exists, overwrite?[confirm]
Turning on lock-step mode

Device#
*Oct 14 09:35:32.661: %BUFCAP-6-ENABLE: Capture Point mycap enabled.
```

Note Allow the capture operation stop automatically after the time has elapsed or the packet count has been met. When you see the following message in the output, will know that the capture operation has stopped:

```
*Oct 14 09:36:34.632: %BUFCAP-6-DISABLE_ASYNC: Capture Point mycap disabled. Rea
son : Wireshark Session Ended
```

The mycap.pcap file now contains the captured packets.

Step 4: Display the packets by entering:

```
Device# show monitor capture file flash:mycap.pcap
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  0.000000    10.1.1.30 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  1.000000    10.1.1.31 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  2.000000    10.1.1.32 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  3.000000    10.1.1.33 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  4.000000    10.1.1.34 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.000000    10.1.1.35 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  6.000000    10.1.1.36 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  7.000000    10.1.1.37 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

```
  8.000000    10.1.1.38 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  9.000000    10.1.1.39 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

Step 5: Delete the capture point by entering:

```
Device# no monitor capture mycap
```

Configuration Examples for Embedded Packet Capture

The following sections provide configuration examples for EPC.

Example: Managing Packet Data Capture

The following example shows how to manage packet data capture:

```
Device> enable
Device# monitor capture mycap access-list v4acl
Device# monitor capture mycap limit duration 1000
Device# monitor capture mycap interface GigabitEthernet 0/0/1 both
Device# monitor capture mycap buffer circular size 10
Device# monitor capture mycap start
Device# monitor capture mycap stop
Device# monitor capture mycap export tftp://10.1.88.9/mycap.pcap
Device# end
```

Example: Monitoring and Maintaining Captured Data

The following example shows how to dump packets in ASCII format:

```
Device# show monitor capture mycap buffer dump
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

0
0000: 01005E00 00020000 0C07AC1D 080045C0 ..^...........E.
0010: 00300000 00000111 CFDC091D 0002E000 .0..............
0020: 000207C1 07C1001C 802A0000 10030AFA .........*......
0030: 1D006369 73636F00 0000091D 0001 ..example.......
1
0000: 01005E00 0002001B 2BF69280 080046C0 ..^.....+.....F.
0010: 00200000 00000102 44170000 0000E000 . ......D.......
0020: 00019404 00001700 E8FF0000 0000 ..............
2
0000: 01005E00 0002001B 2BF68680 080045C0 ..^.....+.....E.
0010: 00300000 00000111 CFDB091D 0003E000 .0..............
0020: 000207C1 07C1001C 88B50000 08030A6E ...............n
0030: 1D006369 73636F00 0000091D 0001 ..example.......
3
0000: 01005E00 000A001C 0F2EDC00 080045C0 ..^...........E.
0010: 003C0000 00000258 CE7F091D 0004E000 .<.....X........
0020: 000A0205 F3000000 00000000 00000000 ................
0030: 00000000 00D10001 000C0100 01000000 ................
0040: 000F0004 00080501 0300
```

The following example shows how to display the list of commands used to configure the capture named mycap:

```
Device# show monitor capture mycap parameter
monitor capture mycap interface GigabitEthernet 1/0/1 both
monitor capture mycap match any
```

```
monitor capture mycap buffer size 10
monitor capture mycap limit pps 1000
```

The following example shows how to debug the capture point:

```
Device# debug epc capture-point
EPC capture point operations debugging is on

Device# monitor capture mycap start
*Jun 4 14:17:15.463: EPC CP: Starting the capture cap1
*Jun 4 14:17:15.463: EPC CP: (brief=3, detailed=4, dump=5) = 0
*Jun 4 14:17:15.463: EPC CP: final check before activation
*Jun 4 14:17:15.463: EPC CP: setting up c3pl infra
*Jun 4 14:17:15.463: EPC CP: Setup c3pl acl-class-policy
*Jun 4 14:17:15.463: EPC CP: Creating a class
*Jun 4 14:17:15.464: EPC CP: Creating a class : Successful
*Jun 4 14:17:15.464: EPC CP: class-map Created
*Jun 4 14:17:15.464: EPC CP: creating policy-name epc_policy_cap1
*Jun 4 14:17:15.464: EPC CP: Creating Policy epc_policy_cap1 of type 49 and client type 21
*Jun 4 14:17:15.464: EPC CP: Storing a Policy
*Jun 4 14:17:15.464: EPC CP: calling ppm_store_policy with epc_policy
*Jun 4 14:17:15.464: EPC CP: Creating Policy : Successful
*Jun 4 14:17:15.464: EPC CP: policy-map created
*Jun 4 14:17:15.464: EPC CP: creating filter for ANY
*Jun 4 14:17:15.464: EPC CP: Adding acl to class : Successful
*Jun 4 14:17:15.464: EPC CP: Setup c3pl class to policy
*Jun 4 14:17:15.464: EPC CP: Attaching Class to Policy
*Jun 4 14:17:15.464: EPC CP: Attaching epc_class_cap1 to epc_policy_cap1
*Jun 4 14:17:15.464: EPC CP: Attaching Class to Policy : Successful
*Jun 4 14:17:15.464: EPC CP: setting up c3pl qos
*Jun 4 14:17:15.464: EPC CP: DBG> Set packet rate limit to 1000
*Jun 4 14:17:15.464: EPC CP: creating action for policy_map epc_policy_cap1 class_map
epc_class_cap1
*Jun 4 14:17:15.464: EPC CP: DBG> Set packet rate limit to 1000
*Jun 4 14:17:15.464: EPC CP: Activating Interface GigabitEthernet1/0/1 direction both
*Jun 4 14:17:15.464: EPC CP: Id attached 0
*Jun 4 14:17:15.464: EPC CP: inserting into active lists
*Jun 4 14:17:15.464: EPC CP: Id attached 0
*Jun 4 14:17:15.465: EPC CP: inserting into active lists
*Jun 4 14:17:15.465: EPC CP: Activating Vlan
*Jun 4 14:17:15.465: EPC CP: Deleting all temp interfaces
*Jun 4 14:17:15.465: %BUFCAP-6-ENABLE: Capture Point cap1 enabled.
*Jun 4 14:17:15.465: EPC CP: Active Capture 1

Device# monitor capture mycap1 stop
*Jun 4 14:17:31.963: EPC CP: Stopping the capture cap1
*Jun 4 14:17:31.963: EPC CP: Warning: unable to unbind capture cap1
*Jun 4 14:17:31.963: EPC CP: Deactivating policy-map
*Jun 4 14:17:31.963: EPC CP: Policy epc_policy_cap1
*Jun 4 14:17:31.964: EPC CP: Deactivating policy-map Successful
*Jun 4 14:17:31.964: EPC CP: removing povision feature
*Jun 4 14:17:31.964: EPC CP: Found action for policy-map epc_policy_cap1 class-map
epc_class_cap1
*Jun 4 14:17:31.964: EPC CP: cleanning up c3pl infra
*Jun 4 14:17:31.964: EPC CP: Removing Class epc_class_cap1 from Policy
*Jun 4 14:17:31.964: EPC CP: Removing Class from epc_policy_cap1
*Jun 4 14:17:31.964: EPC CP: Successfully removed
*Jun 4 14:17:31.964: EPC CP: Removing acl mac from class
*Jun 4 14:17:31.964: EPC CP: Removing acl from class : Successful
*Jun 4 14:17:31.964: EPC CP: Removing all policies
*Jun 4 14:17:31.964: EPC CP: Removing Policy epc_policy_cap1
*Jun 4 14:17:31.964: EPC CP: Removing Policy : Successful
*Jun 4 14:17:31.964: EPC CP: Removing class epc_class_cap1
*Jun 4 14:17:31.965: EPC CP: Removing class : Successful
```

```
*Jun 4 14:17:31.965: %BUFCAP-6-DISABLE: Capture Point cap1 disabled.
*Jun 4 14:17:31.965: EPC CP: Active Capture 0
```

The following example shows how to debug the Embedded Packet Capture (EPC) provisioning:

```
Device# debug epc provision
EPC provisionioning debugging is on

Device# monitor capture mycap start
*Jun 4 14:17:54.991: EPC PROV: No action found for policy-map epc_policy_cap1 class-map
epc_class_cap1
*Jun 4 14:17:54.991: EPC PROV:
*Jun 4 14:17:54.991: Attempting to install service policy epc_policy_cap1
*Jun 4 14:17:54.992: EPC PROV: Attached service policy to epc idb subblock
*Jun 4 14:17:54.992: EPC PROV: Successful. Create feature object
*Jun 4 14:17:54.992: EPC PROV:
*Jun 4 14:17:54.992: Attempting to install service policy epc_policy_cap1
*Jun 4 14:17:54.992: EPC PROV: Successful. Create feature object
*Jun 4 14:17:54.992: %BUFCAP-6-ENABLE: Capture Point cap1 enabled.

