Network Management Configuration Guide, Cisco IOS XE 17.16.x (Catalyst 9600 Switches)

First Published: 2024-08-14

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

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2024 Cisco Systems, Inc. All rights reserved.

Preface

This preface describes the conventions of this document and information on how to obtain other documentation. It also provides information on what's new in Cisco product documentation.

- Document Conventions , on page iii
- Related Documentation, on page v
- Obtaining Documentation and Submitting a Service Request, on page v

Document Conventions

This document uses the following conventions:

Convention	Description
^ or Ctrl	Both the ^ symbol and Ctrl represent the Control (Ctrl) key on a keyboard. For example, the key combination **^D** or **Ctrl-D** means that you hold down the Control key while you press the D key. (Keys are indicated in capital letters but are not case sensitive.)
bold font	Commands and keywords and user-entered text appear in **bold** font.
Italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
`Courier` font	Terminal sessions and information the system displays appear in `courier` font.
`Bold Courier` font	**`Bold Courier`** font indicates text that the user must enter.
[x]	Elements in square brackets are optional.
...	An ellipsis (three consecutive nonbolded periods without spaces) after a syntax element indicates that the element can be repeated.
|	A vertical line, called a pipe, indicates a choice within a set of keywords or arguments.
[x | y]	Optional alternative keywords are grouped in brackets and separated by vertical bars.

Convention	Description
{x | y}	Required alternative keywords are grouped in braces and separated by vertical bars.
[x {y | z}]	Nested set of square brackets or braces indicate optional or required choices within optional or required elements. Braces and a vertical bar within square brackets indicate a required choice within an optional element.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
< >	Nonprinting characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Reader Alert Conventions

This document may use the following conventions for reader alerts:

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the manual.

Tip Means *the following information will help you solve a problem.*

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Timesaver Means *the described action saves time.* You can save time by performing the action described in the paragraph.

Take note of the following general safety warnings:

Warning

Statement 1071—Warning Definition

IMPORTANT SAFETY INSTRUCTIONS

Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Read the installation instructions before using, installing, or connecting the system to the power source. Use the statement number at the beginning of each warning statement to locate its translation in the translated safety warnings for this device.

SAVE THESE INSTRUCTIONS

Related Documentation

Note Before installing or upgrading the device, refer to the device release notes.

- Cisco Catalyst 9600 Series Switches documentation, located at:

  https://www.cisco.com/c/en/us/products/switches/catalyst-9600-series-switches/index.html

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/c/en/us/td/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.

CONTENTS

PREFACE

Preface iii

Document Conventions iii

Related Documentation v

Obtaining Documentation and Submitting a Service Request v

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

Configuring a Platform Type Filter 13

Configuring a Platform Type Filter for a Class Map 14

Configuring a Platform Type Filter for a Parameter Map 14

Configuring a Device Type Filter for a Class Map 15

Configuration Examples for Autoconf 16

Example: Applying a Built-In Template to an End Device 17

Example: Applying a Modified Built-In Template to an End Device 17

Example: Migrating from ASP Macros to Autoconf 17

Example: Configuring a Platform Type Filter 17
Additional References for Autoconf 18
Feature History for Autoconf 18

CHAPTER 2 **Configuring Interface Templates** 21
Restrictions for Interface Templates 21
Information About Interface Templates 21
Interface Template Overview 21
Binding an Interface Template to a Target 23
Priority for Configurations Using Interface Templates 24
How to Configure Interface Templates 24
Configuring Interface Templates 24
Configuring Static Binding for Interface Templates 25
Configuring Dynamic Binding of Interface Templates 26
Verifying Interface Templates 28
Configuration Examples for Interface Templates 32
Example: Configuring User Interface Templates 32
Example: Sourcing Interface Templates 32
Example: Dynamically Binding Interface Templates 33
Feature History for Interface Templates 33

CHAPTER 3 **Configuring Cisco Plug and Play** 35
Configuring Cisco Plug and Play 35
Feature History for Cisco Plug and Play 35

CHAPTER 4 **Configuring Cisco Discovery Protocol** 37
Information about Cisco Discovery Protocol 37
Default Cisco Discovery Protocol Configuration 37
Cisco Discovery Protocol Overview 37
How to Configure Cisco Discovery Protocol 38
Configuring Cisco Discovery Protocol Characteristics 38
Disabling Cisco Discovery Protocol 40
Enabling Cisco Discovery Protocol 41
Disabling Cisco Discovery Protocol on an Interface 42

Enabling Cisco Discovery Protocol on an Interface 43
Monitoring and Maintaining Cisco Discovery Protocol 44
Feature History for Cisco Discovery Protocol 45

CHAPTER 5 **Configuring Simple Network Management Protocol 47**
Prerequisites for SNMP 47
Restrictions for SNMP 49
Information About SNMP 49
SNMP Overview 50
SNMP Manager Functions 50
SNMP Agent Functions 51
SNMP Community Strings 51
SNMP MIB Variables Access 51
SNMP Flash MIB 52
SNMP Notifications 52
SNMP ifIndex MIB Object Values 52
SNMP ENTITY-MIB Identifiers 53
SNMP and Syslog Over IPv6 53
Default SNMP Configuration 53
SNMP Configuration Guidelines 54
How to Configure SNMP 55
SNMP Community Strings 55
Configuring SNMP Groups and Users 55
Opening or Closing SNMP UDP Ports 59
SNMP Notifications 60
Setting the Agent Contact and Location Information 60
Limiting TFTP Servers Used Through SNMP 61
Disabling the SNMP Agent 62
SNMP Examples 64
Monitoring SNMP Status 65
Feature History for Simple Network Management Protocol 65

CHAPTER 6 **Configuring Cisco IOS IP Service Level Agreements 67**
Restrictions on Service Level Agreements 67

Information About Service Level Agreements 67
Cisco IOS IP Service Level Agreements (SLAs) 67
Network Performance Measurement with Cisco IOS IP SLAs 69
IP SLA Responder and IP SLA Control Protocol 69
Response Time Computation for IP SLAs 70
IP SLAs Operation Scheduling 71
IP SLA Operation Threshold Monitoring 71
UDP Jitter 72
How to Configure IP SLAs Operations 73
Default Configuration 73
Configuration Guidelines 73
Configuring the IP SLA Responder 74
Implementing IP SLA Network Performance Measurement 75
Analyzing IP Service Levels by Using the UDP Jitter Operation 79
Analyzing IP Service Levels by Using the ICMP Echo Operation 82
Monitoring IP SLA Operations 86
Monitoring IP SLA Operation Examples 86
Additional References 87
Feature History for Service Level Agreements 88

CHAPTER 7 **Configuring SPAN and RSPAN** 89
Prerequisites for SPAN and RSPAN 89
Restrictions for SPAN and RSPAN 89
Information About SPAN and RSPAN 91
SPAN and RSPAN 91
Local SPAN 92
Remote SPAN 93
SPAN and RSPAN Concepts and Terminology 94
SPAN and RSPAN Interaction with Other Features 99
SPAN and RSPAN and Device Stacks 100
Flow-Based SPAN 100
Default SPAN and RSPAN Configuration 101
Configuring SPAN and RSPAN 101
SPAN Configuration Guidelines 101

RSPAN Configuration Guidelines 101
FSPAN and FRSPAN Configuration Guidelines 102
How to Configure SPAN and RSPAN 102
Creating a Local SPAN Session 102
Creating a Local SPAN Session and Configuring Incoming Traffic 105
Specifying VLANs to Filter 107
Configuring a VLAN as an RSPAN VLAN 109
Creating an RSPAN Source Session 110
Specifying VLANs to Filter 112
Creating an RSPAN Destination Session 114
Creating an RSPAN Destination Session and Configuring Incoming Traffic 116
Configuring an FSPAN Session 118
Configuring an FRSPAN Session 121
Monitoring SPAN and RSPAN Operations 124
Configuration Examples for SPAN and RSPAN 124
Example: Configuring Local SPAN 124
Examples: Creating an RSPAN VLAN 125
Feature History for SPAN and RSPAN 126

CHAPTER 8 **Configuring ERSPAN** 129
Prerequisites for Configuring ERSPAN 129
Restrictions for Configuring ERSPAN 129
Information about Configuring ERSPAN 130
ERSPAN Overview 130
ERSPAN Sources 132
ERSPAN Destination Ports 132
SGT Based ERSPAN 132
ERSPAN Timestamp 133
ERSPAN over MPLS VPN 133
How to Configure ERSPAN 133
Configuring an ERSPAN Source Session (IPv4) 133
Configuring an ERSPAN Destination Session (IPv4) 136
Configuring an ERSPAN Source Session (IPv6) 138
Configuring an ERSPAN Destination Session (IPv6) 141

Configuration examples for ERSPAN 143
Example: Configuring an ERSPAN Source Session 143
Example: Configuring an ERSPAN Destination Session 143
Verifying ERSPAN 144
Additional References 146
Feature History for ERSPAN 147

CHAPTER 9 **Configuring Packet Capture** 149
Prerequisites for Configuring Packet Capture 149
Prerequisites for Configuring Wireshark 149
Prerequisites for Configuring Embedded Packet Capture 149
Restrictions for Configuring Packet Capture 150
Restrictions for Configuring Wireshark 150
Restrictions for Configuring Embedded Packet Capture 151
Information About Packet Capture 152
About Wireshark 152
Capture Points 153
Attachment Points 153
Filters 153
Actions 154
Storage of Captured Packets to Buffer in Memory 154
Storage of Captured Packets to a .pcap File 155
Packet Decoding and Display 155
Packet Storage and Display 156
Wireshark Capture Point Activation and Deactivation 156
Wireshark Features 156
Guidelines for Configuring Wireshark 157
Default Wireshark Configuration 158
About Embedded Packet Capture 159
Benefits of Embedded Packet Capture 159
Packet Data Capture 159
How to Configure Packet Capture 160
How to Configure Wireshark 160
Defining a Capture Point 160

Adding or Modifying Capture Point Parameters 162
Deleting Capture Point Parameters 164
Deleting a Capture Point 166
Activating and Deactivating a Capture Point 167
Clearing the Capture Point Buffer 169
How to Implement Embedded Packet Capture 170
Managing Packet Data Capture 170
Monitoring and Maintaining Captured Data 172
Configuration Examples for Packet Capture 173
Configuration Examples for Wireshark 173
Example: Displaying a Brief Output from a .pcap File 173
Example: Displaying Detailed Output from a .pcap File 174
Example: Displaying a Packet Dump Output from a .pcap File. 175
Example: Displaying Packets from a .pcap File using a Display Filter 176
Example: Displaying the Number of Packets Captured in a .pcap File 176
Example: Displaying a Single Packet Dump from a .pcap File 176
Example: Displaying Statistics of Packets Captured in a .pcap File 177
Example: Simple Capture and Display 177
Example: Simple Capture and Store 178
Example: Using Buffer Capture 180
Example: Simple Capture and Store of Packets in Egress Direction 186
Configuration Examples for Embedded Packet Capture 188
Example: Managing Packet Data Capture 188
Example: Monitoring and Maintaining Captured Data 188
Additional References 190
Feature History for Packet Capture 190

CHAPTER 10 **Configuring Enhanced Drop Detection and Enhanced Packet Drop Analyzer 193**
Restrictions for Configuring Enhanced Drop Detection and Enhanced Packet Drop Analyzer 193
Information About Enhanced Drop Detection and Enhanced Packet Drop Analyzer 194
Enhanced Drop Detection 194
Enhanced Packet Drop Analyzer 194
Troublehsooting Packet Drop Using Enhanced Drop Detection and Enhanced Packet Drop Analyzer 195

Configuring Enhanced Drop Detection and Enhanced Packet Drop Analyzer to Troubleshoot Packet Drops 195
Configuration Examples for Enhanced Drop Detection and Enhanced Packet Drop Analyzer 196
Example: Configuring Enhanced Drop Detection 196
Example: Configuring Enhanced Packet Drop Analyzer 198
Feature History for Enhanced Drop Detection and Enhanced Packet Drop Analyzer 199

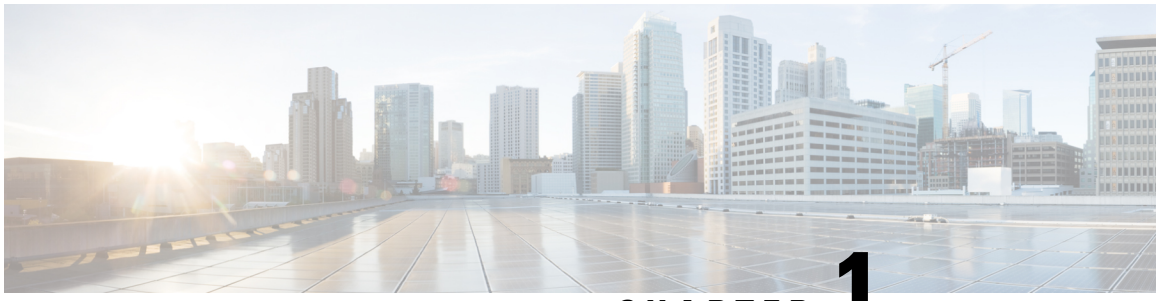
CHAPTER 11 **Configuring Flexible NetFlow** 201
Prerequisites for Flexible NetFlow 201
Restrictions for Flexible NetFlow 202
Information About Flexible NetFlow 204
Flexible NetFlow Overview 204
Original NetFlow and Benefits of Flexible NetFlow 205
Flexible NetFlow Components 206
Flow Records 206
Flow Exporters 211
Flow Monitors 213
Flow Samplers 215
Supported Flexible NetFlow Fields 215
Default Settings 219
Flexible NetFlow—Ingress VRF Support Overview 220
Flexible Netflow—Egress VRF Support Overview 220
Autonomous System Number 220
Ingress Egress Flexible NetFlow on MPLS Overview 220
Configuring VPN ID in Flexible NetFlow 221
How to Configure Flexible Netflow 221
Creating a Flow Record 221
Creating a Flow Exporter 223
Configuring a Flow Exporter for Flexible NetFlow V5 225
Creating a Customized Flow Monitor 228
Creating a Flow Sampler 230
Applying a Flow to an Interface 231
Configuring a Bridged NetFlow on a VLAN 233
Configuring Layer 2 NetFlow 233

Monitoring Flexible NetFlow 235
Configuration Examples for Flexible NetFlow 235
Example: Configuring a Flow 235
Example: Monitoring IPv4 ingress traffic 236
Example: Monitoring IPv4 egress traffic 237
Example: Configuring Flexible NetFlow for Ingress VRF Support 238
Example: Configuring Flexible NetFlow for Egress VRF Support 238
Feature History for Flexible NetFlow 239

CHAPTER 12 **Configuring IP-aware Netflow for VRF Ingress** 241
Restrictions for IP-aware Netflow for VRF Ingress 241
Information About IP-aware Netflow for VRF Ingress 241
How to Configure IP-aware Netflow for VRF Ingress 242
Creating a Flow Record 242
Creating a Flow Exporter 243
Creating a Flow Monitor 244
Applying Flow Monitor to an Interface 246
Configuration Examples for IP-aware Netflow for VRF Ingress 246
Feature History for IP-aware Netflow for VRF Ingress 248

CHAPTER 13 **Top-N Reports** 251
Information About Top-N Reports 251
How to use Top-N Reports 252
Enabling Top-N Reports 252
Displaying Top-N Reports 253
Clearing Top-N Reports 253
Examples : Top-N Reports 254
Feature History for Top-N Reports 255

CHAPTER 14 **Troubleshooting Network Management** 257
Overview 257
Support Articles 257
Feedback Request 258
Disclaimer and Caution 258



CHAPTER 1

Configuring Autoconf

The following sections provide information about Autoconf and how to configure Autoconf:

- 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 16
- Additional References for Autoconf, on page 18
- Feature History for Autoconf, on page 18

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

The following sections provide information about Autoconf.

Benefits of Autoconf

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

The Autoconf feature automatically applies the necessary 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, 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 or policies or both to be applied as templates.

Both service and interface templates are named as containers of configuration and policy. Service templates can be applied only to access sessions, while interface templates can be applied only to ports. When a service template is applied to an access session, the contained configuration and policy are 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 the 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 users to include customized configurations, limiting the need to create a new template.

The templates created by users are referred to as user-defined templates. These templates can be defined on a 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 the **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 the **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 above tasks 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 are displayed 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 at one or more access sessions, any referenced template must be applied to the interface that is 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 an 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 or interface template (as referenced in a parameter map) or both based on the type of filter, and removes templates, if any, associated with a previous policy.
- Device-to-template parameter map: A subscriber parameter map that allows the filter type-to-service or interface template mappings or both to be specified in an efficient and readable manner.

Advantages of Using Templates

Using templates for auto configuration has the following benefits:

- Templates are parsed once when they are being defined. This makes the 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 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 a device are not modified by the dynamic application of a template.
- Policy application is synchronized with the access-session life cycle. This is tracked by the framework by using all the available techniques, including link-up or link-down.