Device# monitor capture mycap stop
*Jun 4 14:18:02.503: EPC PROV: Successful. Remove feature object
*Jun 4 14:18:02.504: EPC PROV: Successful. Remove feature object
*Jun 4 14:18:02.504: EPC PROV: Destroyed epc idb subblock
*Jun 4 14:18:02.504: EPC PROV: Found action for policy-map epc_policy_cap1 class-map
epc_class_cap1
*Jun 4 14:18:02.504: EPC PROV: Deleting EPC action
*Jun 4 14:18:02.504: EPC PROV: Successful. CLASS_REMOVE, policy-map epc_policy_cap1, class
 epc_class_cap1
*Jun 4 14:18:02.504: %BUFCAP-6-DISABLE: Capture Point cap1 disabled.
```

Additional References

Related Documents

Related Topic	Document Title
Display Filters	For syntax of Display Filters, refer to: Display Filter Reference
Pcap file statistics	For syntax used to display pcap file statistics, refer to "-z" option details at: Tshark Command Reference

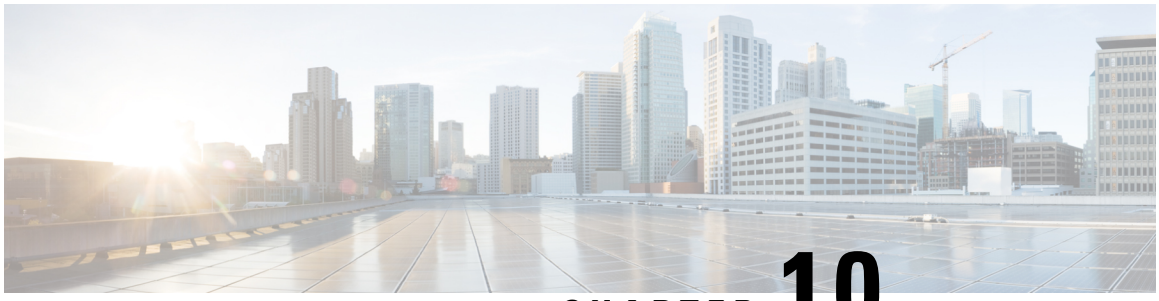
Feature History for Configuring Packet Capture

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Table 11: Feature History for Configuring Packet Capture

Releases	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	Configuring Packet Capture	The Packet Capture feature is an onboard packet capture facility that allows network administrators to capture packets flowing to, through, and from the device.



CHAPTER 10

Configuring Flexible NetFlow

- Prerequisites for Flexible NetFlow, on page 171
- Restrictions for Flexible NetFlow, on page 172
- Information About Flexible Netflow, on page 174
- How to Configure Flexible Netflow, on page 189
- Monitoring Flexible NetFlow, on page 200
- Configuration Examples for Flexible NetFlow, on page 200
- Additional References for NetFlow, on page 204
- Feature Information for Flexible NetFlow, on page 204

Prerequisites for Flexible NetFlow

The following are prerequisites for your Flexible NetFlow configuration:

- You must configure a source interface. If you do not configure a source interface, the exporter remains in a disabled state.
- You must configure a valid record name for every flow monitor.
- You must enable IPv6 routing to export the flow records to an IPv6 destination server.
- You must configure IPFIX export protocol for the flow exporter to export netflow records in IPFIX format.
- You are familiar with the Flexible NetFlow key fields as they are defined in the following commands in the Cisco IOS Flexible NetFlow Command Reference:
  - **match datalink**—Datalink (layer2) fields
  - **match flow**—Flow identifying fields
  - **match interface**—Interface fields
  - **match ipv4**—IPv4 fields
  - **match ipv6**—IPv6 fields
  - **match transport**—Transport layer fields
  - **match flow cts**—CTS fields

- You are familiar with the Flexible NetFlow non-key fields as they are defined in the following commands in the Cisco IOS Flexible NetFlow Command Reference :
  - **collect counter**—Counter fields
  - **collect flow**—Flow identifying fields
  - **collect interface**—Interface fields
  - **collect timestamp**—Timestamp fields
  - **collect transport**—Transport layer fields

IPv4 Traffic

- The networking device must be configured for IPv4 routing.
- One of the following must be enabled on your device and on any interfaces on which you want to enable Flexible NetFlow: Cisco Express Forwarding or distributed Cisco Express Forwarding.

IPv6 Traffic

- The networking device must be configured for IPv6 routing.
- One of the following must be enabled on your device and on any interfaces on which you want to enable Flexible NetFlow: Cisco Express Forwarding IPv6 or distributed Cisco Express Forwarding.

Restrictions for Flexible NetFlow

The following are restrictions for Flexible NetFlow:

- Flexible Netflow Version 5 Export Protocol and Autonomous System Number feature is not supported on the C9500-12Q, C9500-16X, C9500-24Q, C9500-40X models of the Cisco Catalyst 9500 Series Switches.
- Flexible NetFlow is not supported on the Layer 2 port-channel interface, but is supported on the Layer 2 port-channel member ports.
- Flexible NetFlow is not supported on the Layer 3 port-channel interface, but is supported on the Layer 3 port-channel member ports.
- Traditional NetFlow accounting is not supported.
- Flexible NetFlow Version 9 and Version 10 export formats are supported. However, if you have not configured the export protocol, Version 9 export format is applied by default.
- Flexible NetFlow and NBAR cannot be configured together at the same time on the same interface.
- Layer 2, IPv4, and IPv6 traffic types are supported. Multiple flow monitors of different traffic types can be applied for a given interface and direction. Multiple flow monitors of same traffic type cannot be applied for a given interface and direction.
- The device does not support tunnels and SVI interfaces; however Layer 2 and Layer 3 physical interfaces and VLAN configuration mode are supported.

- The following NetFlow table sizes are supported:

Trim Level	Ingress NetFlow Table	Egress NetFlow Table
Network Essentials	32 K	32 K
Network Advantage	32 K	32 K

- Depending on the switch type, a switch will have one or two forwarding ASICs. The capacities listed in the above table are on a per-Core/per-ASIC basis.
- The switch can support upto four ASICs. Each ASIC has two cores. Each core has 32K ingress and 32K egress entries, whereas each TCAM can handle up to 1024 ingress and 2048 egress entries.
- The NetFlow tables are on separate compartments and cannot be combined. Depending on which core processed the packet, the flows will be created in the table in the corresponding core.
- NetFlow hardware implementation supports four hardware samplers. You can select a sampler rate from 1 out of 2 to 1 out of 1024. Both — random and deterministic — sampling modes are supported.
- NetFlow hardware uses hash tables internally. Hash collisions can occur in the hardware. Therefore, in spite of the internal overflow Content Addressable Memory (CAM), the actual NetFlow table utilization could be about 80 percent.
- Depending on the fields that are used for the flow, a single flow could take two consecutive entries. IPv6 and datalink flows also take two entries. In these situations, the effective usage of NetFlow entries is half the table size, which is separate from the above hash collision limitation.
- The device supports up to 15 flow monitors.
- The NetFlow software implementation supports distributed NetFlow export, so the flows are exported from the same device in which the flow was created.
- Ingress flows are present in the ASIC that first received the packets for the flow. Egress flows are present in the ASIC from which the packets actually left the device set up.
- The reported value for the bytes count field (called “bytes long”) is Layer-2-packet-size—18 bytes. For classic Ethernet traffic (802.3), this will be accurate. For all other Ethernet types, this field will not be accurate. Use the "bytes layer2” field, which always reports the accurate Layer 2 packet size. For information about supported Flexible NetFlow fields, see 'Supported Flexible NetFlow Fields' topic.
- Configuration of IPFIX exporter on an AVC flow monitor is not supported.
- Flexible NetFlow export is not supported on the Ethernet management port, GigabitEthernet 0/0.
- When a flow record has only Source Group Tag (SGT) and Destination Group Tag (DGT) fields (or only either of the two) and if both the values are not applicable, then a flow will still be created with zero values for SGT and DGT. The flow records are expected to include source and destination IP addresses, along with SGT and DGT fields.
- On non-Cisco TrustSec interfaces, an SGT value of zero implies that there is no command header. On Cisco TrustSec interfaces, an SGT value of zero implies an unknown tag.
- For an IPv6 flow monitor, Source Group Tag (SGT) and Destination Group Tag (DGT) fields cannot co-exist with MAC address fields.

- When a quality of service (QoS) marked packet is received on an interface which has NetFlow configured in the ingress direction, the QoS value of the packet is captured by the NetFlow collector. However, when the packet is received on an interface which has NetFlow configured in the egress direction and the QoS value has been rewritten on ingress by the switch, the new QoS value of the packet is not captured by the collector.