- Templates can be updated with a single operation. All the applied instances of templates are also updated during this operation.
- Constituent commands of 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 performance of remedial actions when 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 the 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 the interface-level configurations are disabled.

Table 1: Autoconf Functionality

Global	Interface Level	AutoConf Status
Disable	Disable	No automatic configurations are applied when an end device is connected.
Enable	Enable	If Autoconf is enabled at the global level, it is also enabled at the interface level by default. Built-in template configurations are applied based on the end devices that are detected on all the 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 **access-session interface-template sticky** command 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 the 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.

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

The following sections provide information about 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 a 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 an 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 an 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 configuring 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 built-in 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:**	Exits template configuration mode and enters global configuration mode.

	Command or Action	Purpose
	`Device(config-template)# exit`	
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 an 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 the built-in interface template is applied on an 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 by 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.

Configuring a Platform Type Filter

The following tasks shows how to configure a platform type filter for class maps and parameter maps.

Configuring a Platform Type Filter for a Class Map

A control class defines the conditions under which the actions of a control policy are executed. You should define whether all, any, or none of the conditions must be evaluated to execute the actions of the control policy. Platform types are evaluated based on the specified platform name in the control policy.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**class-map type control subscriber** {**match-all** | **match-any** | **match-none**} *control-class-name* **Example:** `Device(config)# class-map type control subscriber match-all DOT1X_NO_AGENT`	Creates a control class and enters control class-map filter mode. • **match-all**: Must match all the conditions in the control class. • **match-any**: Must match at least one condition in the control class. • **match-none**: Must not match any of the conditions in the control class.
Step 4	**match platform-type** *platform-name* **Example:** `Device(config-filter-control-classmap)# match platform-type C3850`	Creates a condition to evaluate control classes based on the specified platform type.
Step 5	**end** **Example:** `Device(config-filter-control-classmap)# end`	Exits control class-map filter mode and returns to privileged EXEC mode.
Step 6	**show class-map type control subscriber** {**all** | **name** *control-class-name*} **Example:** `Device# show class-map type control subscriber all`	(Optional) Displays information about control policies for all the class maps or a specific class map.

Configuring a Platform Type Filter for a Parameter Map

We recommend that you use the parameter map.

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	**parameter-map type subscriber attribute-to-service** *parameter-map-name* **Example:** `Device(config)# parameter-map type subscriber attribute-to-service Aironet-Policy-para`	Specifies the parameter map type and name, and enters parameter-map filter mode.
Step 4	*map-index* **map platform-type** {{**eq** | **not-eq** | **regex**} *filter-name*} **Example:** `Device(config-parameter-map-filter)# 10 map platform-type eq C9xxx`	Specifies the parameter map attribute filter criteria to the platform type.
Step 5	**end** **Example:** `Device(config-parameter-map-filter-submode)# end`	Exits parameter-map filter mode and returns to privileged EXEC mode.
Step 6	**show parameter-map type subscriber attribute-to-service**{**all | name** *parameter-map-name*} **Example:** `Device# show parameter-map type subscriber attribute-to-service`	(Optional) Displays the parameter map attributes.

Configuring a Device Type Filter for a Class Map

A control class defines the conditions under which the actions of a control policy are executed. You should define whether all, any, or none of the conditions must be evaluated to execute the actions of the control policy. Device types are evaluated based on the specified device name in the control policy.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**class-map type control subscriber** {**match-all** | **match-any** | **match-none**} *control-class-name* **Example:** `Device(config)# class-map type control subscriber match-all Device-Type-Match`	Creates a control class and enters control class-map filter mode. • **match-all**: Must match all the conditions in the control class. • **match-any**: Must match at least one condition in the control class. • **match-none**: Must not match any of the conditions in the control class.

Step 4	**match device-type** {*device-name*|**regex** *regular-expression*}
	Example:
	`Device(config-filter-control-classmap)# match device-type laptop`
	`Device(config-filter-control-classmap)# match device-type regex cis*`
		Creates a condition to evaluate control classes based on the specified device type.
		• *device-name*: Enter a device name for the class map attribute filter criteria.
		• **regex** *regular-expression*: Enter a regular expression to specify the filter type.

Step 5	**end** **Example:** `Device(config-filter-control-classmap)# end`	Exits control class-map filter mode and returns to privileged EXEC mode.
Step 6	**show class-map type control subscriber** {**all** | **name** *control-class-name*} **Example:** `Device# show class-map type control subscriber all`	(Optional) Displays information about control policies for all the class maps or a specific class map.

Configuration Examples for Autoconf

The following sections provide configuration examples for the Autoconf feature.

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 apply a modified built-in template to an end device 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
```

Example: Configuring a Platform Type Filter

The following example shows how to configure a platform type filter for a class map:

```
Device> enable
Device# configure terminal
Device(config)# class-map type control subscriber match-all DOT1X_NO_AGENT
Device(config-filter-control-classmap)# match platform-type C9xxx
Device(config-filter-control-classmap)# end
Device#
```

The following example shows how to configure a platform type filter for a parameter map:

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

```
Device(config)# parameter-map type subscriber attribute-to-service Aironet-Policy-para
Device(config-parameter-map-filter)# 10 map platform-type eq C9xxx
Device(config-parameter-map-filter-submode)# end
Device#
```

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*

Feature History for Autoconf

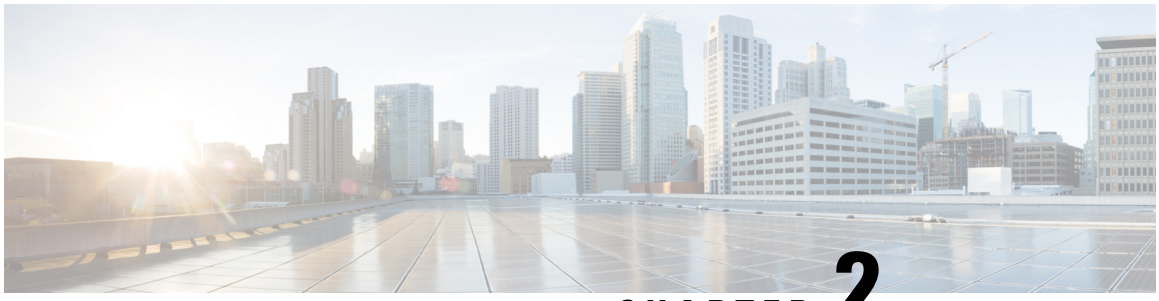
This table provides release and related information for the features explained in this module.

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

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	Autoconf	The Autoconf feature permits hardbinding between an end device and an interface. In Autoconf the core session management capability is decoupled from the application-specific logic, allowing the same framework to be used regardless of the criteria for policy determination or the nature of the policies applied.
Cisco IOS XE Gibraltar 16.12.1	AutoConf Device Granularity to PID of Cisco Switch	The **platform type** filter option was introduced for class map and parameter map configurations.

Release	Feature	Feature Information
Cisco IOS XE Cupertino 17.7.1	Autoconf	This feature was implemented on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

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 Interface Templates

The following sections provide information about Interface Templates and how to configure and bind Interface Templates to a target:

- Restrictions for Interface Templates, on page 21
- Information About Interface Templates, on page 21
- How to Configure Interface Templates, on page 24
- Configuration Examples for Interface Templates, on page 32
- Feature History for Interface Templates, on page 33

Restrictions for Interface Templates

- Remote storing and downloading of templates is not supported.
- To dynamically bind an interface template, the interface template with the same name as referred by AAA Authorization has to be configured on the device.
- Source template configurations are not supported on port channel interface.
- ASP macro and Autoconf are not recommended to be used on the same interface at the same time.

Information About Interface Templates

This section describes an interface template, its types and usage.

Interface Template Overview

An interface template is a container of configurations or policies that can be applied to specific ports. When an interface template is applied to an access port, it impacts all traffic that is exchanged on the port.

There are two types of interface templates; user and builtin templates. Builtin templates are created by the system.

You can modify builtin templates. If you delete a modified builtin template, the system restores the original definition of the template.

The following are the available builtin templates:

- AP_INTERFACE_TEMPLATE (Access Point)
- DMP_INTERFACE_TEMPLATE (Digital Media Player)
- IP_CAMERA_INTERFACE_TEMPLATE
- IP_PHONE_INTERFACE_TEMPLATE
- LAP_INTERFACE_TEMPLATE (Lightweight Access Point)
- MSP_CAMERA_INTERFACE_TEMPLATE
- MSP_VC_INTERFACE_TEMPLATE (Video Conferencing)
- PRINTER_INTERFACE_TEMPLATE
- ROUTER_INTERFACE_TEMPLATE
- SWITCH_INTERFACE_TEMPLATE
- TP_INTERFACE_TEMPLATE (TelePresence)

Following is an example of a builtin interface template:

```
Template Name       : IP_CAMERA_INTERFACE_TEMPLATE
Modified            : No
Template Definition :
 spanning-tree portfast
 spanning-tree bpduguard enable
 switchport mode access
 switchport block unicast
 switchport port-security
 mls qos trust dscp
 srr-queue bandwidth share 1 30 35 5
 priority-queue out
!
```

You can also create specific user templates with the commands that you want to include.

Note The template name must not contain spaces.

You can create an interface template using the **template** command in global configuration mode. In template configuration mode, enter the required commands. The following commands can be entered in template configuration mode:

Command	Description
access-session	Configures access session-specific interface commands. This is applicable to Identity-Based Networking Services (IBNS) 2.0
authentication	Configures authentication manager Interface Configuration commands. This is applicable to IBNS1.0
carrier-delay	Configures delay for interface transitions.

Command	Description
default	Sets a command to its defaults.
description	Configures interface-specific description.
dot1x	Configures interface configuration commands for IEEE 802.1X.
ip	Configures IP template.
keepalive	Enables keepalive.
load-interval	Specifies interval for load calculation for an interface.
mab	Configures MAC authentication bypass Interface.
peer	Configures peer parameters for point to point interfaces.
service-policy	Configures CPL service policy.
source	Gets configurations from another source.
spanning-tree	Configures spanning tree subsystem
storm-control	Configures storm control.
subscriber	Configures subscriber inactivity timeout value.
switchport	Sets switching mode configurations

Note System builtin templates aren't displayed in the running configuration. These templates show up in the running configuration only if you edit them.

Binding an Interface Template to a Target

Each template can be bound to a target, like an interface or a sub-interface. A template can be attached to a target either statically or dynamically. Static binding of a template involves binding the template to a target, like an interface. Only one template can be bound at a time using static binding. Static binding of another template to the same target will unbind the previously bound template. To configure static binding, use the **source template** command in interface configuration mode.

Any number of templates can be bound dynamically to a target. To configure dynamic binding using builtin policy maps and parameter maps, enable the Autoconf feature using the **autoconf enable** command.

Note You can have statically and dynamically bound templates on the same interface at a time.

Priority for Configurations Using Interface Templates

Configuration applied through dynamically-bound templates has the highest priority, followed by configuration applied directly on the interface, and then configuration applied through statically-bound templates. When similar commands are present at different priority levels, the one at the highest priority is applied. If a configuration at a higher priority level is not applied, then the configuration with the next highest priority is applied to the target.

Multiple templates can be dynamically bound to a target. When multiple templates are dynamically bound, the template that is applied last has the highest priority.

To delete a template, you must remove the binding to all targets. If you bind a template that does not exist, a new template is created with no configurations.

How to Configure Interface Templates

Perform the following tasks to configure a user interface template and bind it to a target.

Configuring Interface Templates

Perform the following task to create user interface templates:

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	**template** *name* **Example:** `Device(config)# template user-template1`	Creates a user template and enters template configuration mode. **Note** Builtin templates are system-generated.
Step 4	**load-interval** *interval* **Example:** `Device(config-template)# load-interval 60`	Configures the sampling interval for statistics collections on the template. **Note** Builtin templates are system-generated.
Step 5	**description** *description* **Example:**	Configures the description for the template.

	Command or Action	Purpose
	`Device(config-template)# description This is a user template`	
Step 6	**keepalive** *number* **Example:** `Device(config-template)# Keepalive 60`	Configures the keepalive timer.
Step 7	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring Static Binding for Interface Templates

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** *type number* **Example:** `Device(config)# interface GigabitEthernet 1/0/12`	Specifies the interface type and number and enters interface configuration mode.
Step 4	**source template** *name* **Example:** `Device(config-if)# source template user-template1`	Statically applies an interface template to a target.
Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Example

To verify static binding use the **show running-config interface** *int-name* and the **show derived-config interface** *int-name* commands.

```
Device# show running-config interface GigabitEthernet 1/0/12

Building configuration...

Current configuration : 71 bytes
!
interface GigabitEthernet1/0/12
source template user-template1
end
```

```
Device# show derived-config interface GigabitEthernet 1/0/12
Building configuration...

Derived configuration : 108 bytes
!
interface GigabitEthernet1/0/12
description This is a user template
load-interval 60
keepalive 60
end
```

Configuring Dynamic Binding of Interface Templates

To configure dynamic binding of interface templates, perform the following tasks:

Before you begin

Ensure that 802.1x port-based authentication is configured on the device. When you are using the ISE server to download and assign a template that includes **switchport mode** and vlan change commands, the **access-session interface-template sticky** command is used, which is available only in IBNS 2.0. Hence, VLAN changes using a template require IBNS 2.0.

Procedure

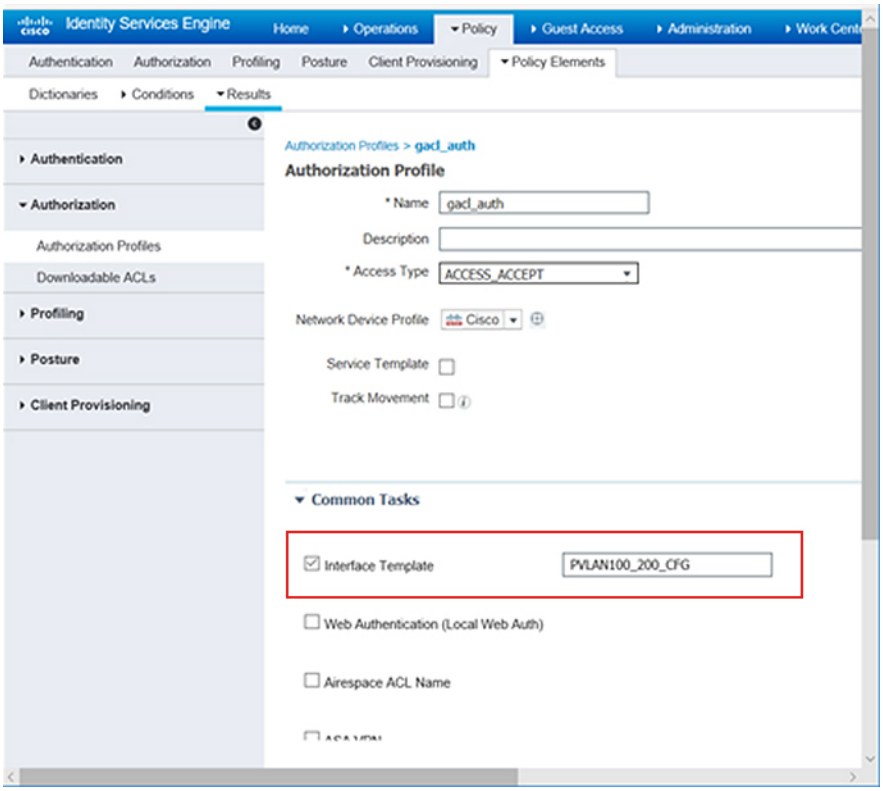
Step 1 Create a template on the device, as specified in the Configuring Interface Templates, on page 24 procedure.

Step 2 Configure the Identity Services Engine (ISE) or any other RADIUS server to download the template name to the device interface. The template is assigned after the device is authenticated successfully.

If you're using ISE, go to the **Policy** > **Policy Elements** > **Authorization** > **Authorization Profile** page.

Check the **Interface Template** check box and enter the name of the template to be assigned to the client interface.

Figure 1: Configure ISE to Assign Interface Template



If you're using a different RADIUS server, configure the attribute **Cisco-AVpair="interface:template=*name*"** with the name of the template. This configuration pushes the template to the device after the initial client authentication is completed.

Step 3 To verify that the template name is downloaded to the interface, use the **show authentication sessions interface** *interface-id* **details** or **show access-session interface** *interface-id* **details** command. To verify that the interface template commands are applied to the interface, use the **show derived-config interface** *interface-id* command.

Example

The following example shows how to verify that a template named del_template is downloaded and applied to the TwentyFiveGigE1/0/3 interface on the device:

```
Device# show running-config | section del_template
template del_template
access-session port-control auto
no access-session monitor
authentication periodic

Device# show authentication sessions interface tw1/0/3 details
Interface: TwentyFiveGigE1/0/3
IIF-ID: 0x1F9EBBA9
MAC Address: 002f.0100.0001
IPv6 Address: Unknown
IPv4 Address: Unknown
```

```
User-Name: NOAS
Device-type: Un-Classified Device
Device-name: Unknown Device
Status: Authorized
Domain: DATA
Oper host mode: multi-auth
Oper control dir: both
Session timeout: N/A
Common Session ID: 0404140B00004E1C6E569E0B
Acct Session ID: Unknown
Handle: 0xdb000e24
Current Policy: DOT1x

Server Policies:
Interface Template: del_template

Method status list:
Method State
dot1x Authc Success


Device# show derived-config interface tw1/0/3
Building configuration...