- NetFlow records do not support MultiProtocol Label Switching-enabled (MPLS-enabled) interfaces.
- Data capture based on MPLS label inside the MPLS network is not supported. Capture of IP header fields of an MPLS tagged packet is not supported.
- Egress flow monitors do not capture flows that are egressing out in EoMPLS mode or in L3VPN Per-Prefix mode.
- A flow monitor cannot be shared across Layer 3 physical interfaces and logical interfaces (such as, Layer 3 port-channel interface, Layer 3 port-channel member, and switch virtual interface [SVI]), but a flow monitor can be shared across logical interfaces or Layer 3 physical interfaces.

Information About Flexible Netflow

Flexible NetFlow Overview

Flexible NetFlow uses flows to provide statistics for accounting, network monitoring, and network planning.

A flow is a unidirectional stream of packets that arrives on a source interface and has the same values for the keys. A key is an identified value for a field within the packet. You create a flow using a flow record to define the unique keys for your flow.

The device supports the Flexible NetFlow feature that enables enhanced network anomalies and security detection. Flexible NetFlow allows you to define an optimal flow record for a particular application by selecting the keys from a large collection of predefined fields.

All key values must match for the packet to count in a given flow. A flow might gather other fields of interest, depending on the export record version that you configure. Flows are stored in the Flexible NetFlow cache.

You can export the data that Flexible NetFlow gathers for your flow by using an exporter and export this data to a remote system such as a Flexible NetFlow collector. The Flexible NetFlow collector can use an IPv4 address.

You define the size of the data that you want to collect for a flow using a monitor. The monitor combines the flow record and exporter with the Flexible NetFlow cache information.

Original NetFlow and Benefits of Flexible NetFlow

Original NetFlow uses a fixed seven tuples of IP information to identify a flow.

Flexible NetFlow allows the flow to be user defined. The benefits of Flexible NetFlow include:

- High-capacity flow recognition, including scalability and aggregation of flow information.
- Enhanced flow infrastructure for security monitoring and dDoS detection and identification.

- New information from packets to adapt flow information to a particular service or operation in the network. The flow information available will be customizable by Flexible NetFlow users.
- Extensive use of Cisco's flexible and extensible NetFlow Version 9 and version 10 export formats. With version 10 export format, support for variable length field for the wireless client's SSID is available.
- A comprehensive IP accounting feature that can be used to replace many accounting features, such as IP accounting, Border Gateway Protocol (BGP) Policy Accounting, and persistent caches.

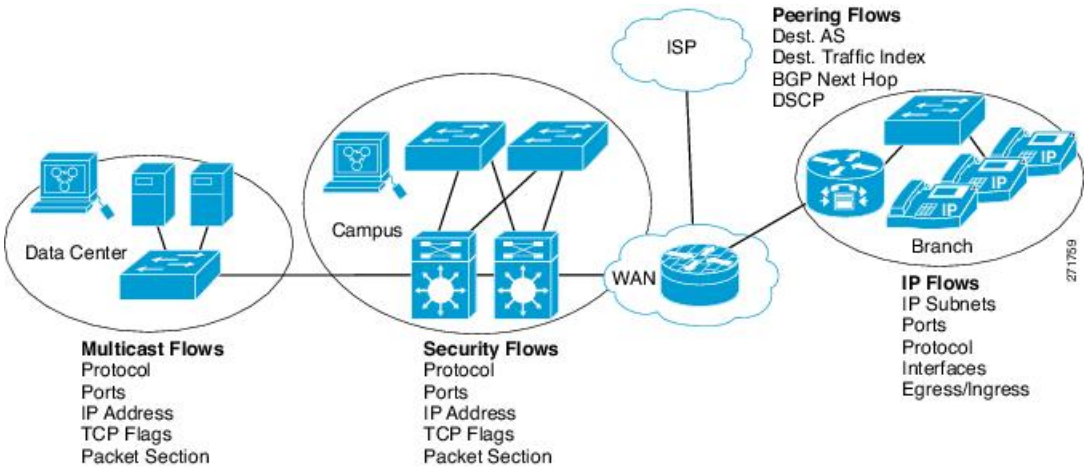
Original NetFlow allows you to understand the activities in the network and thus to optimize network design and reduce operational costs.

Flexible NetFlow allows you to understand network behavior with more efficiency, with specific flow information tailored for various services used in the network. The following are some example applications for a Flexible NetFlow feature:

- Flexible NetFlow enhances Cisco NetFlow as a security monitoring tool. For instance, new flow keys can be defined for packet length or MAC address, allowing users to search for a specific type of attack in the network.
- Flexible NetFlow allows you to quickly identify how much application traffic is being sent between hosts by specifically tracking TCP or UDP applications by the class of service (CoS) in the packets.
- The accounting of traffic entering a Multiprotocol Label Switching (MPLS) or IP core network and its destination for each next hop per class of service. This capability allows the building of an edge-to-edge traffic matrix.

The figure below is an example of how Flexible NetFlow might be deployed in a network.

Figure 9: Typical Deployment for Flexible NetFlow



Flexible NetFlow Components

Flexible NetFlow consists of components that can be used together in several variations to perform traffic analysis and data export. The user-defined flow records and the component structure of Flexible NetFlow facilitates the creation of various configurations for traffic analysis and data export on a networking device with a minimum number of configuration commands. Each flow monitor can have a unique combination of flow record, flow exporter, and cache type. If you change a parameter such as the destination IP address for

a flow exporter, it is automatically changed for all the flow monitors that use the flow exporter. The same flow monitor can be used in conjunction with different flow samplers to sample the same type of network traffic at different rates on different interfaces. The following sections provide more information on Flexible NetFlow components:

Flow Records

In Flexible NetFlow a combination of key and nonkey fields is called a record. Flexible NetFlow records are assigned to Flexible NetFlow flow monitors to define the cache that is used for storing flow data. Flexible NetFlow includes several predefined records that can help you get started using Flexible NetFlow.

A flow record defines the keys that Flexible NetFlow uses to identify packets in the flow, as well as other fields of interest that Flexible NetFlow gathers for the flow. You can define a flow record with any combination of keys and fields of interest. The device supports a rich set of keys. A flow record also defines the types of counters gathered per flow. You can configure 64-bit packet or byte counters. The device enables the following match fields as the defaults when you create a flow record:

- **match datalink**—Layer 2 attributes
- **match flow direction**—Specifies a match to the fields identifying the direction of flow.
- **match interface**—Interface attributes
- **match ipv4**—IPv4 attributes
- **match ipv6**—IPv6 attributes
- **match transport**—Transport layer fields
- **match flow cts**—Cisco TrustSec fields

NetFlow Predefined Records

Flexible NetFlow includes several predefined records that you can use to start monitoring traffic in your network. The predefined records are available to help you quickly deploy Flexible NetFlow and are easier to use than user-defined flow records. You can choose from a list of already defined records that may meet the needs for network monitoring. As Flexible NetFlow evolves, popular user-defined flow records will be made available as predefined records to make them easier to implement.

Note Predefined records are not supported for regular Flexible NetFlow on Cisco Catalyst 9000 Series Switches.

User-Defined Records

Flexible NetFlow enables you to define your own records for a Flexible NetFlow flow monitor cache by specifying the key and nonkey fields to customize the data collection to your specific requirements. When you define your own records for a Flexible NetFlow flow monitor cache, they are referred to as *user-defined records*. The values in nonkey fields are added to flows to provide additional information about the traffic in the flows. A change in the value of a nonkey field does not create a new flow. In most cases the values for nonkey fields are taken from only the first packet in the flow. Flexible NetFlow enables you to capture counter values such as the number of bytes and packets in a flow as nonkey fields.

You can create user-defined records for applications such as QoS and bandwidth monitoring, application and end user traffic profiling, and security monitoring for dDoS attacks. Flexible NetFlow also includes several