Derived configuration : 321 bytes
!
interface TwentyFiveGigE1/0/3
switchport access vlan 44
switchport mode access
switchport port-security violation restrict
switchport port-security
authentication periodic
access-session port-control auto
no access-session monitor
mab
dot1x pae authenticator
service-policy type control subscriber DOT1x
end
```

Verifying Interface Templates

Use one or more of the commands listed below to verify the interface template configuration.

Table 2: Show commands to verify Interface Template Configuration

Command	Purpose
show template interface all {**all** | **binding** {*temp-name* | **all** | **target** *int-name*} | **brief** }	Shows all interface template configurations.
show template interface source {**built-in** [**original**] | **user**}{*temp-name* | **all**}}	Shows interface template source configurations.
show template service{**all** | **binding target** *int-name* | **brief** | **source** {**aaa** | **built-in** | **user** {**temp-name** | **all**}}	Shows all interface template service configurations.

Example: Verifying Interface User Template

```
Device# show template interface source user all
    Template Name : TEST-1
    Template Definition:
    load-interval 60
    description TEST_1_TEMPLATE
    keepalive 200
    !
    Template Name : TEST-2
    Template Definition:
    load-interval 60
    description TEST-1_TEMPLATE
    keepalive 200
```

Example: Verifying all Builtin Templates

```
Device#  show template interface source built-in all

Building configuration...

Template Name : AP_INTERFACE_TEMPLATE
Modified : No
Template Definition :
switchport mode trunk
switchport nonegotiate
service-policy input AutoConf-4.0-Trust-Cos-Input-Policy
service-policy output AutoConf-4.0-Output-Policy
!
Template Name : DMP_INTERFACE_TEMPLATE
Modified : No
Template Definition :
switchport mode access
switchport block unicast
switchport port-security
spanning-tree portfast
spanning-tree bpduguard enable
service-policy input AutoConf-4.0-Trust-Dscp-Input-Policy
service-policy output AutoConf-4.0-Output-Policy
!
Template Name : IP_CAMERA_INTERFACE_TEMPLATE
Modified : No
Template Definition :
switchport mode access
switchport block unicast
switchport port-security
spanning-tree portfast
spanning-tree bpduguard enable
service-policy input AutoConf-4.0-Trust-Dscp-Input-Policy
service-policy output AutoConf-4.0-Output-Policy
!
Template Name : IP_PHONE_INTERFACE_TEMPLATE
Modified : No
Template Definition :
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
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
load-interval 30
!
Template Name : LAP_INTERFACE_TEMPLATE
Modified : No
Template Definition :
switchport mode access
switchport block unicast
switchport port-security violation protect
switchport port-security aging time 2
switchport port-security aging type inactivity
switchport port-security
storm-control broadcast level pps 1k
storm-control multicast level pps 2k
storm-control action trap
spanning-tree portfast
spanning-tree bpduguard enable
ip dhcp snooping limit rate 15
load-interval 30
!
Template Name : MSP_CAMERA_INTERFACE_TEMPLATE
Modified : No
Template Definition :
switchport mode access
switchport block unicast
switchport port-security
spanning-tree portfast
spanning-tree bpduguard enable
!
Template Name : MSP_VC_INTERFACE_TEMPLATE
Modified : No
Template Definition :
switchport mode access
switchport port-security
spanning-tree portfast
spanning-tree bpduguard enable
load-interval 30
!
Template Name : PRINTER_INTERFACE_TEMPLATE
Modified : No
Template Definition :
switchport mode access
switchport port-security maximum 2
switchport port-security
spanning-tree portfast
spanning-tree bpduguard enable
load-interval 60
!
Template Name : ROUTER_INTERFACE_TEMPLATE
Modified : No
Template Definition :
switchport mode trunk
spanning-tree portfast trunk
spanning-tree bpduguard enable
service-policy input AutoConf-4.0-Trust-Cos-Input-Policy
service-policy output AutoConf-4.0-Output-Policy
!
Template Name : SWITCH_INTERFACE_TEMPLATE
Modified : No
```

```
Template Definition :
switchport mode trunk
service-policy input AutoConf-4.0-Trust-Cos-Input-Policy
service-policy output AutoConf-4.0-Output-Policy
!
Template Name : TP_INTERFACE_TEMPLATE
Modified : No
Template Definition :
switchport mode access
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
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-Trust-Dscp-Input-Policy
service-policy output AutoConf-4.0-Output-Policy
ip dhcp snooping limit rate 15
load-interval 30
!
end
```

Example: Verifying Interface Template Binding for all templates

```
Device# show template interface binding all
    Template-name               Source      Method          Interface
    =============               ====      ==========      ==========
IP_PHONE_INTERFACE_TEMPLATE    Built-in    Dynamic        Gi1/0/1, Gi1/0/2, Gi1/0/3
                                                          Gi1/0/4, Gi1/0/5, Gi1/0/6
                                                          Gi1/0/7, Gi1/0/8, Gi1/0/9
                                                          Gi1/0/10, Gi1/0/11, Gi1/0/12
                                                          Gi1/0/13, Gi1/0/14, Gi1/0/15
                                                          Gi1/0/16, Gi1/0/17, Gi1/0/18
                                                          Gi1/0/19, Gi1/0/20, Gi1/0/21
                                                          Gi1/0/22, Gi1/0/23, Gi1/0/24
                                                          Gi1/1/1, Gi1/1/2, Gi1/1/3

IP_PHONE_INTERFACE_TEMPLATE    Built-in     Static        Gi4/0/4
```

Example: Verifying Static Template Binding for a Target Interface

```
Device# show template interface binding target GigabitEthernet 1/0/4
    Interface            Method       Source             Template
    =========           ==========    =====              =========
    Gi1/0/4              Dynamic      built-in           IP_PHONE_INTERFACE_TEMPLATE
                         Static       user               TEST
                         Dynamic      Modified-built-in TEST
```

Example: Verifying Dynamic Template Binding for all templates

```
Device# show template service all

    User-defined template:
    ======================

    Template Name      : SVC-1
    Template Definition:
    vlan 100
```

```
access-group acl1

built-in template:
======================

Template Name      : SVC-2
Template Definition:
vlan 100
access-group acl1

aaa downloaded template:
==========================
Template Name      : SVC-2
Template Definition:
vlan 100
access-group acl1
```

Example: Verifying Template Binding for a Target Interface

```
Device# show template binding target GigabitEthernet 1/0/4

  Interface Templates:
    Interface            method       Source            Template
    =========           ==========    =====             =========
    Gi1/0/4              Dynamic      built-in          IP_PHONE_INTERFACE_TEMPLATE
                         Static       user              TEST
                         Dynamic      Modified-built-in TEST
  Service Templates:
   Template            Source         Session-Mac
   ========            ====           ================
   SVC1                user           aa-bb-cc-dd-ee-ff
   SVC2                built-in       ab-ab-ab-ab-ab-ab
   SVC3                aaa            ac-ac-ac-ac-ac-ac
```

Configuration Examples for Interface Templates

Example: Configuring User Interface Templates

Example: Configuring User Templates

```
Device# enable
Device (config)# configure terminal
Device(config)# template user-template1
Device(config-template)# load-interval 60
Device(config-template)# description This is a user template
Device(config-template)# Keepalive 60
Device(config)# end
```

Example: Sourcing Interface Templates

```
Device> enable
Device# configure terminal
Device(config)# interface fastethernet 4/0/0
```

```
Device(config-if)# source template user-template1
Device(config-if)# end
```

Example: Dynamically Binding Interface Templates

Configure a template on the device:

```
Device# configure terminal
Device(config)# tempalte user_template
Device(config-template)# access-session port-control auto
Device(config-template)# no access-session monitor
Device(config-template)# authentication periodic
```

Configure RADIUS Server attribute with the template name:

```
Cisco-AVpair="interface:template=user_template"
```

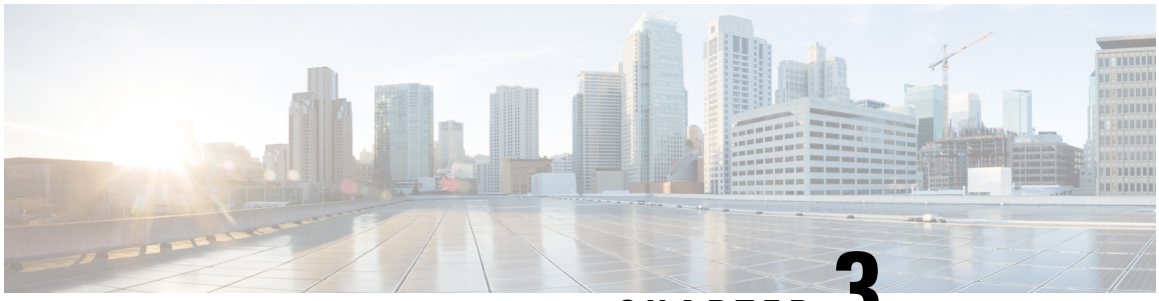
Feature History for Interface Templates

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 notedotherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	Interface Templates	An interface template provides a mechanism to configure multiple commands at the same time and associate it with a target such as an interface.
Cisco IOS XE Bengaluru 17.5.1	Interface Templates - **cts manual** command	The command **cts manual** was introduced. It can be configured in the template configuration mode.
Cisco IOS XE Cupertino 17.7.1	Interface Templates	This feature was implemented on Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

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 3

Configuring Cisco Plug and Play

- Configuring Cisco Plug and Play, on page 35
- Feature History for Cisco Plug and Play, on page 35

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

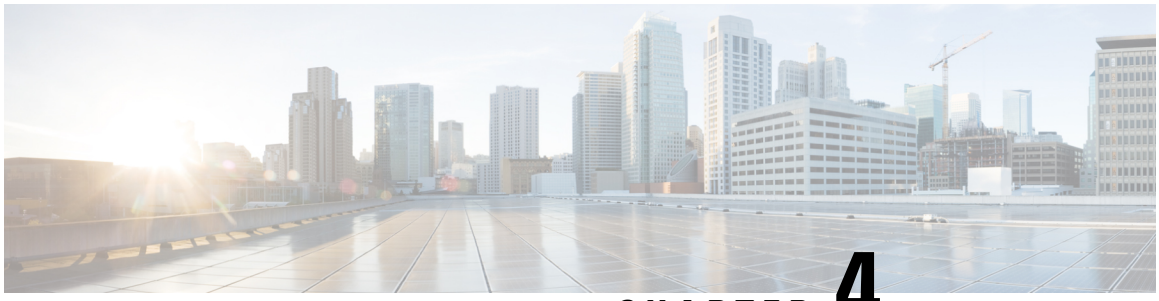
Feature History for Cisco Plug and Play

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 notedotherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	Cisco Plug and Play	The Cisco Plug and Play solution is a converged solution that provides a highly secure, scalable, seamless, and unified zero-touch deployment experience.
Cisco IOS XE Cupertino 17.7.1	Cisco Plug and Play	This feature was implemented on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

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 4

Configuring 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 Cisco Discovery Protocol, on page 37
- How to Configure Cisco Discovery Protocol, on page 38
- Monitoring and Maintaining Cisco Discovery Protocol, on page 44
- Feature History for Cisco Discovery Protocol, on page 45

Information about Cisco Discovery Protocol

The following sections provide information about Cisco Discovery Protocol

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

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.

The following applies to a device and connected endpoint devices:

- 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.

How to Configure Cisco Discovery Protocol

The following sections provide information about how to configure Cisco Discovery Protocol.

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:**	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
	`Device>enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	
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.

	Command or Action	Purpose
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

Use the **show run all** command to check if Cisco Discovery Protocol has been enabled. If you run the **show run** command, 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.
- 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.
- 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**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device>enable`	• 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 3: 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 nei

Command	Description
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 the specified neighbor or information about the version of softwa the 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.
show ap cdp neighbors	Displays information regarding the access point's Cisco Discove neighbors.
show ap cdp neighbors detail	Displays detailed information regarding the access point's Cisco Protocol neighbors.
show ap name *ap-name* **cdp neighbors**	Displays the Cisco Discovery Protocol information for an access
show ap name *ap-name* **cdp neighbors detail**	Displays details about a specific access point neighbor that is usi Discovery Protocol.

Feature History for Cisco Discovery Protocol

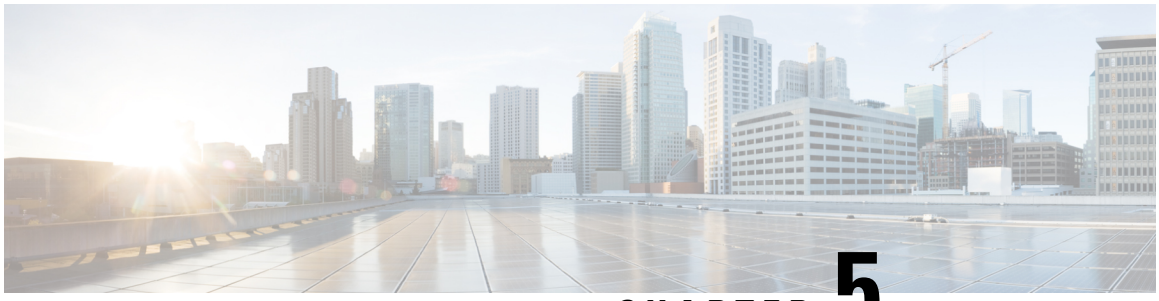
This table provides release and related information for the features explained in this module.

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 Gibraltar 16.11.1	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.

Release	Feature	Feature Information
Cisco IOS XE Cupertino 17.7.1	Cisco Discovery Protocol	This feature was implemented on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

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 5

Configuring Simple Network Management Protocol

- Prerequisites for SNMP, on page 47
- Restrictions for SNMP, on page 49
- Information About SNMP, on page 49
- How to Configure SNMP, on page 55
- SNMP Examples, on page 64
- Monitoring SNMP Status, on page 65
- Feature History for Simple Network Management Protocol, on page 65

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 4: 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

- SNMPv1 does not support informs.
- The backslash "\" character is not permitted in SNMPv3 authentication and privacy credentials.

Information About SNMP

The following sections provide 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 5: 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.

Note We recommend that the SNMP Manager exclude the **ciscoFlashFileDate** MIB object from its query, to avoid performance related issues. This is because, though the **ciscoFlashFileDate** object is published in the MIB, it is not supported on the product.

SNMP Agent Functions

The SNMP agent can receive requests from one or more SNMP managers. Every request carries the NMS IP address, the number of times an NMS polls the agent, and a timestamp of polling. This information can be tracked for both IPv4 and IPv6 servers.

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.

Use the **show snmp stats hosts** command to display the list of the SNMP managers requests in the queue, and use the **clear snmp stats hosts** command to clear the queue.

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:

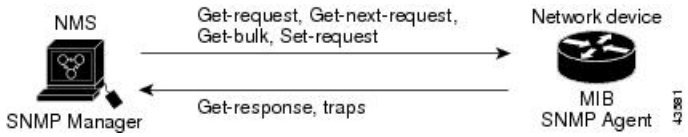
- Read-only (RO)—Gives all objects in the MIB except the community strings read access to authorized management stations, but does not allow write access.
- Read-write (RW)—Gives all objects in the MIB read and write access to authorized management stations, but does not allow access to the community 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 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 2: SNMP Network



SNMP Flash MIB

The Cisco Flash MIB is used to query flash file data from Cisco devices. Earlier the number of files listed per partition per device in the Flash MIB was limited to 100. Starting with Cisco IOS XE 17.1.1 release, the limitation of Flash MIB listing 100 files per partition per device has been removed. Now Flash MIB will fetch all the files from the flash file system. It is mandatory to use the **snmp mib flash cache** command to perform a Flash MIB walk. The **snmp mib flash cache** command will prefetch all the files into the local Flash MIB cache. Since the 100 file limitation has been removed the Flash MIB walk to retrieve the files will take longer.

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.

SNMP and Syslog Over IPv6

To support both IPv4 and IPv6, IPv6 network management requires both IPv6 and IPv4 transports. Syslog over IPv6 supports address data types for these transports.

Simple Network Management Protocol (SNMP) and syslog over IPv6 provide these features:

- Support for both IPv4 and IPv6
- IPv6 transport for SNMP and to modify the SNMP agent to support traps for an IPv6 host
- SNMP- and syslog-related MIBs to support IPv6 addressing
- Configuration of IPv6 hosts as trap receivers

For support over IPv6, SNMP modifies the existing IP transport mapping to simultaneously support IPv4 and IPv6. These SNMP actions support IPv6 transport management:

- Opens User Datagram Protocol (UDP) SNMP socket with default settings
- Provides a new transport mechanism called *SR_IPV6_TRANSPORT*
- Sends SNMP notifications over IPv6 transport
- Supports SNMP-named access lists for IPv6 transport
- Supports SNMP proxy forwarding using IPv6 transport
- Verifies SNMP Manager feature works with IPv6 transport

For information on SNMP over IPv6, including configuration procedures, see the “Managing Cisco IOS Applications over IPv6” chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

For information about syslog over IPv6, including configuration procedures, see the “Implementing IPv6 Addressing and Basic Connectivity” chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

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.

Feature	Default Setting
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

The device requires one of the following global configuration commands configured in order to open SNMP UDP ports 161 and 162 and enable the SNMP agent: **snmp-server host**, or **snmp-server user**, or **snmp-server community**, or **snmp-server manager**.

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.
- When you configure the SNMP server host with the default UDP port, 162, the output of the **show running-config** command does not display the UDP port value. If you specify a UDP port value other than the default by using the **snmp-server host** {*host-addr*} *community-string* **udp-port** *value* command, the UDP port number will be displayed in the **show running-config** command output. You can configure the **snmp-server host** command with or without the default UDP port 162; however, you cannot configure both simultaneously.

  The following examples are correct:

```
Device(config)# snmp-server host 10.10.10.10 community udp-port 163
Device(config)# snmp-server host 10.10.10.10 community

Device(config)# snmp-server host 10.10.10.10 community udp-port 163
Device(config)# snmp-server host 10.10.10.10 community udp-port 162
```

The following examples are incorrect:

```
Device(config)# snmp-server host 10.10.10.10 community udp-port 163
Device(config)# snmp-server host 10.10.10.10 community
Device(config)# snmp-server host 10.10.10.10 community udp-port 162

Device(config)# snmp-server host 10.10.10.10 community udp-port 163
Device(config)# snmp-server host 10.10.10.10 community udp-port 162
Device(config)# snmp-server host 10.10.10.10 community
```

How to Configure SNMP

The following sections provide information on how to configure SNMP.

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.

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**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• 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. • **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.

	Command or Action	Purpose
		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: • **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).

	Command or Action	Purpose
		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. **Note** The algorithms — **md5**, **des**, **3des** is not supported in a SNMPv3 group when the compliance shield is disabled. You need to enable the compliance shield using the **crypto engine compliance shield enable** command and reboot the device to configure the algorithms — **md5**, **des** and **3des**.
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.

Opening or Closing SNMP UDP Ports

The SNMP process uses ports 161 and 162 where port 161 is used for polling the device and port 162 is used for sending notifications from the agent to the server. The SNMP UDP ports remain closed unless one of the requisite commands is configured. This design provides additional security by opening the ports only when needed and prevents a device from listening to a port unnecessarily.

Beginning in user EXEC mode, follow these steps to open the SNMP UDP 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	**snmp-server** {**host** | **user** | **community** | **manager**} **Example:** `Device(config)# snmp-server host`	Opens SNMP UDP ports 161 and 162. Configuring any one of the options (**host**, **user**, **community**, **manager**) opens both ports. To close the ports, enter the **no** form of all the options that you have configured. The ports remain open as long as even one of the keywords is configured. If you enter the **no snmp-server** command, without any of the keywords, the SNMP process is shut down and not just the SNMP UDP ports.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show udp** **Example:** `Device# show udp`	Displays the SNMP UDP ports. If one of the requisite commands is configured, ports 161 and 162 will display value `listen` under the remote field.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

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.

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.

	Command or Action	Purpose
Step 4	**snmp-server location** *text* **Example:** `Device(config)# snmp-server location Building 3/Room 222`	Sets the system location string.
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 and shuts down the SNMP process. You reenable all versions of the SNMP agent by entering one of the following commands in global configuration mode: **snmp-server host**, or **snmp-server user**, or **snmp-server community**, or **snmp-server manager**. 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.

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
```

This example shows how to display the entries of SNMP Managers polled to an SNMP Agent:

```
Device# show snmp stats host
Request Count                  Last Timestamp              Address
2                               00:00:01 ago               3.3.3.3
1                               1w2d ago                   2.2.2.2
```

This example shows the message displayed by the device when you configure any of the three algorithms — **md5**, **des**, **3des** in a SNMPv3 group when complaince shield is disabled:

```
Device(config)# snmp-server user md5user grp v3 auth md5 cisco1234 priv des
Sep  1 00:14:51.582 IST: %SNMP-6-AUTHPROTOCOLMD5: Authentication protocol md5 support will
 be deprecated in future
Sep  1 00:14:51.582 IST: %SNMP-6-PRIVPROTOCOLDES: Privacy protocol des support will be
deprecated in future
Sep  1 00:14:51.645 IST: %SNMP-5-WARMSTART: SNMP agent on host Switch is undergoing a warm
 start
```

This example shows the message displayed by the device when you configure any of the three algorithms — **md5**, **des**, **3des** in a SNMPv3 group when complaince shield is enabled. The crypto algorithms is supported along with a warning message:

```
Device(config)# snmp-server user md5user grp v3 auth md5 cisco1234
weaker algorithm MD5, DES and 3DES is not allowed for snmp user
```

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 6: Commands for Displaying SNMP Information

Command	Purpose
show snmp	Displays SNMP statistics.
	Displays information on the local SNMP engine and all remote e have been configured on the device.
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 us **Note** You must use this command to display SNMPv3 configuration in **auth** | **noauth** | **priv** mode. This information is not displayed in t **running-config** output.

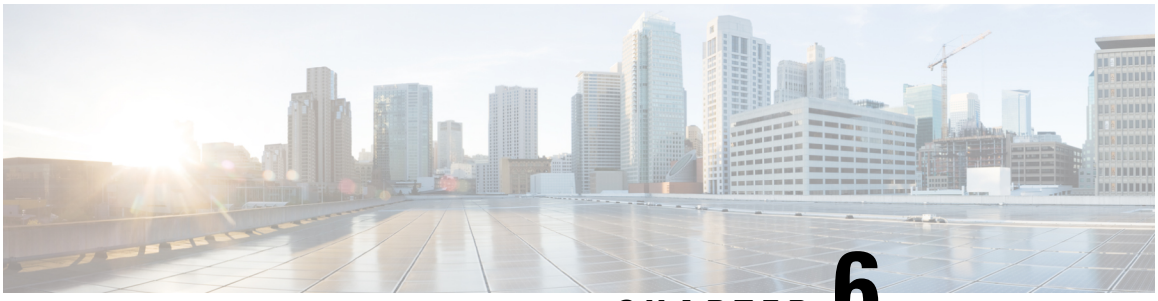
Feature History for Simple Network Management Protocol

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 notedotherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	SNMP	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)
Cisco IOS XE Amsterdam 17.1.1	Opening or Closing SNMP UDP ports	The SNMP UDP ports are opened only if one of the requisite commands is configured. This design provides additional security by opening the ports only when needed and prevents a device from listening to a port unnecessarily.
	Tracking NMS Polling	The details of the SNMP Manager polled to an SNMP Agent can be displayed. Use the show snmp server stats hosts command to display the NMS IP address, the number of times an NMS polls the agent, and a timestamp of polling.
Cisco IOS XE Cupertino 17.7.1	SNMP	This feature was implemented on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

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 6

Configuring Cisco IOS IP 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 Service Level Agreements, on page 67
- Information About Service Level Agreements, on page 67
- How to Configure IP SLAs Operations, on page 73
- Monitoring IP SLA Operations, on page 86
- Monitoring IP SLA Operation Examples, on page 86
- Additional References, on page 87
- Feature History for Service Level Agreements, on page 88

Restrictions on Service Level Agreements

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 Service Level Agreements

The following sections provide information about Service Level Agreements.

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 SLA 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:

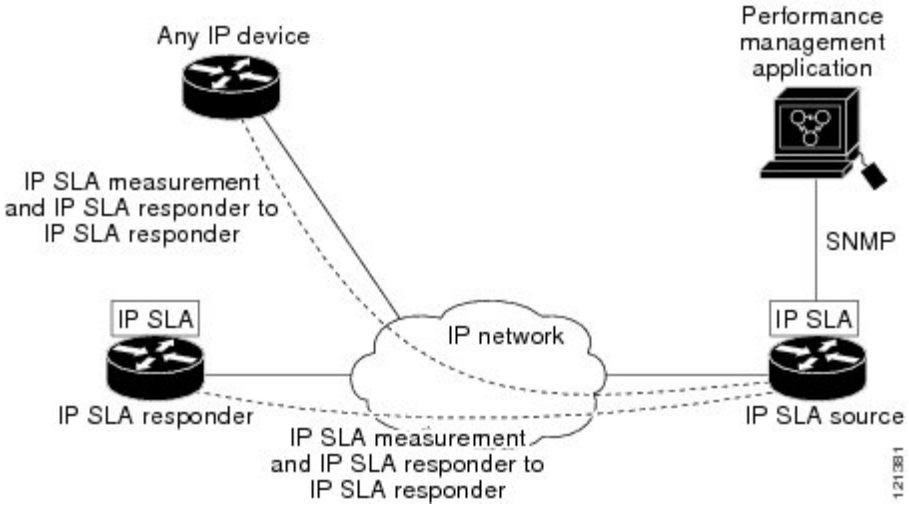
- Service-level agreement monitoring, measurement, and verification.
- Network performance monitoring:
  - Measurement of jitter, latency, or packet loss in the network.
  - Continuous, reliable, and predictable measurements.
- IP service network health assessment to verify that the existing QoS is sufficient 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 3: 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.



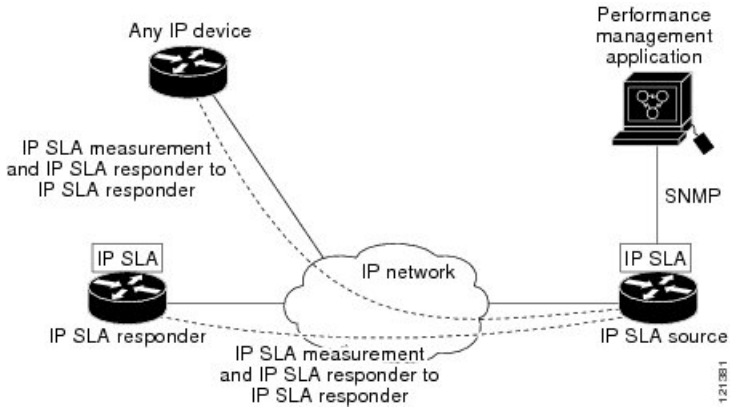
IP SLA Responder and IP SLA Control Protocol

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 4: 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).

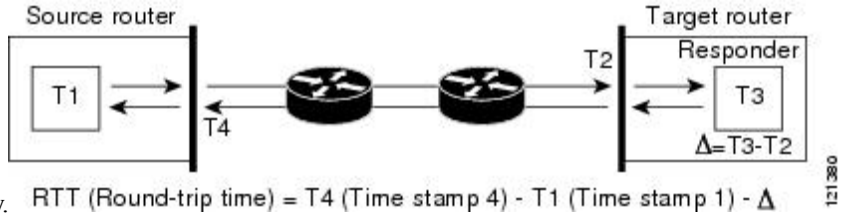
Response Time Computation for IP SLAs

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 5: 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*.

When you try to reconfigure a scheduled IP SLA operation, the active status of the IP SLA operation is checked. If the operation is active, the following message is displayed.

```
Entry already running and cannot be modified.
(only can delete (no) and start over)
(check to see if the probe has finished exiting)
```

Starting with the Cisco IOS XE 17.14.1 release, you can reconfigure the parameters of a scheduled IP SLA operation. You can use the **configure replace** command to replace the current running configuration with a new configuration. The **configure replace** command is a general command and is not specific to IP SLAs. The command will replace the entire current configuration with the new configuration provided. The IP SLA operation will be stopped, and restarted with the new parameters. You cannot change the IP SLA probe type or IP SLA socket related parameters such as the destination IP address, source IP address, destination port and source port using the **configure replace** command.

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* .

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:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# config t`	Enters the 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	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
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.

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# config t`	Enters the 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*]	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).

	Command or Action	Purpose
	Example: `Device(config-ip-sla)# udp-jitter 172.29.139.134 5000 source-ip 172.29.139.140 source-port 4000`	• *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. • (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.

	Command or Action	Purpose
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:** `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. 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.

	Command or Action	Purpose
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 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: 1.1.1.1/0.0.0.0
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# config t`	Enters the 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

	Command or Action	Purpose
		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. • (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**]	Configures the scheduling parameters for an individual IP SLA operation.

	Command or Action	Purpose
	Example: `Device(config)# ip sla schedule 10 start-time now life forever`	• *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. 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:**	(Optional) Saves your entries in the configuration file.

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

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: 1.1.1.1/0.0.0.0
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# config terminal`	Enters the 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	**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.

	Command or Action	Purpose
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**—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.

	Command or Action	Purpose
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 10
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 10 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 7: Monitoring IP SLA Operations

show ip sla application	Displays global information a
show ip sla authentication	Displays IP SLA authenticatio
show ip sla configuration [*entry-number*]	Displays configuration values operations or a specific operat
show ip sla enhanced-history {**collection-statistics** | **distribution statistics**} [*entry-number*]	Displays enhanced history sta or distribution statistics for all 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

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

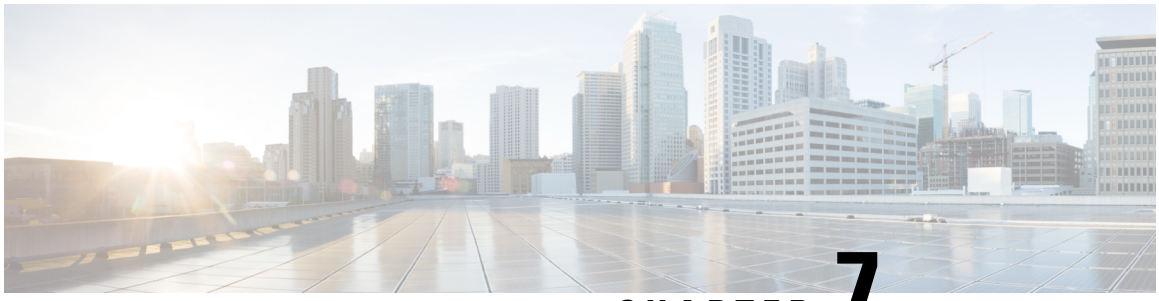
Feature History for Service Level Agreements

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 notedotherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	Service Level Agreements	Cisco IOS IP Service Level Agreements 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 XE Cupertino 17.7.1	Service Level Agreements	This feature was implemented on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

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 7

Configuring SPAN and RSPAN

- Prerequisites for SPAN and RSPAN, on page 89
- Restrictions for SPAN and RSPAN, on page 89
- Information About SPAN and RSPAN, on page 91
- Configuring SPAN and RSPAN, on page 101
- How to Configure SPAN and RSPAN, on page 102
- Monitoring SPAN and RSPAN Operations, on page 124
- Configuration Examples for SPAN and RSPAN, on page 124
- Feature History for SPAN and RSPAN, on page 126

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.

  On the Cisco Catalyst 9600 Series Supervisor 2 Module, encapsulation replicate is enabled by default and no other encapsulation methods are supported.
- 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.
- Cisco Catalyst 9600 Series Supervisor 2 Module does not support the **filter vlan** keyword.
- You cannot configure etherchannel member ports as source ports on Cisco Catalyst 9600 Series Supervisor 2 Module.
- Cisco Catalyst 9600 Series Supervisor 2 Module does not support an EtherChannel group as a destination port.
- On the Cisco Catalyst 9600 Series Supervisor 2 Module, TX source port configurations are supported only for traffic bridged within the same VLAN. TX source port configurations are not supported for traffic routed between VLANs.

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.
- A source cannot be a part of more than one SPAN session on Cisco Catalyst 9600 Series Supervisor 2 Module..
- 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 Cisco Catalyst 9600 Series Supervisor 1 Module..

  Adding multiple destination ports is not supported on Cisco Catalyst 9600 Series Supervisor 2 Module.
- 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.

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

The following sections provide 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.

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

Figure 6: 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.