predefined records that emulate original NetFlow. Flexible NetFlow user-defined records provide the capability to monitor a contiguous section of a packet of a user-configurable size, and use it in a flow record as a key or a nonkey field along with other fields and attributes of the packet. The section may include any Layer 3 data from the packet. The packet section fields allow the user to monitor any packet fields that are not covered by the Flexible NetFlow predefined keys. The ability to analyze packet fields that are not collected with the predefined keys enables more detailed traffic monitoring, facilitates the investigation of dDoS attacks, and enables implementation of other security applications such as URL monitoring.

Flexible NetFlow provides predefined types of packet sections of a user-configurable size. The following Flexible NetFlow commands (used in Flexible NetFlow flow record configuration mode) can be used to configure the predefined types of packet sections:

- **collect ipv4 section header size** *bytes* --Starts capturing the number of bytes specified by the *bytes*argument from the beginning of the IPv4 header of each packet.
- **collect ipv4 section payload size** *bytes* --Starts capturing bytes immediately after the IPv4 header from each packet. The number of bytes captured is specified by the *bytes* argument.
- **collect ipv6 section header size** *bytes* --Starts capturing the number of bytes specified by the *bytes*argument from the beginning of the IPv6 header of each packet.
- **collect ipv6 section payload size** *bytes* --Starts capturing bytes immediately after the IPv6 header from each packet. The number of bytes captured is specified by the *bytes* argument.

The *bytes* values are the sizes in bytes of these fields in the flow record. If the corresponding fragment of the packet is smaller than the requested section size, Flexible NetFlow will fill the rest of the section field in the flow record with zeros. If the packet type does not match the requested section type, Flexible NetFlow will fill the entire section field in the flow record with zeros.

Flexible NetFlow adds a new Version 9 export format field type for the header and packet section types. Flexible NetFlow will communicate to the NetFlow collector the configured section sizes in the corresponding Version 9 export template fields. The payload sections will have a corresponding length field that can be used to collect the actual size of the collected section.

Flexible NetFlow Match Parameters

The following table describes Flexible NetFlow match parameters. You must configure at least one of the following match parameters for the flow records.

Table 12: Match Parameters

Command	Purpose
match datalink {**dot1q** | **ethertype** | **mac** | **vlan** }	Specifies a match to datalink or Layer 2 fields. The following command options are available: • **dot1q**—Matches to the dot1q field. • **ethertype**—Matches to the ethertype of the packet. • **mac**—Matches the source or destination MAC fields. • **vlan**—Matches to the VLAN that the packet is located on (input or output).

Command	Purpose
match flow direction	Specifies a match to the flow identifying fields.
match interface {**input** | **output**}	Specifies a match to the interface fields. The following command options are available: • **input**—Matches to the input interface. • **output**—Matches to the output interface.
match ipv4 {**destination** | **protocol** | **source** | **tos** | **ttl** | **version**}	Specifies a match to the IPv4 fields. The following command options are available: • **destination**—Matches to the IPv4 destination address-based fields. • **protocol**—Matches to the IPv4 protocols. • **source**—Matches to the IPv4 source address based fields. • **tos**—Matches to the IPv4 Type of Service fields. • **ttl**—Matches to the IPv4 Time To Live fields. • **version**—Matches to the IP version from the IPv4 header.
match ipv6 {**destination** | **hop-limit** | **protocol** | **source** | **traffic-class** | **version** }	Specifies a match to the IPv6 fields. The following command options are available: • **destination**—Matches to the IPv6 destination address-based fields. • **hop-limit**—Matches to the IPv6 hop limit fields. • **protocol**—Matches to the IPv6 payload protocol fields. • **source**—Matches to the IPv6 source address based fields. • **traffic-class**—Matches to the IPv6 traffic class. • **version**—Matches to the IP version from the IPv6 header.

Command	Purpose
match transport {**destination-port** | **igmp** | **icmp** | **source-port**}	Specifies a match to the Transport Layer fields. The following command options are available: • **destination-port**—Matches to the transport destination port. • **icmp**—Matches to ICMP fields, including ICMP IPv4 and IPv6 fields. • **igmp**—Matches to IGMP fields. • **source-port**—Matches to the transport source port.
match flow cts {source | destination} group-tag	Specifies a match to the CTS fields support in FNF record. The following command options are available: • **source** —Matches to the source of CTS entering the domain. • **destination** —Matches to the destination of the CTS leaving the domain.

Flexible NetFlow Collect Parameters

The following table describes the Flexible NetFlow collect parameters.

Table 13: Collect Parameters

Command	Purpose
collect counter { **bytes** { **layer2** { **long** } | **long** } | **packets** { **long** } }	Collects the counter fields total bytes and total packets.
collect interface {**input** | **output**}	Collects the fields from the input or output interface.
collect timestamp absolute {**first** | **last**}	Collects the fields for the absolute time the first packet was seen or the absolute time the most recent packet was last seen (in milliseconds).

Command	Purpose
collect transport tcp flags	Collects the following transport TCP flags: • **ack**—TCP acknowledgement flag • **cwr**—TCP congestion window reduced flag • **ece**—TCP ECN echo flag • **fin**—TCP finish flag • **psh**—TCP push flag • **rst**—TCP reset flag • **syn**—TCP synchronize flag • **urg**—TCP urgent flag **Note** On the device, you cannot specify which TCP flag to collect. You can only specify to collect transport TCP flags. All TCP flags will be collected with this command.
collect counter bytes	Configures the number of bytes seen in a flow as a nonkey field and enables collecting the total number of bytes from the flow.
collect counter packets	Configures the number of packets seen in a flow as a nonkey field and enables collecting the total number of packets from the flow.

Flow Exporters

Flow exporters export the data in the flow monitor cache to a remote system, such as a server running NetFlow collector, for analysis and storage. Flow exporters are created as separate entities in the configuration. Flow exporters are assigned to flow monitors to provide data export capability for the flow monitors. You can create several flow exporters and assign them to one or more flow monitors to provide several export destinations. You can create one flow exporter and apply it to several flow monitors.

NetFlow Data Export Format Version 9

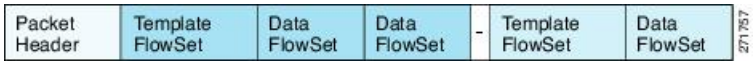
The basic output of NetFlow is a flow record. Several different formats for flow records have evolved as NetFlow has matured. The most recent evolution of the NetFlow export format is known as Version 9. The distinguishing feature of the NetFlow Version 9 export format is that it is template-based. Templates provide an extensible design to the record format, a feature that should allow future enhancements to NetFlow services without requiring concurrent changes to the basic flow-record format. Using templates provides several key benefits:

- Third-party business partners who produce applications that provide collector or display services for NetFlow do not have to recompile their applications each time a new NetFlow feature is added. Instead, they should be able to use an external data file that documents the known template formats.

- New features can be added to NetFlow quickly without breaking current implementations.
- NetFlow is "future-proofed" against new or developing protocols because the Version 9 format can be adapted to provide support for them.

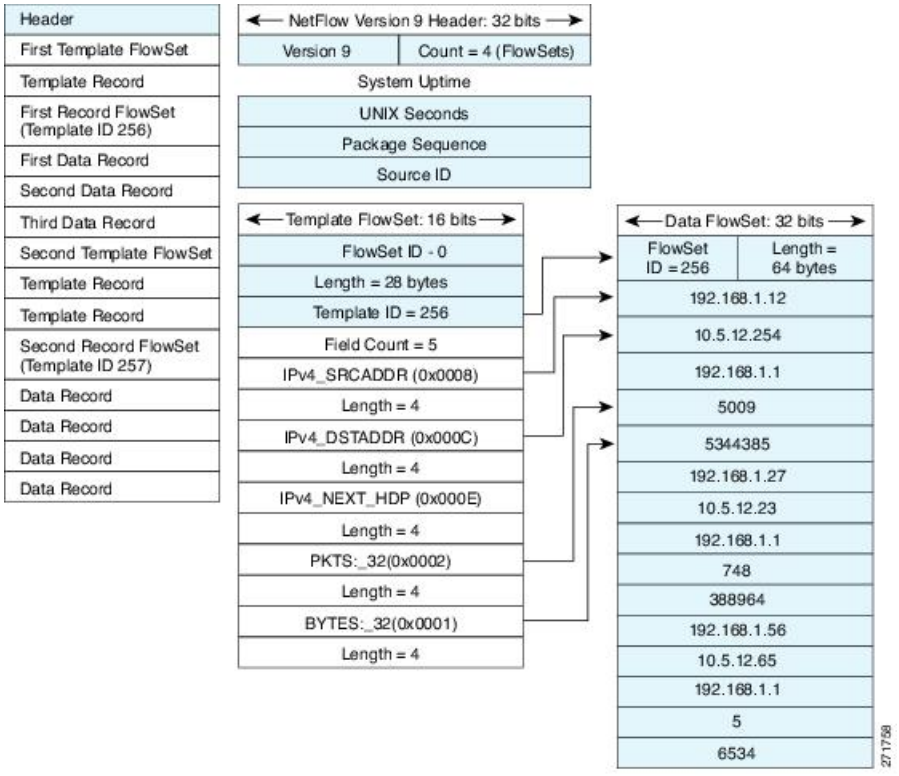