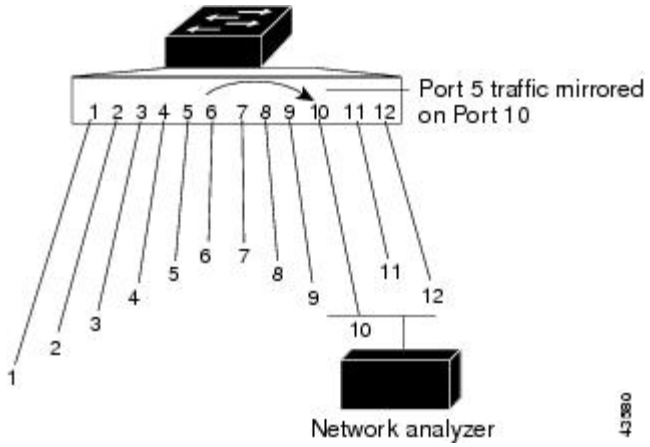
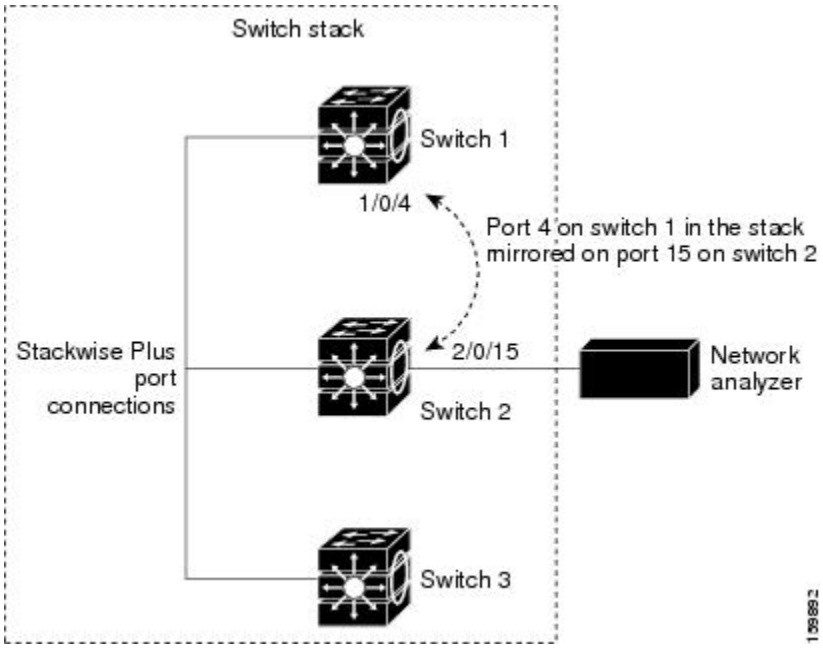


Figure 7: Example of Local SPAN Configuration on a Device Stack

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



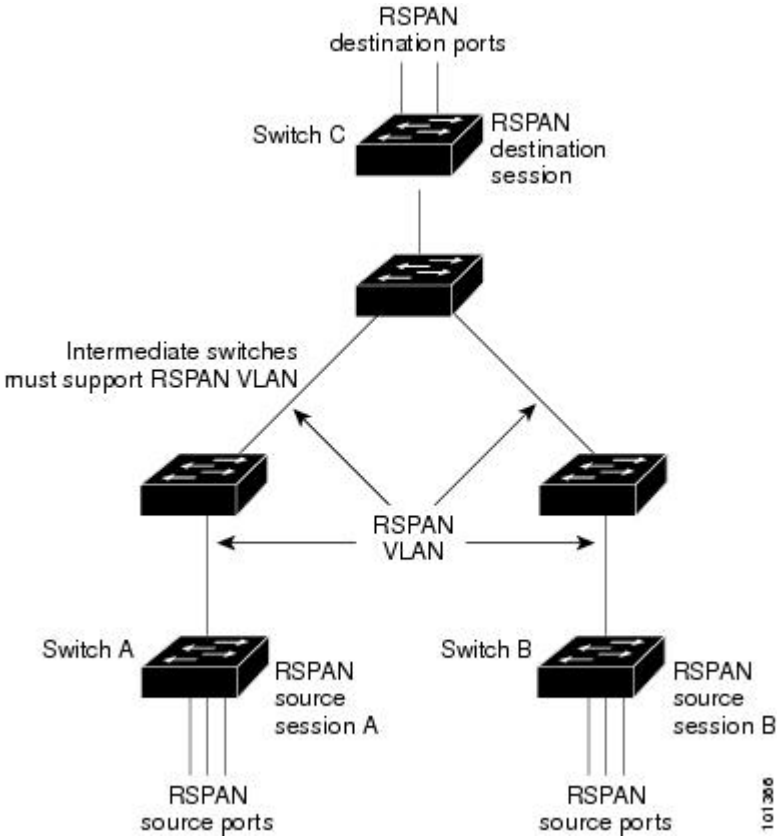
Remote SPAN

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 8: 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.

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).
- 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.

Note Ingress forwarding is not supported on Cisco Catalyst 9600 Series Supervisor 2 Module.

- 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.

- Starting with the Cisco IOS XE Cupertino 17.9.1 release, DHCP Snooping and Local SPAN can be configured on the same VLAN for non-SDA deployments.

SPAN and RSPAN and Device Stacks

Because the stack of switches represents one logical switch, local SPAN source ports and destination ports can be in different switches in the stack. Therefore, the addition or deletion of switches 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 switch is removed from the stack or an inactive session can become active when a switch 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 8: 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.

Configuring SPAN and RSPAN

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* {**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 device.
- 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

The following sections provide information on 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 or an EtherChannel. 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:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device# copy running-config 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*}]}	Specifies the SPAN session, the destination port, the packet encapsulation, and the ingress VLAN and encapsulation.

	Command or Action	Purpose

	Example:
	
	`Device(config)# monitor session 2 destination interface gigabitethernet1/0/2 encapsulation replicate ingress dot1q vlan 6`
		• 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 or an EtherChannel. 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.

	Command or Action	Purpose
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 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.

	Command or Action	Purpose
	`Device(config)# monitor session 2 filter vlan 1 - 5 , 9`	• 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** {**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 or an EtherChannel. 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:**	(Optional) Saves your entries in the configuration file.

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

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.

	Command or Action	Purpose
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

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.

	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* | **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.

	Command or Action	Purpose
	`Device(config)# monitor session 1 destination remote vlan 100`	• For *session_number*, enter the number defined in Step 4. • 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:**	Removes any existing SPAN configuration for the session. • For *session_number*, the range is 1 to 66.

	Command or Action	Purpose
	`Device(config)# no monitor session 2`	• **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:** `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:** `Device(config-vlan)# exit`	Returns to global configuration mode.

	Command or Action	Purpose
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:** `Device(config)# monitor session 2 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.

	Command or Action	Purpose
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**] **Example:**	Specifies the SPAN session and the source port (monitored port). • For *session_number*, the range is 1 to 66.

	Command or Action	Purpose
	`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.

	Command or Action	Purpose
		The destination interface must be a physical port or an EtherChannel. 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 9: Monitoring SPAN and RSPAN Operations

Command	Purpose
show monitor	Displays the current SPAN, R configuration.