The Version 9 export format consists of a packet header followed by one or more template flow or data flow sets. A template flow set provides a description of the fields that will be present in future data flow sets. These data flow sets may occur later within the same export packet or in subsequent export packets. Template flow and data flow sets can be intermingled within a single export packet, as illustrated in the figure below.

Figure 10: Version 9 Export Packet



NetFlow Version 9 will periodically export the template data so the NetFlow collector will understand what data is to be sent and also export the data flow set for the template. The key advantage to Flexible NetFlow is that the user configures a flow record, which is effectively converted to a Version 9 template and then forwarded to the collector. The figure below is a detailed example of the NetFlow Version 9 export format, including the header, template flow, and data flow sets.

Figure 11: Detailed Example of the NetFlow Version 9 Export Format



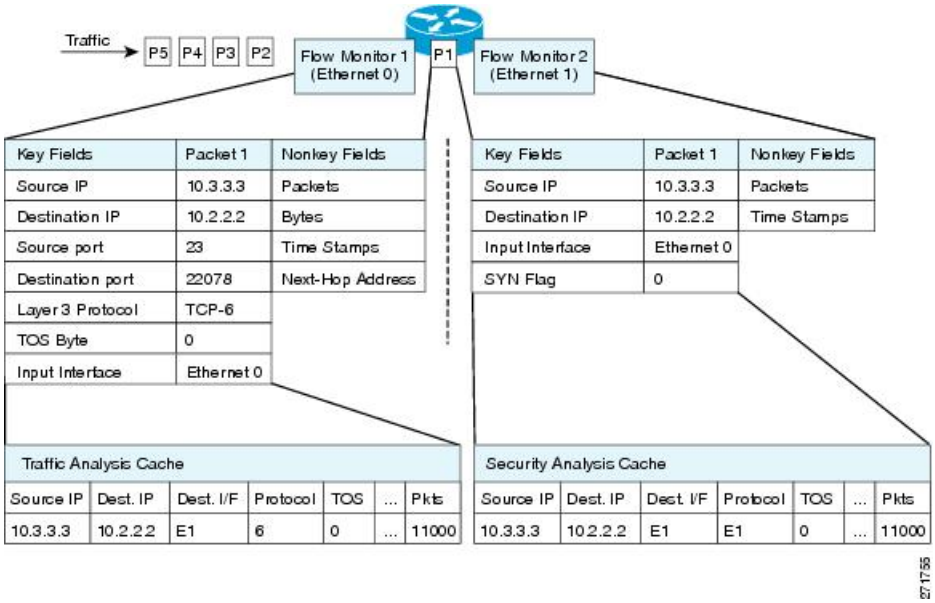
Flow Monitors

Flow monitors are the Flexible NetFlow component that is applied to interfaces to perform network traffic monitoring.

Flow data is collected from the network traffic and added to the flow monitor cache during the monitoring process based on the key and nonkey fields in the flow record.

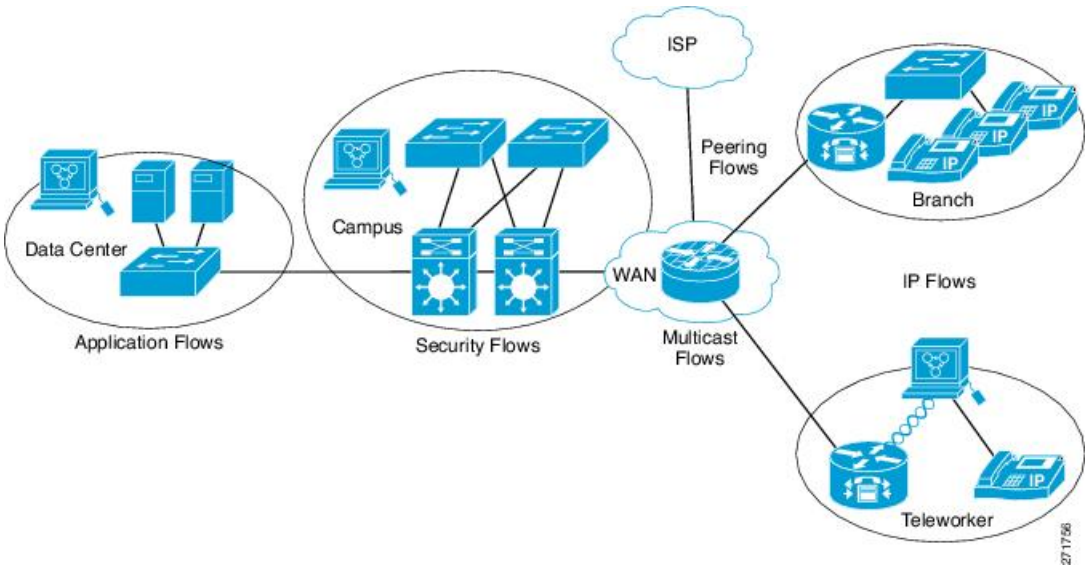
Flexible NetFlow can be used to perform different types of analysis on the same traffic. In the figure below, packet 1 is analyzed using a record designed for standard traffic analysis on the input interface and a record designed for security analysis on the output interface.

Figure 12: Example of Using Two Flow Monitors to Analyze the Same Traffic



The figure below shows a more complex example of how you can apply different types of flow monitors with custom records.

Figure 13: Complex Example of Using Multiple Types of Flow Monitors with Custom Records



There are three types of flow monitor caches. You change the type of cache used by the flow monitor after you create the flow monitor. The three types of flow monitor caches are described in the following sections:

Normal

The default cache type is “normal”. In this mode, the entries in the cache are aged out according to the timeout active and timeout inactive settings. When a cache entry is aged out, it is removed from the cache and exported via any exporters configured.

Immediate

A cache of type "immediate" ages out every record as soon as it is created. As a result, every flow contains just one packet. The commands that display the cache contents will provide a history of the packets seen.

This mode is desirable when you expect only very small flows and you want a minimum amount of latency between seeing a packet and exporting a report.

Caution This mode may result in a large amount of export data that can overload low-speed links and overwhelm any systems that you are exporting to. We recommended that you configure sampling to reduce the number of packets that are processed.

Note The cache timeout settings have no effect in this mode.

Permanent

A cache of type "permanent" never ages out any flows. A permanent cache is useful when the number of flows you expect to see is low and there is a need to keep long-term statistics on the router. For example, if the only key field in the flow record is the 8-bit IP ToS field, only 256 flows can be monitored. To monitor the long-term usage of the IP ToS field in the network traffic, you can use a permanent cache. Permanent caches are useful for billing applications and for an edge-to-edge traffic matrix for a fixed set of flows that are being tracked. Update messages will be sent periodically to any flow exporters configured according to the "timeout update" setting.

Note When a cache becomes full in permanent mode, new flows will not be monitored. If this occurs, a "Flows not added" message will appear in the cache statistics.

Note A permanent cache uses update counters rather than delta counters. This means that when a flow is exported, the counters represent the totals seen for the full lifetime of the flow and not the additional packets and bytes seen since the last export was sent.

Flow Samplers

Flow samplers are created as separate components in a router’s configuration. Flow samplers are used to reduce the load on the device that is running Flexible NetFlow by limiting the number of packets that are selected for analysis.

Flow sampling exchanges monitoring accuracy for router performance. When you apply a sampler to a flow monitor, the overhead load on the router of running the flow monitor is reduced because the number of packets that the flow monitor must analyze is reduced. The reduction in the number of packets that are analyzed by the flow monitor causes a corresponding reduction in the accuracy of the information stored in the flow monitor's cache.

Samplers are combined with flow monitors when they are applied to an interface with the **ip flow monitor** command.

Supported Flexible NetFlow Fields

The following tables provide a consolidated list of supported fields in Flexible NetFlow (FNF) for various traffic types and traffic direction.

Note If the packet has a VLAN field, then that length is not accounted for.

Field	Layer 2 In	Layer 2 Out	IPv4 In	IP v4 Out	IPv6 In	IPv6 Out	Notes
Key or Collect Fields							
Interface input	Yes	—	Yes	—	Yes	—	If you apply a flow monitor in the input direction: • Use the **match** keyword and use the input interface as a key field. • Use the **collect** keyword and use the output interface as a collect field. This field will be present in the exported records but with a value of 0.

Field	Layer 2 In	Layer 2 Out	IPv4 In	IP v4 Out	IPv6 In	IPv6 Out	Notes
Interface output	—	Yes	—	Yes	—	Yes	If you apply a flow monitor in the output direction: • Use the **match** keyword and use the output interface as a key field. • Use the **collect** keyword and use the input interface as a collect field. This field will be present in the exported records but with a value of 0.

Field	Layer 2 In	Layer 2 Out	IPv4 In	IP v4 Out	IPv6 In	IPv6 Out	Notes
Key Fields							
Flow direction	Yes	Yes	Yes	Yes	Yes	Yes	
Ethertype	Yes	Yes	—	—	—	—	
VLAN input	Yes	—	Yes	—	Yes	—	Supported only for a switch port.
VLAN output	—	Yes	—	Yes	—	Yes	Supported only for a switch port.
dot1q VLAN input	Yes	—	Yes	—	Yes	—	Supported only for a switch port.
dot1q VLAN output	—	Yes	—	Yes	—	Yes	Supported only for a switch port.
dot1q priority	Yes	Yes	Yes	Yes	Yes	Yes	Supported only for a switch port.
MAC source address input	Yes	Yes	Yes	Yes	Yes	Yes	

Field	Layer 2 In	Layer 2 Out	IPv4 In	IP v4 Out	IPv6 In	IPv6 Out	Notes
MAC source address output	—	—	—	—	—	—	
MAC destination address input	Yes	—	Yes	—	Yes	—	
MAC destination address output	—	Yes	—	Yes	—	Yes	
IPv4 version	—	—	Yes	Yes	Yes	Yes	
IPv4 TOS	—	—	Yes	Yes	Yes	Yes	
IPv4 protocol	—	—	Yes	Yes	Yes	Yes	Must use if any of src/dest port, ICMP code/type, IGMP type or TCP flags are used.
IPv4 TTL	—	—	Yes	Yes	Yes	Yes	
IPv4 source address	—	—	Yes	Yes	—	—	
IPv4 destination address	—	—	Yes	Yes	—	—	
ICMP IPv4 type	—	—	Yes	Yes	—	—	
ICMP IPv4 code	—	—	Yes	Yes	—	—	
IGMP type	—	—	Yes	Yes	—	—	

Field	Layer 2 In	Layer 2 Out	IPv4 In	IP v4 Out	IPv6 In	IPv6 Out	Notes
Key Fields continued							

Field	Layer 2 In	Layer 2 Out	IPv4 In	IP v4 Out	IPv6 In	IPv6 Out	Notes
IPv6 version	—	—	Yes	Yes	Yes	Yes	Same as IP version.
IPv6 protocol	—	—	Yes	Yes	Yes	Yes	Same as IP protocol. Must use if any of src/dest port, ICMP code/type, IGMP type or TCP flags are used.
IPv6 source address	—	—	—	—	Yes	Yes	
IPv6 destination address	—	—	—	—	Yes	Yes	
IPv6 traffic-class	—	—	Yes	Yes	Yes	Yes	Same as IP TOS.
IPv6 hop-limit	—	—	Yes	Yes	Yes	Yes	Same as IP TTL.
ICMP IPv6 type	—	—	—	—	Yes	Yes	
ICMP IPv6 code	—	—	—	—	Yes	Yes	
source-port	—	—	Yes	Yes	Yes	Yes	
dest-port	—	—	Yes	Yes	Yes	Yes	

Field	Layer 2 In	Layer 2 Out	IPv4 In	IP v4 Out	IPv6 In	IPv6 Out	Notes
Collect Fields							

Field	Layer 2 In	Layer 2 Out	IPv4 In	IP v4 Out	IPv6 In	IPv6 Out	Notes
Bytes long	Yes	Yes	Yes	Yes	Yes	Yes	Packet size = (Ethernet frame size including FCS - 18 bytes) **Recommended:** Avoid this field and use Bytes layer2 long.
Packets long	Yes	Yes	Yes	Yes	Yes	Yes	
Timestamp absolute first	Yes	Yes	Yes	Yes	Yes	Yes	
Timestamp absolute last	Yes	Yes	Yes	Yes	Yes	Yes	