Configuration Examples for SPAN and RSPAN

The following sections provide configuration examples for SPAN and RSPAN

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
```

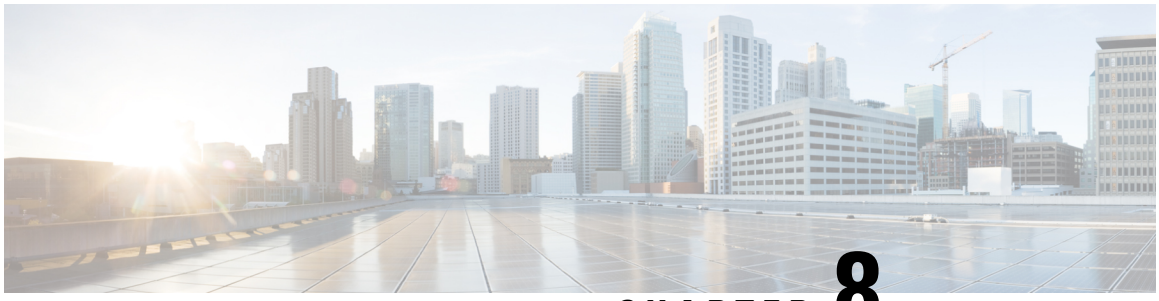
Feature History for SPAN and RSPAN

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 notedotherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	Switch Port Analyzer (SPAN)	Switch Port Analyzer allows monitoring of device traffic on a port or VLAN using a sniffer/analyzer or RMON probe.
	Flow-based Switch Port Analyzer (SPAN)	Flow-based 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.
	Switch Port Analyzer (SPAN) - distributed egress SPAN	The feature 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.
Cisco IOS XE Cupertino 17.7.1	Switch Port Analyzer (SPAN)	Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2) .

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 8

Configuring ERSPAN

- Prerequisites for Configuring ERSPAN, on page 129
- Restrictions for Configuring ERSPAN, on page 129
- Information about Configuring ERSPAN, on page 130
- How to Configure ERSPAN, on page 133
- Configuration examples for ERSPAN, on page 143
- Verifying ERSPAN, on page 144
- Additional References, on page 146
- Feature History for ERSPAN, on page 147

Prerequisites for Configuring ERSPAN

- Apply the Access control list (ACL) filter before sending the monitored traffic on to the tunnel.

Restrictions for Configuring ERSPAN

The following restrictions apply for this feature:

- Truncation is supported only on IPv4 and IPv6 spanned packets and not on Layer 2 packets without an IP header.
- 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 commands to remove the session and then reconfigure it.
- ERSPAN source sessions do not copy locally-sourced ERSPAN Generic routing encapsulation (GRE)-encapsulated traffic from source ports.
- Disabling the **ip routing** command for IPv4 connections and **ipv6 unicast-routing** command for IPv6 connections stops ERSPAN traffic flow to the destination port.
- ERSPAN over MPLS VPN is supported on Layer 3 VPNs, Segment Routing and Seamless MPLS.
- ERSPAN over MPLS VPN is not supported for L2VPN, 6PE, 6VPE, MPLS over GRE and InterAS.

- You cannot configure an MPLS core switch as an ERSPAN destination. ERSPAN traffic can be transported from one Provider Edge (PE) to another Provide Edge (PE). It cannot be transported to the core switch between the two Provider Edges.
- ERSPAN 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.
- If a backup configuration having ERSPAN session enabled is restored to the running configuration, ERSPAN sessions are created automatically in disabled state. You must manually enable these ERSPAN sessions.
- ERSPAN does not support QoS shaping and policing.

Apart from the restrictions listed above, the following additional restrictions apply to Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2):

- ERSPAN destination type/session is not supported.
- Filter-based ERSPAN is not supported.
- RSPAN is not supported.
- ERSPAN session do not capture CPU-originated traffic.
- ERSPAN source session in TX/egress direction is not supported.
- MTU size can be set up as 344 and 9000 bytes only. Any values between 344 and 9000 are not allowed.
- IPv4 Time to Live (TTL) and IPv4 Differentiated Services Code Point (DSCP) options are not supported.
- IPv6 TTL, IPv6 DSCP, and IPv6 flow label options are not supported.
- A source cannot be a part of more than one ERSPAN session.

Information about Configuring ERSPAN

The following sections provide information about 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 help 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.

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. 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. To avoid over-subscription of traffic,

which can lead to drop in management traffic on the destination device, ensure that the destination session is configured and is working on the destination device, before configuring a source session on the source 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.

Note ERSPAN does not support monitoring multicast traffic on Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

A device supports up to 66 sessions. A maximum of eight 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. The number of source sessions decreases by the number of configured ERSPAN destination sessions.

A device can support a maximum of 50 Security Group Tag (SGT) filter per session.

An ERSPAN source session is defined by the following parameters:

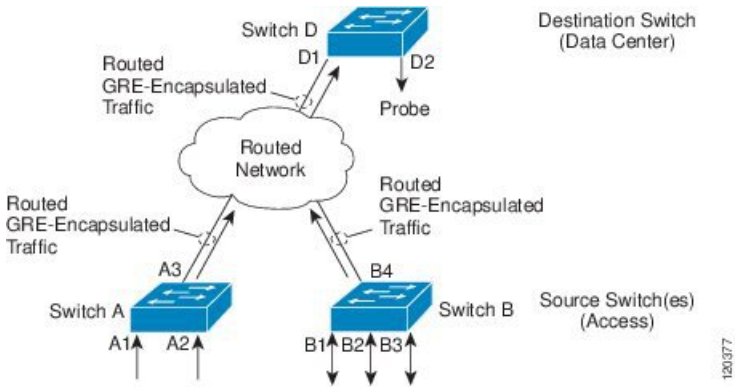
- A session ID.
- ERSPAN flow ID.
- List of source ports or source VLANs that are monitored by the session.
- Optional attributes, such as, IP type of service (ToS) and IP Time to Live (TTL), related to the Generic Routing Encapsulation (GRE) envelope.
- The destination and origin IP addresses. These are used as the destination and source IP addresses of the GRE envelope for the captured traffic, respectively.

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.
- Because encapsulation and decapsulation are performed in the hardware, the CPU performance is not impacted.
- IPv4 and IPv6 delivery and transport headers are supported; including Type-II and Type-III headers.

  Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2) do not support Type-III headers.

Figure 9: 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.

ERSPAN Destination Ports

A destination port is a Layer 2 or Layer 3 LAN port to which ERSPAN source sends traffic for analysis.

When you configure a port as a destination port, it can no longer receive any traffic. The port is dedicated for use only by the ERSPAN feature. An ERSPAN destination port does not forward any traffic except that required for the ERSPAN session. You can configure trunk ports as destination ports, which allows destination trunk ports to transmit encapsulated traffic.

SGT Based ERSPAN

A Security Group Tag (SGT) is a 16-bit value that the Cisco Identity Services Engine (ISE) assigns to the user or endpoint session upon login. The network infrastructure views the SGT as another attribute to assign to the session and inserts the Layer 2 tag to all traffic from that session. A platform can support a maximum of 50 SGT policies per session.

On an existing flow-based SPAN (FSPAN) or VLAN filter session, SGT filtering configurations are not allowed.

Note SGT filtering configuration is not supported on Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

ERSPAN Timestamp

ERSPAN Timestamp is automatically enabled when the ERSPAN header is set to type III. The timestamp field is used to calculate packet latency in devices. The ERSPAN source session fills in the timestamp field with local time information when a packet is received. The destination session can hand over this timestamp to the application. ERSPAN supports all timestamps in 32-bit format. It supports 100 nanosecond (ns) granularity and the timestamp field wraparound time is around 7 minutes.

ERSPAN over MPLS VPN

Starting with the Cisco IOS XE Bengaluru 17.5.x release, you can transport ERSPAN traffic over a Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN). To enable ERSPAN over MPLS VPN you will have to enable Multiprotocol Label Switching (MPLS), Label Distribution Protocol (LDP), and Cisco Express Forwarding in your network.

You can configure the ERSPAN destination to select the source VRF for the ERSPAN traffic over MPLS VPN. You can use the **vrf** keyword in the ERSPAN destination session source command to configure the source VRF.

How to Configure ERSPAN

The following sections provide information about how to configure ERSPAN.

Configuring an ERSPAN Source Session (IPv4)

The ERSPAN source session defines the session configuration parameters and the ports or VLANs to be monitored. To define an IPv4 ERSPAN source session, complete the following procedure:

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	**monitor session** *span-session-number* **type erspan-source** **Example:** `Device(config)# monitor session 1 type erspan-source`	Defines an ERSPAN source session using the session ID and the session type, and enters ERSPAN monitor source session configuration mode. • The *span-session-number* argument range is from 1 to 66. The same session number cannot be used more than once.

	Command or Action	Purpose
		• The 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 session ID (configured by the *span-session-number* argument) and the session type (configured by the **erspan-source** keyword) cannot be changed once entered. 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** *string* **Example:** `Device(config-mon-erspan-src)# description source1`	(Optional) Describes the ERSPAN source session. • The *string* argument can be up to 240 characters and cannot contain special characters or spaces.
Step 5	[**no**] **header-type 3** **Example:** `Device(config-mon-erspan-src)# header-type 3`	(Optional) Configures a switch to Type-III ERSPAN header. The default type is Type-II ERSPAN header. **Note** Type-III headers are not supported on Cisco Catalyst 9600 Series Supervisor 2 Module.
Step 6	**source** {**interface** *interface-type interface-number* | **vlan** *vlan-id*} [**,** | **-** | **both** | **rx** | **tx**] **Example:** `Device(config-mon-erspan-src)# source interface fastethernet 0/1 rx`	Configures the source interface or the VLAN, and the traffic direction to be monitored. **Note** The **both** and **tx** options are not supported on Cisco Catalyst 9600 Series Supervisor 2 Module.
Step 7	**filter** {**ip access-group** {*standard-access-list* | *expanded-access-list* | *acl-name* } | **ipv6 access-group** *acl-name* | **mac access-group** *acl-name* | **sgt** *sgt-ID* [**,** | **-**] | **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. The **filter sgt** *sgt-ID* command configures SGT filtering in the ERSPAN source session. **Note** • You cannot include source VLANs and filter VLANs in the same session. • Flow-based SPAN (FSPAN) session and SGT filtering configuration are not supported on Cisco Catalyst 9600 Series Supervisor 2 Module

	Command or Action	Purpose
Step 8	**destination** **Example:** `Device(config-mon-erspan-src)# destination`	Enters ERSPAN source session destination configuration mode.
Step 9	**erspan-id** *erspan-flow-id* **Example:** `Device(config-mon-erspan-src-dst)# erspan-id 100`	Configures the ID used by source and destination sessions to identify the ERSPAN traffic, which must also be entered in the ERSPAN destination session configuration.
Step 10	**ip address** *ip-address* **Example:** `Device(config-mon-erspan-src-dst)# ip address 10.1.0.2`	Configures the IP address that is used as the destination of the ERSPAN traffic.
Step 11	**ip dscp** *dscp-value* **Example:** `Device(config-mon-erspan-src-dst)# ip dscp 10`	(Optional) Enables the use of IP differentiated services code point (DSCP) for packets that originate from a circuit emulation (CEM) channel. **Note** The **ip dscp** command is not supported on Cisco Catalyst 9600 Series Supervisor 2 Module.
Step 12	**ip ttl** *ttl-value* **Example:** `Device(config-mon-erspan-src-dst)# ip ttl 32`	(Optional) Configures the IP TTL value of packets in the ERSPAN traffic. **Note** The **ip ttl** command is not supported on Cisco Catalyst 9600 Series Supervisor 2 Module.
Step 13	**mtu** *mtu-size* **Example:** `Device(config-mon-erspan-src-dst)# mtu 512`	Configures the MTU size for truncation. Any ERSPAN packet that is larger than the configured MTU size is truncated to the configured size. The MTU size range is 176 to 9000 bytes. The default value is 9000 bytes. On Cisco Catalyst 9600 Series Supervisor 2 Module, MTU size can be set up as 344 and 9000 bytes only.
Step 14	**origin ip-address** *ip-address* **Example:** `Device(config-mon-erspan-src-dst)# origin ip address 10.10.0.1`	Configures the IP address used as the source of the ERSPAN traffic.
Step 15	**vrf** *vrf-id* **Example:**	(Optional) Configures the VRF name to use instead of the global routing table.

	Command or Action	Purpose
	`Device(config-mon-erspan-src-dst)# vrf 1`	
Step 16	**exit** **Example:** `Device(config-mon-erspan-src-dst)# exit`	Exits ERSPAN source session destination configuration mode, and returns to ERSPAN source session configuration mode.
Step 17	**no shutdown** **Example:** `Device(config-mon-erspan-src)# no shutdown`	Enables the configured sessions on an interface.
Step 18	**end** **Example:** `Device(config-mon-erspan-src)# end`	Exits ERSPAN source session configuration mode, and returns to privileged EXEC mode.

Configuring an ERSPAN Destination Session (IPv4)

The ERSPAN destination session defines the session configuration parameters and the ports that receive the monitored traffic.

Note Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2) do not support IPv4 ERSPAN destination type/session.

To define an IPv4 ERSPAN destination session, complete the following procedure:

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	**monitor session** *session-number* **type erspan-destination** **Example:** `Device(config)# monitor session 1 type erspan-destination`	Defines an ERSPAN destination session using the session ID and the session type, and enters ERSPAN monitor destination session configuration mode. • The *session-number* argument range is from 1 – 66. The session number must be

	Command or Action	Purpose
		unique and cannot be used more than once. • The 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 session ID (configured by the *session-number* argument) and the session type (configured by the **erspan-destination**) cannot be changed once entered. Use the **no** form of this command to remove the session, and then recreate the session with a new session ID or a new session type.
Step 4	**description** *string* **Example:** `Device(config-mon-erspan-dst)# description source1`	(Optional) Describes the ERSPAN destination session. • The *string* argument can be up to 240 characters in length and cannot contain special characters or spaces.
Step 5	**destination interface** *interface-type interface-number* **Example:** `Device(config-mon-erspan-dst)# destination interface GigabitEthernet1/0/1`	Associates the ERSPAN destination session number with source ports, and selects the traffic direction to be monitored.
Step 6	**source** **Example:** `Device(config-mon-erspan-dst)# source`	Enters ERSPAN destination session source configuration mode.
Step 7	**erspan-id** *erspan-flow-id* **Example:** `Device(config-mon-erspan-dst-src)# erspan-id 100`	Configures the ID used by source and destination sessions to identify the ERSPAN traffic, which must also be entered in the ERSPAN source session configuration.
Step 8	**ip address** *ip-address* [**force**] **Example:** `Device(config-mon-erspan-dst-src)# ip address 10.1.0.2`	Configures the IP address that is used as the destination of the ERSPAN traffic. • This IP address must be an address on a local interface or loopback interface, and match the address on the destination switch. • The **ip address** *ip-address* **force** command changes the destination IP

	Command or Action	Purpose
		address for all ERSPAN destination sessions.
Step 9	**vrf** *vrf-id* **Example:** `Device(config-mon-erspan-dst-src)# vrf 1`	(Optional) Configures the VRF name to use instead of the global routing table.
Step 10	**no shutdown** **Example:** `Device(config-mon-erspan-dst-src)# no shutdown`	Enables the configured sessions on an interface.
Step 11	**end** **Example:** `Device(config-mon-erspan-dst-src)# end`	Exits ERSPAN destination session source configuration mode, and returns to privileged EXEC mode.

Configuring an ERSPAN Source Session (IPv6)

The ERSPAN source session defines the session configuration parameters and the ports or VLANs to be monitored. To define an IPv6 ERSPAN source session, complete the following procedure:

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	**monitor session** *session-number* **type erspan-source** **Example:** `Device(config)# monitor session 1 type erspan-source`	Defines an ERSPAN source session using the session ID and the session type, and enters ERSPAN monitor source session configuration mode. • The *span-session-number* range is from 1 to 66. The same session number cannot be used more than once. • The 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.

	Command or Action	Purpose
		• The session ID (configured by the *span-session-number* argument) and the session type (configured by the **erspan-source** keyword) cannot be changed once entered. 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** *string* **Example:** `Device(config-mon-erspan-src)# description source1`	(Optional) Describes the ERSPAN source session. • The *string* argument can be up to 240 characters and cannot contain special characters or spaces.
Step 5	[**no**] **header-type 3** **Example:** `Device(config-mon-erspan-src)# header-type 3`	(Optional) Configures a switch to Type-III ERSPAN header. The default type is Type-II ERSPAN header. **Note** Type-III headers are not supported on Cisco Catalyst 9600 Series Supervisor 2 Module.
Step 6	**source** {**interface** *interface-type interface-number* | **vlan** *vlan-id*} [**,** | **-** | **both** | **rx** | **tx**] **Example:** `Device(config-mon-erspan-src)# source interface fortygigabitethernet 1/0/3`	Configures the source interface or the VLAN, and the traffic direction to be monitored. **Note** The **both** and **tx** options are not supported on Cisco Catalyst 9600 Series Supervisor 2 Module.
Step 7	**filter** {**ip access-group** {*standard-access-list* | *expanded-access-list* | *acl-name* } | **ipv6 access-group** *acl-name* | **mac access-group** *acl-name* | **sgt** *sgt-ID* [**,** | **-**] | **vlan** *vlan-ID* [**,** | **-**]} **Example:** `Switch(config-mon-erspan-src)# filter ipv6 access-group exampleacl`	(Optional) Configures source VLAN filtering when the ERSPAN source is a trunk port. The **filter sgt** *sgt-ID* command configures SGT filtering in the ERSPAN source session. **Note** • You cannot include source VLANs and filter VLANs in the same session. • Flow-based SPAN (FSPAN) session and SGT filtering configuration are not supported on Cisco Catalyst 9600 Series Supervisor 2 Module
Step 8	**destination** **Example:** `Device(config-mon-erspan-src)# destination`	Enters ERSPAN source session destination configuration mode.

	Command or Action	Purpose
Step 9	**erspan-id** *erspan-flow-id* **Example:** `Device(config-mon-erspan-src-dst)# erspan-id 100`	Configures the ID used by source and destination sessions to identify the ERSPAN traffic, which must also be entered in the ERSPAN destination session configuration.
Step 10	**ipv6 address** *ipv6-address* **Example:** `Device(config-mon-erspan-src-dst)# ipv6 address 2001:DB8::1`	Configures the IPv6 address that is used as the destination of the ERSPAN traffic.
Step 11	**ipv6 dscp** *dscp-value* **Example:** `Device(config-mon-erspan-src-dst)# ipv6 dscp 2`	(Optional) Enables the use of IPv6 differentiated services code point (DSCP) for packets that originate from a circuit emulation (CEM) channel. **Note** The **ipv6 dscp** command is not supported on Cisco Catalyst 9600 Series Supervisor 2 Module.
Step 12	**ipv6 ttl** *ttl-value* **Example:** `Device(config-mon-erspan-src-dst)# ipv6 ttl 4`	(Optional) Configures the IPv6 TTL value of packets in the ERSPAN traffic. **Note** The **ipv6 ttl** command is not supported on Cisco Catalyst 9600 Series Supervisor 2 Module.
Step 13	**ipv6 flow-label** *flow-label-value* **Example:** `Device(config-mon-erspan-src-dst)# ipv6 flow-label 2`	(Optional) Configures the IPv6 flow label in the ERSPAN traffic. *flow-label-value*: The range is from 0 to 1048575. **Note** The **ipv6 flow-label** command is not supported on Cisco Catalyst 9600 Series Supervisor 2 Module.
Step 14	**mtu** *mtu-size* **Example:** `Device(config-mon-erspan-src-dst)# mtu 512`	Configures the MTU size for truncation. Any ERSPAN packet that is larger than the configured MTU size is truncated to the configured size. The MTU size range is 176 to 9000 bytes. The default value is 9000 bytes. On Cisco Catalyst 9600 Series Supervisor 2 Module, MTU size can be set up as 344 and 9000 bytes only.

	Command or Action	Purpose
Step 15	**origin ipv6-address** *ipv6-address* **Example:** `Device(config-mon-erspan-src-dst)# origin ipv6 address 2001:DB8:1::1`	Configures the IPv6 address used as the source of the ERSPAN traffic.
Step 16	**vrf** *vrf-id* **Example:** `Device(config-mon-erspan-src-dst)# vrf 1`	(Optional) Configures the VRF name to use instead of the global routing table.
Step 17	**exit** **Example:** `Device(config-mon-erspan-src-dst)# exit`	Exits ERSPAN source session destination configuration mode, and returns to ERSPAN source session configuration mode.
Step 18	**no shutdown** **Example:** `Device(config-mon-erspan-src)# no shutdown`	Enables the configured sessions on an interface.
Step 19	**end** **Example:** `Device(config-mon-erspan-src)# end`	Exits ERSPAN source session configuration mode, and returns to privileged EXEC mode.

Configuring an ERSPAN Destination Session (IPv6)

The ERSPAN destination session defines the session configuration parameters and the ports that receives the monitored traffic. To define an IPv6 ERSPAN destination session, complete the following procedure:

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	**monitor session** *session-number* **type erspan-destination** **Example:** `Device(config)# monitor session 3 type erspan-destination`	Defines an ERSPAN destination session using the session ID and the session type, and enters ERSPAN monitor destination session configuration mode.

	Command or Action	Purpose
		• The *session-number* argument range is from 1 to 66. The session number must be unique and cannot be used more than once. • The 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 session ID (configured by the *session-number* argument) and the session type (configured by the **erspan-destination**) cannot be changed once entered. Use the **no** form of this command to remove the session, and then recreate the session with a new session ID or a new session type.
Step 4	**description** *string* **Example:** `Device(config-mon-erspan-dst)# description source 1`	(Optional) Describes the ERSPAN destination session. • The *string* argument can be up to 240 characters in length and cannot contain special characters or spaces.
Step 5	**destination interface** *interface-type interface-number* **Example:** `Device(config-mon-erspan-dst)# destination interface fortygigabitethernet 1/0/3`	Associates the ERSPAN destination session number with source ports, and selects the traffic direction to be monitored.
Step 6	**source** **Example:** `Device(config-mon-erspan-dst)# source`	Enters ERSPAN destination session source configuration mode.
Step 7	**erspan-id** *erspan-flow-id* **Example:** `Device(config-mon-erspan-dst-src)# erspan-id 100`	Configures the ID used by source and destination sessions to identify the ERSPAN traffic, which must also be entered in the ERSPAN source session configuration.
Step 8	**ipv6 address** *ipv6-address* **Example:** `Device(config-mon-erspan-dst-src)# ip address 2001:DB8::1`	Configures the IPv6 address that is used as the destination of the ERSPAN traffic. This IPv6 address must be an address on a local interface or loopback interface, and match the address on the destination switch.

	Command or Action	Purpose
Step 9	**exit** **Example:** `Switch(config-mon-erspan-dst-src)#exit`	Exits ERSPAN destination session source configuration mode, and returns to ERSPAN destination session configuration mode.
Step 10	**no shutdown** **Example:** `Device(config-mon-erspan-dst)# no shutdown`	Enables the configured sessions on an interface.
Step 11	**end** **Example:** `Device(config-mon-erspan-dst)# end`	Exits ERSPAN destination session source configuration mode, and returns to privileged EXEC mode.

Configuration examples for ERSPAN

The following sections provide configuration examples for ERSPAN.

Example: Configuring an ERSPAN Source Session

The following example shows how to configure an ERSPAN source session:

```
Device> enable
Device# configure terminal
Device(config)# monitor session 1 type erspan-source
Device(config-mon-erspan-src)# description source1
Device(config-mon-erspan-src)# source interface GigabitEthernet 1/0/1 rx
Device(config-mon-erspan-src)# source interface GigabitEthernet 1/0/4 - 8 tx
Device(config-mon-erspan-src)# source interface GigabitEthernet 1/0/3
Device(config-mon-erspan-src)# destination
Device(config-mon-erspan-src-dst)# erspan-id 100
Device(config-mon-erspan-src-dst)# ip address 10.1.0.2
Device(config-mon-erspan-src-dst)# ip dscp 10
Device(config-mon-erspan-src-dst)# ip ttl 32
Device(config-mon-erspan-src-dst)# mtu 512
Device(config-mon-erspan-src-dst)# origin ip address 10.10.0.1
Device(config-mon-erspan-src-dst)# vrf monitoring
Device(config-mon-erspan-src-dst)# exit
Device(config-mon-erspan-src)# no shutdown
Device(config-mon-erspan-src)# end
```

Example: Configuring an ERSPAN Destination Session

The following example shows how to configure an ERSPAN destination session:

```
Device(config)# monitor session 2 type erspan-destination
Device(config-mon-erspan-dst)# destination interface GigabitEthernet1/3/2
Device(config-mon-erspan-dst)# destination interface GigabitEthernet2/2/0
Device(config-mon-erspan-dst)# source
```

```
Device(config-mon-erspan-dst-src)# erspan-id 100
Device(config-mon-erspan-dst-src)# ip address 10.1.0.2
```

The following example shows how to configure a source VRF for an ERSPAN destination session:

```
Device(config)# monitor session 2 type erspan-destination
Device(config-mon-erspan-dst)# destination interface GigabitEthernet1/3/2
Device(config-mon-erspan-dst)# destination interface GigabitEthernet2/2/0
Device(config-mon-erspan-dst)# source
Device(config-mon-erspan-dst-src)# erspan-id 100
Device(config-mon-erspan-dst-src)# ip address 10.1.0.2
Device(config-mon-erspan-dst-src)# vrf 1
```

Verifying ERSPAN

To verify the ERSPAN configuration, use the following commands:

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

```
Device# show monitor session 53

Session 53
----------
Type                    : ERSPAN Source Session
Status                  : Admin Enabled
Source Ports            :
MTU                     : Fo1/0/2
```

The following is sample output from the **show platform software monitor session** command:

```
Device# show platform software monitor session 53

Span Session 53 (FED Session 0):
Type: ERSPAN Source
Prev type: Unknown
Ingress Src Ports:
Egress Src Ports:
Ingress Local Src Ports: (null)
Egress Local Src Ports: (null)
Destination Ports:
Ingress Src Vlans:
Egress Src Vlans:
Ingress Up Src Vlans: (null)
Egress Up Src Vlans: (null)
Src Trunk filter Vlans:
RSPAN dst vlan: 0
RSPAN src vlan: 0
RSPAN src vlan sav: 0
Dest port encap = 0x0000
Dest port ingress encap = 0x0000
Dest port ingress vlan = 0x0
SrcSess: 1 DstSess: 0 DstPortCfgd: 0 RspnDstCfg: 0 RspnSrcVld: 0
DstCliCfg: 0 DstPrtInit: 0 PsLclCfgd: 0
Flags: 0x00000000
Remote dest port: 0 Dest port group: 0
FSPAN disabled
FSPAN not notified
ERSPAN Id : 0
```