TCP flags	Yes	Yes	Yes	Yes	Yes	Yes	Collects all flags.
Bytes layer2 long	Yes	Yes	Yes	Yes	Yes	Yes	

Default Settings

The following table lists the Flexible NetFlow default settings for the device.

Table 14: Default Flexible NetFlow Settings

Setting	Default
Flow active timeout	1800 seconds
Flow timeout inactive	15 seconds

Flexible NetFlow—Ingress VRF Support Overview

The Flexible NetFlow—Ingress VRF Support feature enables collecting the virtual routing and forwarding (VRF) ID from incoming packets on a device by applying an input flow monitor having a flow record that collects the VRF ID as a key field.

How to Configure Flexible Netflow

To configure Flexible Netflow, follow these general steps:

1. Create a flow record by specifying keys and non-key fields to the flow.
2. Create an optional flow exporter by specifying the protocol and transport destination port, destination, and other parameters.
3. Create a flow monitor based on the flow record and flow exporter.
4. Create an optional sampler.
5. Apply the flow monitor to a Layer 2 port, Layer 3 port, or VLAN.

Creating a Customized Flow Record

Perform this task to configure a customized flow record.

Customized flow records are used to analyze traffic data for a specific purpose. A customized flow record must have at least one **match** criterion for use as the key field and typically has at least one **collect** criterion for use as a nonkey field.

There are hundreds of possible permutations of customized flow records. This task shows the steps that are used to create one of the possible permutations. Modify the steps in this task as appropriate to create a customized flow record for your requirements.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**flow record** *record-name* **Example:** `Device(config)# flow record FLOW-RECORD-1`	Creates a flow record and enters Flexible NetFlow flow record configuration mode. • This command also allows you to modify an existing flow record.
Step 4	**description** *description* **Example:**	(Optional) Creates a description for the flow record.

	Command or Action	Purpose
	`Device(config-flow-record)# description Used for basic traffic analysis`	
Step 5	**match** {**ip** | **ipv6**} {**destination** | **source**} **address** **Example:** `Device(config-flow-record)# match ipv4 destination address`	Configures a key field for the flow record. **Note** This example configures the IPv4 destination address as a key field for the record. For information about the other key fields available for the **match ipv4** command, and the other **match** commands that are available to configure key fields.
Step 6	Repeat Step 5 as required to configure additional key fields for the record.	—

Step 7	**match flow cts** {**source** | **destination**} **group-tag**
	Example:
	`Device(config-flow-record)# match flow cts source group-tag`
	`Device(config-flow-record)# match flow cts destination group-tag`
		Note This example configures the CTS source group tag and destination group tag as a key field for the record. For information about the other key fields available for the **match ipv4** command, and the other **match** commands that are available to configure key fields.
		Note
		• Ingress:
		  • In an incoming packet, if a header is present, SGT will reflect the same value as the header. If no value is present, it will show zero.
		  • The DGT value will not depend on the ingress port SGACL configuration.
		• Egress:
		  • If either propagate SGT or CTS is disabled on the egress interface, then SGT will be zero.
		  • In an outgoing packet, if SGACL configuration that corresponds to the (SGT, DGT) exists, DGT will be non-zero.
		  • If SGACL is disabled on the egress port/VLAN or if global SGACL enforcement is disabled, then DGT will be zero

	Command or Action	Purpose
Step 8	**Example:**	Configures the input interface as a nonkey field for the record. **Note** This example configures the input interface as a nonkey field for the record.
Step 9	Repeat the above step as required to configure additional nonkey fields for the record.	—
Step 10	**end** **Example:** `Device(config-flow-record)# end`	Exits Flexible NetFlow flow record configuration mode and returns to privileged EXEC mode.
Step 11	**show flow record** *record-name* **Example:** `Device# show flow record FLOW_RECORD-1`	(Optional) Displays the current status of the specified flow record.
Step 12	**show running-config flow record** *record-name* **Example:** `Device# show running-config flow record FLOW_RECORD-1`	(Optional) Displays the configuration of the specified flow record.

Creating a Flow Exporter

You can create a flow export to define the export parameters for a flow.

Note Each flow exporter supports only one destination. If you want to export the data to multiple destinations, you must configure multiple flow exporters and assign them to the flow monitor.

You can export to a destination using IPv4 address.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**flow exporter** *name* **Example:** `Device(config)# flow exporter ExportTest`	Creates a flow exporter and enters flow exporter configuration mode.
Step 3	**description** *string* **Example:** `Device(config-flow-exporter)# description ExportV9`	(Optional) Describes this flow record as a maximum 63-character string.
Step 4	**destination** {*ipv4-address*} **Example:** `Device(config-flow-exporter)# destination 192.0.2.1 (IPv4 destination)`	Sets the IPv4 destination address or hostname for this exporter.
Step 5	**dscp** *value* **Example:** `Device(config-flow-exporter)# dscp 0`	(Optional) Specifies the differentiated services codepoint value. The range is from 0 to 63. The default is 0.
Step 6	**source** { | } **Example:** `Device(config-flow-exporter)# source gigabitEthernet1/0/1`	(Optional) Specifies the interface to use to reach the NetFlow collector at the configured destination. The following interfaces can be configured as source: \
Step 7	**transport udp** *number* **Example:** `Device(config-flow-exporter)# transport udp 200`	(Optional) Specifies the UDP port to use to reach the NetFlow collector.
Step 8	**ttl** *seconds* **Example:** `Device(config-flow-exporter)# ttl 210`	(Optional) Configures the time-to-live (TTL) value for datagrams sent by the exporter. The range is from 1 to 255 seconds. The default is 255.

	Command or Action	Purpose
Step 9	**export-protocol** {**netflow-v9**} **Example:** `Device(config-flow-exporter)# export-protocol netflow-v9`	Specifies the version of the NetFlow export protocol used by the exporter.
Step 10	**end** **Example:** `Device(config-flow-record)# end`	Returns to privileged EXEC mode.
Step 11	**show flow exporter** [**name** *record-name*] **Example:** `Device# show flow exporter ExportTest`	(Optional) Displays information about NetFlow flow exporters.
Step 12	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

Define a flow monitor based on the flow record and flow exporter.

Creating a Customized Flow Monitor

Perform this required task to create a customized flow monitor.

Each flow monitor has a separate cache assigned to it. Each flow monitor requires a record to define the contents and layout of its cache entries. These record formats can be one of the predefined formats or a user-defined format. An advanced user can create a customized format using the **flow record** command.

Note When Flexible NetFlow is configured on a Layer 3 port-channel interface, the last applied flow monitor configuration takes effect across all members of that port channel. Therefore, we recommend that you must have the same flow monitor configuration on all members of a L3 port-channel interface.

Before you begin

If you want to use a customized record instead of using one of the Flexible NetFlow predefined records, you must create the customized record before you can perform this task. If you want to add a flow exporter to the flow monitor for data export, you must create the exporter before you can complete this task.

Note You must use the **no ip flow monitor** command to remove a flow monitor from all of the interfaces to which you have applied it before you can modify the parameters for the **record** command on the flow monitor.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**flow monitor** *monitor-name* **Example:** `Device(config)# flow monitor FLOW-MONITOR-1`	Creates a flow monitor and enters Flexible NetFlow flow monitor configuration mode. • This command also allows you to modify an existing flow monitor.
Step 4	**description** *description* **Example:** `Device(config-flow-monitor)# description Used for basic ipv4 traffic analysis`	(Optional) Creates a description for the flow monitor.
Step 5	**record** {*record-name* | **netflow-original** | **netflow** {**ipv4** | **ipv6**} *record* [**peer**]} **Example:** `Device(config-flow-monitor)# record FLOW-RECORD-1`	Specifies the record for the flow monitor.

Step 6	**cache** {**timeout** {**active** | **inactive** | **update**} *seconds* | **type normal** }
	Example:
	`Device(config-flow-monitor)# cache type normal`
	`Device(config-flow-monitor)# cache timeout active`
		(Optional) Modifies the flow monitor cache parameters such as timeout values, and the cache type. Associates a flow cache with the specified flow monitor.

Step 7	Repeat Step 6 as required to finish modifying the cache parameters for this flow monitor.	—

	Command or Action	Purpose
Step 8	**statistics packet protocol** **Example:** `Device(config-flow-monitor)# statistics packet protocol`	(Optional) Enables the collection of protocol distribution statistics for Flexible NetFlow monitors.
Step 9	**statistics packet size** **Example:** `Device(config-flow-monitor)# statistics packet size`	(Optional) Enables the collection of size distribution statistics for Flexible NetFlow monitors.
Step 10	**exporter** *exporter-name* **Example:** `Device(config-flow-monitor)# exporter EXPORTER-1`	(Optional) Specifies the name of an exporter that was created previously.
Step 11	**end** **Example:** `Device(config-flow-monitor)# end`	Exits Flexible NetFlow flow monitor configuration mode and returns to privileged EXEC mode.
Step 12	**show flow monitor** [[**name**] *monitor-name* [**cache** [**format** {**csv** | **record** | **table**}]] [**statistics**]] **Example:** `Device# show flow monitor FLOW-MONITOR-2 cache`	(Optional) Displays the status and statistics for a Flexible NetFlow flow monitor.
Step 13	**show running-config flow monitor** *monitor-name* **Example:** `Device# show running-config flow monitor FLOW_MONITOR-1`	(Optional) Displays the configuration of the specified flow monitor.
Step 14	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Creating a Flow Sampler

Perform this required task to configure and enable a flow sampler.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**sampler** *sampler-name* **Example:** `Device(config)# sampler SAMPLER-1`	Creates a sampler and enters sampler configuration mode. • This command also allows you to modify an existing sampler.
Step 4	**description** *description* **Example:** `Device(config-sampler)# description Sample at 50%`	(Optional) Creates a description for the flow sampler.
Step 5	**mode** {**random**} **1 out-of** *window-size* **Example:** `Device(config-sampler)# mode random 1 out-of 2`	Specifies the sampler mode and the flow sampler window size. • The range for the *window-size* argument is from 0 to 1024.
Step 6	**exit** **Example:** `Device(config-sampler)# exit`	Exits sampler configuration mode and returns to global configuration mode.
Step 7	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 0/0/0`	Specifies an interface and enters interface configuration mode.
Step 8	{**ip** | **ipv6**} **flow monitor** *monitor-name* [[**sampler**] *sampler-name*] {**input** | **output**} **Example:** `Device(config-if)# ip flow monitor FLOW-MONITOR-1 sampler SAMPLER-1 input`	Assigns the flow monitor and the flow sampler that you created to the interface to enable sampling.

	Command or Action	Purpose
Step 9	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 10	**show sampler sampler-name** **Example:** `Device# show sampler SAMPLER-1`	Displays the status and statistics of the flow sampler that you configured and enabled.

Applying a Flow to an Interface

You can apply a flow monitor and an optional sampler to an interface.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device(config)# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type* **Example:** `Device(config)# interface GigabitEthernet1/0/1`	Enters interface configuration mode and configures an interface. Flexible NetFlow is not supported on the L2 port-channel interface, but is supported on the L2 port-channel member ports. Flexible NetFlow is not supported on the L3 port-channel interface, but is supported on the L3 port-channel member ports.
Step 4	{**ip flow monitor** | **ipv6 flow monitor** | **datalink flow monitor**} *name* [**sampler** *name*] {**input** | **output**} **Example:** `Device(config-if)# ip flow monitor MonitorTest input`	Associates an IPv4, IPv6 and datalink flow monitor, and an optional sampler to the interface for input or output packets. **ip flow monitor** – Enables Flexible NetFlow to monitor IPv4 traffic. **ipv6 flow monitor** – Enables Flexible NetFlow to monitor IPv6 traffic. **datalink flow monitor** – Enables Flexible NetFlow to monitor non-IP traffic.

	Command or Action	Purpose
		Note You can associate multiple monitors to an interface in both input and output directions.
Step 5	**end** **Example:** `Device(config-flow-monitor)# end`	Returns to privileged EXEC mode.
Step 6	**show flow interface** [*interface-type number*] **Example:** `Device# show flow interface`	(Optional) Displays information about NetFlow on an interface.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring a Bridged NetFlow on a VLAN

You can apply a flow monitor and an optional sampler to a VLAN.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**vlan** [**configuration**] *vlan-id*
	Example:
	`Device(config)# vlan configuration 30`
	`Device(config-vlan-config)#`
		Enters VLAN or VLAN configuration mode.

Step 3	**ip flow monitor** *monitor name* [**sampler** *sampler name*] {**input** } **Example:**	Associates a flow monitor and an optional sampler to the VLAN for input packets.

	Command or Action	Purpose
	`Device(config-vlan-config)# ip flow monitor MonitorTest input`	
Step 4	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Layer 2 NetFlow

You can define Layer 2 keys in Flexible NetFlow records that you can use to capture flows in Layer 2 interfaces.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**flow record** *name*
	Example:
	`Device(config)# flow record L2_record`
	`Device(config-flow-record)#`
		Enters flow record configuration mode.

Step 3	**match datalink** {**dot1q** |**ethertype** | **mac** | **vlan**} **Example:** `Device(config-flow-record)# match datalink ethertype`	Specifies the Layer 2 attribute as a key.
Step 4	**end** **Example:** `Device(config-flow-record)# end`	Returns to privileged EXEC mode.
Step 5	**show flow record** [*name*] **Example:**	(Optional) Displays information about NetFlow on an interface.

	Command or Action	Purpose
	`Device# show flow record`	
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring Flexible NetFlow

The commands in the following table can be used to monitor Flexible NetFlow.

Table 15: Flexible NetFlow Monitoring Commands

Command	Purpose
show flow exporter [**broker** | **export-ids** | **name** | *name* | **statistics** | **templates**]	Displays information about NetFlow flow exporters and statistics.
show flow exporter [**name** *exporter-name*]	Displays information about NetFlow flow exporters and statistics.
show flow interface	Displays information about NetFlow interfaces.
show flow monitor [**name** *exporter-name*]	Displays information about NetFlow flow monitors and statistics.
show flow monitor statistics	Displays the statistics for the flow monitor
show flow monitor cache format {**table** | **record** | **csv**}	Displays the contents of the cache for the flow monitor, in the format specified.
show flow record [**name** *record-name*]	Displays information about NetFlow flow records.
show sampler [**broker** | **name** | *name*]	Displays information about NetFlow samplers.

Configuration Examples for Flexible NetFlow

Example: Configuring a Flow

This example shows how to create a flow and apply it to an interface:

```
Device# configure terminal
```

```
Enter configuration commands, one per line. End with CNTL/Z.

Device(config)# flow export export1
Device(config-flow-exporter)# destination 10.0.101.254
Device(config-flow-exporter)# transport udp 2055
Device(config-flow-exporter)# exit
Device(config)# flow record record1
Device(config-flow-record)# match ipv4 source address
Device(config-flow-record)# match ipv4 destination address
Device(config-flow-record)# match ipv4 protocol
Device(config-flow-record)# match transport source-port
Device(config-flow-record)# match transport destination-port
Device(config-flow-record)# match flow cts source group-tag
Device(config-flow-record)# match flow cts destination group-tag
Device(config-flow-record)# collect counter byte long
Device(config-flow-record)# collect counter packet long
Device(config-flow-record)# collect timestamp absolute first
Device(config-flow-record)# collect timestamp absolute last
Device(config-flow-record)# exit
Device(config)# flow monitor monitor1
Device(config-flow-monitor)# record record1
Device(config-flow-monitor)# exporter export1
Device(config-flow-monitor)# exit
Device(config)# interface tenGigabitEthernet 1/0/1
Device(config-if)# ip flow monitor monitor1 input
Device(config-if)# end
```

Example: Monitoring IPv4 ingress traffic

This example shows how to monitor IPv4 ingress traffic (int g1/0/11 sends traffic to int g1/0/36 and int g3/0/11).

```
Device# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Device(config)# flow record  fr-1
Device(config-flow-record)# match ipv4 source address
Device(config-flow-record)# match ipv4 destination address
Device(config-flow-record)# match interface input
Device(config-flow-record)# collect counter bytes long
Device(config-flow-record)# collect counter packets long
Device(config-flow-record)# collect timestamp absolute first
Device(config-flow-record)# collect timestamp absolute last
Device(config-flow-record)# collect counter bytes layer2 long
Device(config-flow-record)# exit

Device(config)# flow exporter fe-ipfix6
Device(config-flow-exporter)# destination 2001:0:0:24::10
Device(config-flow-exporter)# source Vlan106
Device(config-flow-exporter)# transport udp 4739
Device(config-flow-exporter)# export-protocol ipfix
Device(config-flow-exporter)# template data timeout 240
Device(config-flow-exporter)# exit

Device(config)# flow exporter fe-ipfix
Device(config-flow-exporter)# description IPFIX format collector 100.0.0.80
Device(config-flow-exporter)# destination 100.0.0.80
Device(config-flow-exporter)# dscp 30
Device(config-flow-exporter)# ttl 210
```

```
Device(config-flow-exporter)# transport udp 4739
Device(config-flow-exporter)# export-protocol ipfix
Device(config-flow-exporter)# template data timeout 240
Device(config-flow-exporter)# exit

Device(config)# flow exporter fe-1
Device(config-flow-exporter)# destination 10.5.120.16
Device(config-flow-exporter)# source Vlan105
Device(config-flow-exporter)# dscp 32
Device(config-flow-exporter)# ttl 200
Device(config-flow-exporter)# transport udp 2055

Device(config-flow-exporter)# template data timeout 240
Device(config-flow-exporter)# exit

Device(config)# flow monitor fm-1
Device(config-flow-monitor)# exporter fe-ipfix6
Device(config-flow-monitor)# exporter fe-ipfix
Device(config-flow-monitor)# exporter fe-1
Device(config-flow-monitor)# cache timeout inactive 60
Device(config-flow-monitor)#  cache timeout active 180
Device(config-flow-monitor)# record fr-1
Device(config-flow-monitor)# end

Device# show running-config interface g1/0/11
Device# show running-config interface g1/0/36
Device# show running-config interface g3/0/11
Device# show flow monitor fm-1 cache format table
```

Example: Monitoring IPv4 egress traffic

```
Device# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Device(config)# flow record  fr-1 out
Device(config-flow-record)# match ipv4 source address
Device(config-flow-record)# match ipv4 destination address
Device(config-flow-record)# match interface output
Device(config-flow-record)# collect counter bytes long
Device(config-flow-record)# collect counter packets long
Device(config-flow-record)# collect timestamp absolute first
Device(config-flow-record)# collect timestamp absolute last
Device(config-flow-record)# exit

Device(config)# flow exporter fe-1
Device(config-flow-exporter)# destination 10.5.120.16
Device(config-flow-exporter)# source Vlan105
Device(config-flow-exporter)# dscp 32
Device(config-flow-exporter)# ttl 200
Device(config-flow-exporter)# transport udp 2055
Device(config-flow-exporter)# template data timeout 240
Device(config-flow-exporter)# exit

Device(config)# flow exporter fe-ipfix6
Device(config-flow-exporter)# destination 2001:0:0:24::10
Device(config-flow-exporter)# source Vlan106
Device(config-flow-exporter)# transport udp 4739
Device(config-flow-exporter)# export-protocol ipfix
Device(config-flow-exporter)# template data timeout 240
```

```
Device(config-flow-exporter)# exit

Device(config)# flow exporter fe-ipfix
Device(config-flow-exporter)# description IPFIX format collector 100.0.0.80
Device(config-flow-exporter)# destination 100.0.0.80
Device(config-flow-exporter)# dscp 30
Device(config-flow-exporter)# ttl 210
Device(config-flow-exporter)# transport udp 4739
Device(config-flow-exporter)# export-protocol ipfix
Device(config-flow-exporter)# template data timeout 240
Device(config-flow-exporter)# exit

Device(config)# flow monitor fm-1-output
Device(config-flow-monitor)# exporter fe-1
Device(config-flow-monitor)# exporter fe-ipfix6
Device(config-flow-monitor)# exporter fe-ipfix
Device(config-flow-monitor)# cache timeout inactive 50
Device(config-flow-monitor)#  cache timeout active 120
Device(config-flow-monitor)# record fr-1-out
Device(config-flow-monitor)# end

Device# show flow monitor fm-1-output cache format table
```

Example: Configuring Flexible NetFlow for Ingress VRF Support

The following example configures the collection of theVRF ID from incoming packets on a device by applying an input flow monitor having a flow record that collects the VRF ID as a key field.

```
Device> enable
Device# configure terminal
Device(config)# flow record rm_1
Device(config-flow-record)# match routing vrf input
Device(config-flow-record)# match ipv4 source address
Device(config-flow-record)# match ipv4 destination address
Device(config-flow-record)# collect interface input
Device(config-flow-record)# collect interface output
Device(config-flow-record)# collect counter packets
Device(config-flow-record)# exit

Device(config)# flow monitor mm_1
Device(config-flow-record)# record rm_1
Device(config-flow-record)# exit

Device(config)# interface GigabitEthernet 1/0/1
Device(config-if)# ip vrf forwarding green
Device(config-if)# ip address 172.16.2.2 255.255.255.252
Device(config-if)# ip flow monitor mm_1 input
Device(config-if)# end
```

Additional References for NetFlow

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter	*Command Reference (Catalyst 9500 Series Switches)*

Standards and RFCs

Standard/RFC	Title
RFC 3954	Cisco Systems NetFlow Services Export Version 9

MIBs

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

Technical Assistance

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

Feature Information for Flexible NetFlow

Release	Modification
Cisco IOS XE Everest 16.5.1a	This feature was introduced.