```
ERSPAN Org Ip: 0.0.0.0
ERSPAN Dst Ip: 0.0.0.0
ERSPAN Ip Ttl: 255
ERSPAN DSCP : 0
ERSPAN MTU : 1500 >>>>
ERSPAN VRFID : 0
ERSPAN State : Disabled
ERSPAN Tun id: 61
ERSPAN header-type: 2
ERSPAN SGT :
```

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

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

Type                     : ERSPAN Source Session
Status                   : Admin Enabled
Description              : -
Source Ports             :
    RX Only              : None
    TX Only              : None
    Both                 : None
Source Subinterfaces     :
    RX Only              : None
    TX Only              : None
    Both                 : None
Source VLANs             :
    RX Only              : None
    TX Only              : None
    Both                 : None
Source Drop-cause        : None
Source EFPs              :
    RX Only              : None
    TX Only              : None
    Both                 : None
Source RSPAN VLAN        : None
Destination Ports        : None
Filter VLANs             : None
Filter SGT               : None
Dest RSPAN VLAN          : None
IP Access-group          : None
MAC Access-group         : None
IPv6 Access-group        : None
Filter access-group  :None
smac for wan interface   : None
dmac for wan interface   : None
Destination IP Address   : 192.0.2.1
Destination IPv6 Address : None
Destination IP VRF       : None
MTU                      : 1500
Destination ERSPAN ID    : 251
Origin IP Address        : 10.10.10.216
Origin IPv6 Address      : None
IP QOS PREC              : 0
IPv6 Flow Label          : None
IP TTL                   : 255
ERSPAN header-type       : 3
```

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

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

```
ERSPAN Source Session:ERSPAN Source Session Supported: TRUE
No of Rx ERSPAN source session: 8
No of Tx ERSPAN source session: 8
ERSPAN Header Type supported: II and III
ACL filter Supported: TRUE
SGT 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-source** command displays information about the configured ERSPAN source sessions for Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2):

```
Device#show capability feature monitor erspan-source
ERSPAN Source Session: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: FALSE
SGT filter Supported: FALSE
Fragmentation Supported: TRUE
Truncation Supported: TRUE
Sequence number Supported: FALSE
QOS Supported: FALSE
```

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

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

ERSPAN Destination Session:ERSPAN Destination Session Supported: TRUE
Maximum No of ERSPAN destination session: 8
ERSPAN Header Type supported: II and III
```

Since ERSPAN destination session/type is not supported on Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2), the following is the output from the **show capability feature monitor erspan-destination**:

```
Device# show capability feature monitor erspan-destination
ERSPAN Destination Session: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

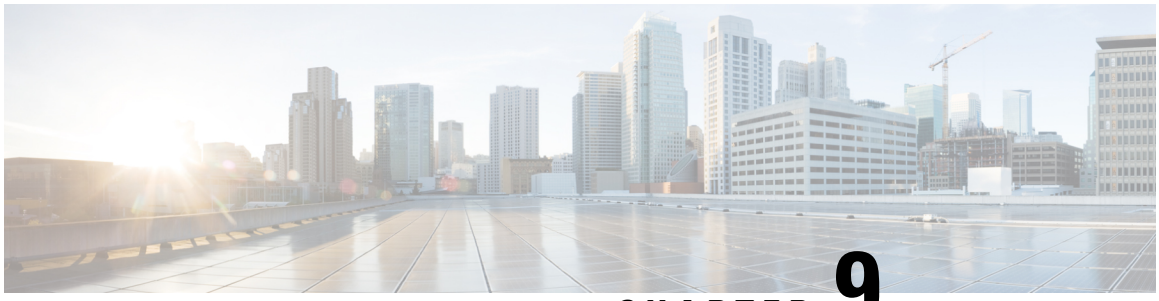
Feature History for ERSPAN

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 Gibraltar 16.11.1	ERSPAN	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.
Cisco IOS XE Amsterdam 17.1.1	ERSPAN IPv6	IPv6 support was introduced for ERSPAN. This enables configuration of an IPv6 ERSPAN source and destination session.
Cisco IOS XE Bengaluru 17.5.1	ERSPAN over MPLS VPN	MPLS VPN support was introduced for ERSPAN. ERSPAN traffic can be transported over an MPLS VPN. Support for ERSPAN is limited to L3VPN IPV4 MPLS.

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 9

Configuring Packet Capture

- Prerequisites for Configuring Packet Capture, on page 149
- Restrictions for Configuring Packet Capture, on page 150
- Information About Packet Capture, on page 152
- How to Configure Packet Capture, on page 160
- Configuration Examples for Packet Capture, on page 173
- Additional References, on page 190
- Feature History for Packet Capture, on page 190

Prerequisites for Configuring Packet Capture

Packet capture is supported on the Cisco Catalyst 9600 Series Switches.

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. It also depends 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. The following table provides some guidelines for using the system resources.

Table 10: System Requirements for the EPC Subsystem

System Resources	Requirements
Hardware	CPU utilization requirements are platform-dependent.

System Resources	Requirements
Memory	The DRAM stores the packet buffer. The size of the packet buffer is user specified.
Disk space	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

- Wireshark doesn't support Global packet capture.
- Wireshark doesn't support limiting circular file storage by file size.
- If you delete the file used by an active capture session, the capture session can't create a new file. All further packets captured are lost. You need to restart the capture point.
- File limit is limited to the size of the flash in DNA Advantage.
- Wireshark can't 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, restart the capture manually.
- Wireshark doesn't capture packets dropped by floodblock.
- A Wireshark class map allows only one ACL (IPv4, IPv6, or MAC).
- ACL logging and Wireshark are incompatible. Once you activate Wireshark, it takes priority. All traffic, including that captured by ACL logging on any ports, is 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 occurs 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 is 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 aren’t synchronized to the standby Supervisor in NSF and SSO scenarios.

Embedded Wireshark is supported with the following limitations:

- Capture filters and display filters aren't supported.
- Active capture decoding isn’t 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.
- Packet length range as a filter for packet capture isn't supported for non- IPv4/IPv6 packets and fragmented packets.
- Packet length range as a filter can't be used in addition with any other filters.

Restrictions for Configuring Embedded Packet Capture

The following restrictions are applicable to all models of the Cisco Catalyst 9600 Series Switches:

- Layer 2 EtherChannels aren't supported.
- You can't use VRFs, management ports, and private VLANs as attachment points.
- Embedded Packet Capture (EPC) isn't supported on logical ports, which includes port channels, switch virtual interfaces (SVIs), and subinterfaces. It's supported only on physical ports.
- A VLAN interface that is in shutdown state doesn't support EPC.
- If you change interface from switch port to routed port (Layer 2 to Layer 3) or the opposite way, you must delete the capture point and create a new one, once the interface comes back up. Stop/start the capture point won't work.
- Packets captured in the output direction of an interface might not reflect the changes made by the device rewrite. This 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's 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 doesn't capture the replicated packets on egress.
- The Rewrite information of both ingress and egress packets aren't captured.
- CPU-injected packets are considered control plane packets. Therefore, these types of packets won't be captured on an interface egress capture.
- Control plane packets aren't rate limited and performance impacting. Use filters to limit control plane packet capture.
- DNA Advantage supports decoding of protocols such as Control and Provisioning of Wireless Access Points (CAPWAP).
- You can define up to eight capture points, but only one can be active at a time. Stop one before you can start the other.
- MAC filter won't 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 won't be supported on a Layer 3 port or SVI.
- MAC filter can't capture Layer 2 packets (ARP) on Layer 3 interfaces.

- VACL doesn't support IPv6-based ACLs.
- EPC cannot capture based on the underlying routing protocols in MPLS packets.

The following restrictions are applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2):

- EPC is not suppored on port channel member interface.
- EPC is not supported on Locator/ID Separation Protocol (LISP) interface and tunnel interface.
- EPC is not supported with Ethernet-over-MPLS (EoMPLS).
- Hardware drop statistics are not available as no CPU queue is allocated for ACL logging.
- The following behavior is applicable for ACL logging with EPC on ingress and egress:
  - On ingress, if ACL logging, QoS ACL, and EPC ACL are configured at the same time, EPC ACL will not take priority.

    The following behaviors are applicable if both ACL logging and EPC ACLs are enabled:

    - ACL with deny logging: If a packet matches both EPC ACL and ACL with deny logging, then two copies will be sent to the CPU, one from EPC and one from ACL log.
    - ACL with permit logging: If a packet matches both EPC ACL and ACL with permit logging, only the copy from ACL log will be sent to the CPU.

  - On egress, if both EPC and ACL logging are configured, irrespective of the order in which they are configured, ACL logging will take priority and EPC will not work.

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. You can 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 the characteristics associated with a given instance of Wireshark. These include 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 don't 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.

In case of stacked systems, the capture point is activated on the active member. A switchover terminates any active packet capture session and you have to restart the session.

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. These are copied and passed to Wireshark. Wireshark displays a packet, if it passes through an attachment point, and all 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 to copy the hardware-forwarded traffic to software for Wireshark purposes.
- Capture filter—Wireshark applies the capture filter. 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 copied. They are 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 doesn't use the syntax of the capture filter.

- Display filter—Wireshark applies the display filter. Its match criteria are similar to the criteria of the capture filter. Packets that fail the display filter aren’t 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.

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 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

You can invoke Wireshark 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

The decode and display action is applicable only when invoked on a .pcap file.

Storage of Captured Packets to Buffer in Memory

You can store packets in the capture buffer in memory. You can use the packets for subsequent decoding, analysis, or storage to a .pcap file.

The capture buffer can be in linear or circular mode. In linear mode, when the buffer is full it discards new packets. In circular mode, if the buffer is full, it discards the older packets to accommodate the new packets. Although you can clear the buffer when needed, this mode is used for debugging network traffic. However, it's not possible to 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

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 you define a Wireshark capture point with its attachment points, filters, actions, and other options, you must activate it. Until you activate the capture point, it doesn't actually capture packets.

Before you activate a capture point, some functional checks are performed. A capture point can't be activated if a core system filter or attachment points aren't defined. Attempting to activate a capture point that doesn't meet these requirements generates an error.

The display filters are specified as needed.

After you activate Wireshark capture points, you can deactivate them in multiple ways. You can halt a capture point that is storing only packets to a .pcap file manually. You can also configure it with time or packet limits, after which the capture point halts automatically.

When you activate a Wireshark capture point, a fixed rate policer is applied automatically in the hardware. This ensures the CPU isn't flooded with Wireshark directed packets. The disadvantage of the rate policer is that you can't 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 three attachment points, the rates of all three attachment points together is policed to 1000 pps.

Note Policer isn't 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:

- 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 won't capture them.
- SPAN—Wireshark can't 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.

Guidelines for Configuring Wireshark

- During Wireshark packet capture, hardware forwarding happens concurrently.
- Because packet forwarding typically occurs in hardware, packets aren't 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, 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 little traffic matches the core filter.

- Writing to flash disk is a CPU-intensive operation. 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 don't specify the display option) while capturing live packets. Do not use it for 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 want to decode and display live packets in the console window, ensure that you bound the Wireshark session 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.
- 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's necessary to remove a parameter.
- 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 exists at the time of creation of the capture point, Wireshark aks you if the file can be overwritten. If the file exists at the time of activating the capture point, Wireshark overwrites 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 it meets a stop condition such as duration or packet capture limit is met. It can terminate if an internal error occurs, or resource is full (specifically if disk is full in file mode).
- Dropped packets won't be shown at the end of the capture. However, only the count of dropped and oversized packets are displayed.

Default Wireshark Configuration

The table below shows the default Wireshark configuration.

Feature	Default Setting
Duration	No limit
Packets	No limit

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

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. You can throttle the packet capture rate using further administrative controls. For example, You can filter the packets using an Access Control List. You can further define the controls by specifying a maximum packet capture rate or by specifying a sampling interval.

Prior to Cisco IOS XE Amsterdam 17.2.1 , EPC isn't supported on an interface in shutdown state. Starting from Cisco IOS XE Amsterdam 17.2.1 , EPC is supported on an interface in shutdown state. This is useful in capturing packets on an interface as it's being brought up.

In the Cisco IOS XE Dublin 17.12.4 release, Embedded Packet Capture is supported on Layer 2 interfaces and on control plane interfaces. This is applicable only to the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2)..

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.

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 parameters of the capture point.
3. Activate or deactivate a capture point.
4. Delete the capture point when you're no longer using it.

Defining a Capture Point

The example in this procedure defines a 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 Define an attachment point, direction of capture, and core filter to have a functional capture point.

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

	Command or Action	Purpose
		in the example). Capture Name should be less than or equal to eight 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 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. • (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. 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**

	Command or Action	Purpose
		If a capture point doesn't have a core system filter or attachment points defined, it can't be activated. Attempting to activate a capture point that doesn't 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 defined in Step 2 and confirms that you defined a capture point.

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

Note You can't change the parameters of a capture point using the methods presented in this topic.

If you enter an incorrect capture name, or an invalid/non existing attachment point, the switch shows errors, for example, "*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.
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:**	Displays the capture point parameters that you defined previously.

	Command or Action	Purpose
	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	
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**]
	Example:
	`Device# show monitor capture mycap parameter`
	`monitor capture mycap interface GigabitEthernet1/0/1 in`
		Displays the capture point parameters that remain defined after your parameter deletion operations. You can run this command at any 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.

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

You can activate a capture point even if an attachment point and a core system filter are defined and the associated filename exists. In such an instance, the existing file is overwritten.

You can activate a capture point with no associated filename only to display. When the filename isn't 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 aren't displayed live. All the packets captured by the core system filter are displayed. The default display mode is brief.

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*} **stop** **Example:** `Device# monitor capture name stop`	Deactivates a capture point.
Step 3	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
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.

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

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

Missing filter on activation

```
Device# monitor capture mycap int g1/0/1 both
Device# 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.
Device# monitor capture mycap match any
Device# monitor capture mycap start
Device#
*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

```
Device# monitor capture mycap start
PD start invoked while previous run is active Failed to start capture : Wireshark operation
 failure
Unable to activate Capture.
Device# 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)
Device# monitor capture test stop
Capture statistics collected at software (Buffer & Wireshark):
 Capture duration - 157 seconds
 Packets received - 0
 Packets dropped - 0
 Packets oversized - 0

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

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. Export - Saves the captured packets in the buffer and deletes the buffer.

	Command or Action	Purpose
Step 3	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
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

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 are 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 isn't required in this particular case since we've set a limit and the capture stops 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:

```
# 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 to stop automatically after the time has elapsed or the packet count are 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 are 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   0c..............
0040  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   ................
0060  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 to display its contents, 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 can't run show from buffer during an active capture. Stop capture 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 pcap file of the current capture session as well when the capture is active.

```
Device# monitor capture mycap start
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 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 you run the command, export is "started" but not complete when it returns the prompt to you. 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 to stop automatically after the time has elapsed or the packet count are met. When you see the following message in the output, you'll 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

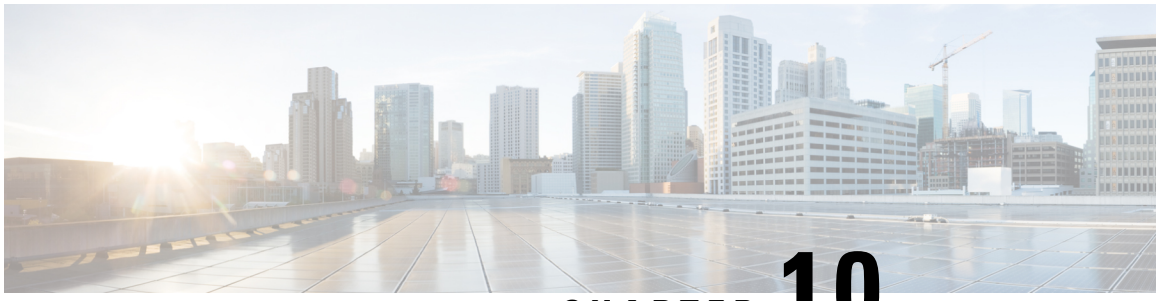
Feature History for Packet Capture

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 notedotherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	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.
Cisco IOS XE Amsterdam 17.3.1	Enhancements to Embedded Packet Capture (EPC) Packet filters: Packet length and Ether type	In Embedded Packet Capture (EPC), packets can be capture based on packet length or ether type.
Cisco IOS XE Cupertino 17.7.1	Packet Capture	This feature was implemented on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).
Cisco IOS XE Cupertino 17.8.1	Embedded Packet Capture	This feature was implemented on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

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 10

Configuring Enhanced Drop Detection and Enhanced Packet Drop Analyzer

- Restrictions for Configuring Enhanced Drop Detection and Enhanced Packet Drop Analyzer, on page 193
- Information About Enhanced Drop Detection and Enhanced Packet Drop Analyzer, on page 194
- Configuring Enhanced Drop Detection and Enhanced Packet Drop Analyzer to Troubleshoot Packet Drops, on page 195
- Configuration Examples for Enhanced Drop Detection and Enhanced Packet Drop Analyzer , on page 196
- Feature History for Enhanced Drop Detection and Enhanced Packet Drop Analyzer, on page 199

Restrictions for Configuring Enhanced Drop Detection and Enhanced Packet Drop Analyzer

The following restrictions are applicable to Enhanced Drop Detection:

- All the traps are not documented in the list of traps for Enhanced Drop Detection.
- Traffic Manager traps need to be cleared after use to ensure the accuracy of the trap counters.

The following restrictions are applicable to Enhanced Packet Drop Analyzer:

- Traffic Manager traps that are enabled in Enhanced Packet Drop Analyzer display aggregated counters.
- In Enhanced Packet Drop Analyzer, the rate of packet capture is limited to 100 packets per second.
- Network Processing Unit traps need to be enabled selectively. There is no mechanism to enable all Network Processing Unit traps at once.

Information About Enhanced Drop Detection and Enhanced Packet Drop Analyzer

The following sections provide information about configuring Enhanced Drop Detection and Enhanced Packet Drop Analyzer.

Enhanced Drop Detection

Enhanced Drop Detection (EDD) extends the functionality of packet capture by allowing you to determine where packets are being dropped in the processing path. Packets can be dropped for various reasons. Enhanced Drop Detection helps in detecting the dropped packets and displays them. Packets are displayed in a tabular format with different counters that show the name of the packet, the previous counter value, the current counter value and the calculated delta value. EDD allows you to display all the packets dropped at an ASIC level. It can also display packets specifically dropped by the Network Processing Unit (NPU) or the Traffic Manager (TM).

EDD can also display the traps at an ASIC level. A trap is a mechanism that is used by the data path to raise events called trap events. Trap events are based on various conditions that are evaluated while processing a packet. Based on trap events a packet can be duplicated, dropped or redirected to a new destination. EDD can display NPU traps and TM traps. The types of traps which are displayed are drop traps, punt traps and snoop traps.

The information about the traps can be used to troubleshoot where the packets are being dropped. The list of traps in EDD displays all the traps, not just the traps that represent dropped packets. By displaying the traps at regular intervals you can identify the traps that have the largest changes in their delta values.

Using EDD and information from the traps can be the first step in troubleshooting where packets are being dropped in the processing path. Further, by using Enhanced Packet Drop Analyzer the nature and causes of the packet drops can be evaluated.

Enhanced Packet Drop Analyzer

Enhanced Packet Drop Analyzer (EPDA) allows you to select and configure traps to punt dropped packets to a CPU based destination. Traps are mechanisms that are created by the system to analyze different conditions during packet processing. Based on specific conditions traps can punt packets to different destinations.

EPDA allows you to configure specific Network Processing Unit (NPU) traps or Traffic Manager (TM) traps to punt dropped packets. All the TM traps can be enabled at once. But NPU traps need to be enabled one by one. Once EPDA is stopped, you should clear all traps to ensure they continue to work efficiently.

EPDA allows you to configure a buffer to capture the dropped packets. EPDA can display the details of the dropped packets and the statistics for all the dropped packets. This information can help in troubleshooting the reasons for packets being dropped.

Troublehsooting Packet Drop Using Enhanced Drop Detection and Enhanced Packet Drop Analyzer

EDD counts packets for all traps, including drop traps. EPDA allows you to configure specific drop traps to know which packets are being dropped. Using EDD along with EPDA you can get a greater understanding of why particluar packets are being dropped.

EDD displays all the traps. The previous, current and delta sections in the command outputs can help identify the relevant traps. Using the command multiple times will help you identify the traps with the largest changes in their outputs.

Once the relevant traps are identified, you can use EPDA to configure the identified traps to punt the dropped packets to the specified CPU destination. After the dropped packets have been punted, EPDA can display the details of the dropped packets. These details will help you understand the reasons for the packet drops.

After troubleshooting you need to clear all the traps. This is important to ensure the proper functionality of the traps.

Configuring Enhanced Drop Detection and Enhanced Packet Drop Analyzer to Troubleshoot Packet Drops

To troubleshoot where the packets are being dropped and the reasons for packet drops, perform this procedure

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**show platform hardware fed active fwd-asic traps** [**npu traps** | **tm traps**] **asic** [*asic instance* | **all**] **Example:** `Device# show platform hardware fed active fwd-asic traps npu-traps asic all` or `Device# show platform hardware fed active fwd-asic traps tm-traps asic 0`	Displays all the traps. The **npu-traps** keyword has both the *asic instance* and **all** options. The **tm-traps** keyword has only the *asic instance* option. Using the command multiple times can identify the traps with largest changes in their counters.
Step 3	**debug platform software fed active drop-capture set-trap** [**npu-traps** | **tm-traps**] *trap-name* **Example:** `Device# debug platform software fed active drop-capture set-trap npu-traps l3 l3-null-adj`	Configures the specified trap to punt dropped packets.

	Command or Action	Purpose
Step 4	**debug platform software fed active drop-capture start** **Example:** `Device# debug platform  software fed active drop-capture start`	Starts the punting of dropped packets to the specified destination.
Step 5	**debug platform software fed active drop-capture stop** **Example:** `Device# debug platform  software fed active drop-capture stop`	Stops the punting of dropped packets.
Step 6	**show platform software fed active drop packet-capture** **Example:** `Device# show platform software fed active drop packet-capture brief`	Displays the details of the dropped packets.
Step 7	**debug platform software fed active drop-capture clear-trap** **Example:** `Device# debug platform software fed active drop-capture clear-trap npu-traps l3 l3-null-adj`	Clears the traps configured to punt dropped packets.

Configuration Examples for Enhanced Drop Detection and Enhanced Packet Drop Analyzer

Example: Configuring Enhanced Drop Detection

The following examples shows how to display all the packets dropped at an ASIC level:

```
Device# sh platform hardware fed active fwd-asic drops asic 0
Note: Slice and IFG showing -1 are global counters
==========================================================================================
|  #     |                      Counters Name                          |slice_number|
ifg_number  |  prev_value   |  current_value  |     delta      |
==========================================================================================
|  1    |Fwd drop counter (DSP==1): pkts                                |          -1|
      -1|              0|                0|               0|
|  2    |Fwd drop counter (DSP==1): bytes                               |          -1|
      -1|              0|                0|               0|
|  3    |LM drop counter for slice                                      |          -1|
      -1|              0|                0|               0|
|  4    |RX IFGB6 Port0 drop_counter TC0                                |           3|
       0|              0|                0|               0|
|  5    |RX IFGB6 Port0 drop_counter TC1                                |           3|
```

```
       0|              0|                0|               0|
|  6    |RX IFGB6 Port0 drop_counter TC2                                |           3|
       0|              0|                0|               0|
|  7    |RX IFGB6 Port0 drop_counter TC3                                |           3|
       0|              0|                0|               0|
|  8    |RX IFGB6 Port4 drop_counter TC0                                |           3|
       0|              0|                0|               0|
|  9    |RX IFGB6 Port4 drop_counter TC1                                |           3|
       0|              0|                0|               0|
|  10   |RX IFGB6 Port4 drop_counter TC2                                |           3|
       0|              0|                0|               0|
```

The following example shows how to display all the NPU traps at an ASIC level:

```
Device# show platform hardware fed active fwd-asic traps npu-traps asic all
Trap ID |     Asic     |                    NPU Trap Name                    |     Prev
 |   Current    |    Delta
-------------------------------------------------------------------------------------------------------------

     1 |            0 |                         la_event_e_ETHERNET_ACL_DROP |            0
 |            0 |            0
     2 |            0 |                  la_event_e_ETHERNET_ACL_FORCE_PUNT |            0
 |            0 |            0
     4 |            0 |       la_event_e_ETHERNET_NO_TERMINATION_ON_L3_PORT |            0
 |            0 |            0
     5 |            0 |                 la_event_e_ETHERNET_CISCO_PROTOCOLS |            2
 |            2 |            0
     6 |            0 |                        la_event_e_ETHERNET_DA_ERROR |            0
 |            0 |            0
     7 |            0 |                    la_event_e_ETHERNET_DHCPV4_CLIENT |            0
 |            0 |            0
     8 |            0 |                    la_event_e_ETHERNET_DHCPV4_SERVER |            0
 |            0 |            0
     9 |            0 |                    la_event_e_ETHERNET_DHCPV6_CLIENT |            0
 |            0 |            0
   <snip>
```

The following example shows how to display all the TM traps at an ASIC level:

```
Device# show platform hardware fed active fwd-asic traps tm-traps asic all
Warning:
1. Per VOQ Counters will be affected, Please disable TM Traps in EDD CLI using clear option
2. Please note TM traps enabled in EPDA will display aggregated counters
-------------------------------------------------------------------------------------------------------------

Trap ID | Asic |                                      TM Trap
      |     Prev     |   Current    |    Delta
-------------------------------------------------------------------------------------------------------------

 1 |  0 |                la_tm_traps_e_EXACT_METER_PACKET_GOT_DROPPED_DUE_TO_EXACT_METER |
           0 |            0 |            0
 2 |  0 |    la_tm_traps_e_STATISTICAL_METER_PACKET_GOT_DROPPED_DUE_TO_STATISTICAL_METER |
           0 |            0 |            0
 3 |  0 |                                 la_tm_traps_e_ETHERNET_METERS_PACKET_OUT_OF_RATE |
           0 |            0 |            0
 4 |  0 |                                      la_tm_traps_e_RXPDR_A_COUNTER_DSP_OVERFLOW |
           0 |            0 |            0
 5 |  0 |                                       la_tm_traps_e_RXPDR_A_COUNTER_VN_OVERFLOW |
           0 |            0 |            0
 6 |  0 |                                     la_tm_traps_e_RXPDR_A_COUNTER_MCID_OVERFLOW |
           0 |            0 |            0
 7 |  0 |                                          la_tm_traps_e_RXPDR_B_COUNTER_OVERFLOW |
           0 |            0 |            0   <snip>
```

Example: Configuring Enhanced Packet Drop Analyzer

The following example shows how to enable NPU traps. NPU traps need to be enable one at a time:

```
Device# #debug platform software fed active drop-capture set-trap npu-traps ?
  L3        npu layer 3 traps
  OAMP      npu OAMP traps
  app       npu app traps
  ethernet  npu ethernet traps
  internal  npu internal traps
  ipv4      npu ipv4 traps
  ipv6      npu ipv6 traps
  mpls      npu mpls traps

Device# debug platform software fed active drop-capture set-trap npu-traps l3 ?
l3-absr-tbl-miss           npu trap l3 absr table miss
  l3-bfd-mic-ip-dis          npu trap l3 bfd micro ip disabled
  l3-drop-adjacency          npu trap l3 drop adj
  l3-enc-tbl-miss            npu trap l3 enacap table miss
  l3-int-hop-limit           npu trap l3 int hop limit
  l3-invalid-spi             npu trap l3 invalid spi
  l3-ip-mc-drop              npu trap l3 ip mc drop
  l3-lpm-def-drop            npu trap l3 lpm default drop
<snip>
```

The following exampl shows how to enable TM traps. All the TM traps can be enabled at once:

```
Device# #debug platform software fed active drop-capture set-trap tm-traps ?
  all                           Enable all TM Traps
  eth-meter-oor                 Ethernet Meter packets Out of rate
  exact-meter                   Exact Meter Drop reason
  rxpdr-b-overflow              RXPDR B overflow
  rxpdr-dsp-overflow            RXPDR DSP overflow
<snip>
Device# #debug platform software fed active drop-capture set-trap tm-traps all
```

The following example shows how to start punting the dropped packets:

```
Device# debug platform software fed active drop-capture start
```

The following example shows how to stop punting the dropped packets:

```
Device# debug platform software fed active drop-capture stop
```

The following example shows how to display the details of the dropped packets in brief:

```
Device# show platform software fed active drop packet-capture brief
DropPackets packet capturing: disabled. Buffer wrapping: disabled
Total captured so far :  2313 packet(s)
Capture capacity      :  4096 packet(s)
Max. Meta header size :    88 byte(s)
Max. Packet data size :   128 byte(s)

------ DropPackets Packet Number: 1, Timestamp: 2024/03/25 15:04:46.823 ------
 interface : phy:  [if-id: 0x00000000], pal:  [if-id: 0x00000000]
 misc info : cause: 0 [Reserved ], sub-cause: 0, linktype: UNKNOWN [0]
 CE    hdr : dest mac: 4e41.5000.0111, src mac: 4e41.5000.0111, ethertype: 0x7106
 meta  hdr : Nxt. Hdr: 0x1, Fwd. Hdr: 0x2, SSP: 0x19
 meta  hdr : DSP: 0xffff, SLP: 0xe, DLP: 0x95
 ether hdr : dest mac: 341b.2d76.fd02, src mac: 6c29.d29d.36c3
 ether hdr : vlan: 3012, ethertype: 0x8100
 ipv4  hdr : dest ip: 172.16.10.11, src ip: 192.168.100.18
 ipv4  hdr : packet len: 100, ttl: 254, protocol: 1 (ICMP)
 icmp  hdr : icmp type: 8, code: 0
```

The following example shows how to display the details of the dropped packets in detail. The detailed view includes information about the source interface. This information is not present in the brief view:

```
Device# show platform software fed active drop packet-capture detailed
DropPackets packet capturing: disabled. Buffer wrapping: disabled
Total captured so far :     1 packet(s)
Capture capacity      :  4096 packet(s)
Max. Meta header size :    88 byte(s)
Max. Packet data size :   128 byte(s)

------ DropPackets Packet Number: 1, Timestamp: 2024/07/11 13:25:48.157 ------
 interface : phy:  [if-id: 0x00000000], pal:  [if-id: 0x00000000]
 misc info : cause: 0 [Reserved ], sub-cause: 0, linktype: UNKNOWN [0]
 CE    hdr : dest mac: 4e41.5000.0111, src mac: 4e41.5000.0111, ethertype: 0x7106
 meta  hdr : Nxt. Hdr: 0x1, Fwd. Hdr: 0, SSP: 0xffff
 meta  hdr : DSP: 0x1d, SLP: 0x8002a, DLP: 0x2a
 ether hdr : dest mac: 0900.2b00.0005, src mac: 6c29.d293.6f46
 ether hdr : length: 1402

 Metadata Hex-Dump (captured metadata length: 54 bytes) :
 4E41500001114E41  5000011171060800  0C3500FFFF001D80  02A0002A0064000A
 74B91E3B00751E3A  F821100200400000  000000000000

 Packet Data Hex-Dump (captured packet length: 128 bytes) :
 09002B0000056C29  D2936F46057AFEFE  0383140100110100  0002010098004013
 001E057725D30300  0000F00502000000  258101CC01040349  00008404C0A828B2
 08FF000000000000  0000000000000000  0000000000000000  0000000000000000
 0000000000000000  0000000000000000  0000000000000000  0000000000000000

 Punt Header
 Nxt. Hdr        = 0x1     Ethernet         Fwd. Hdr        = 0       Ethernet
 Punt Padding    = 0                        Next Hdr. Off.  = 0
 Source         = 0xc     Eg. Trap        Code           = 0x35    ETHERNET_SAME_INTERFACE

 Flow Type       = 0                        SSP             = 0xffff
 DSP             = 0x1d    HundredGigE1/0/5  SLP             = 0x8002a Port5lVlan100
 DLP             = 0x2a                     Padding         = 0
 Relay ID        = 0x64                     Time Stamp      = 0xa74b91e3b0075

 Receive Time    = 0x1e3af821
```

The following example shows how to clear all traps:

```
Device# debug platform software fed active drop-capture clear-trap npu-traps all
```

or

```
Device# debug platform software fed active drop-capture clear-trap tm-traps all
```

Note All TM traps need to be cleared after each use.

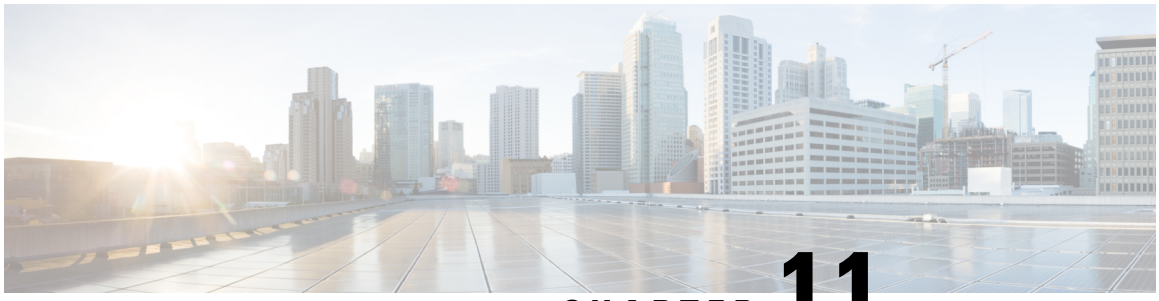
Feature History for Enhanced Drop Detection and Enhanced Packet Drop Analyzer

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 Dublin 17.12.4 This feature is not applicable to the Cisco IOS XE 17.13.x release and the Cisco IOS XE 17.14.x release.	Enhanced Drop Detection	Enhanced Drop Detection allows you to determine where packets are being dropped in the processing path. Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).
Cisco IOS XE Dublin 17.12.4 This feature is not applicable to the Cisco IOS XE 17.13.x release and the Cisco IOS XE 17.14.x release.	Enhanced Packet Drop Analyzer	Enhanced Packet Drop Analyzer allows you to configure traps to punt dropped packets to a CPU based destination for the purpose of troubleshooting. Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).
Cisco IOS XE 17.15.1	Enhanced Drop Detection	Enhanced Drop Detection allows you to determine where packets are being dropped in the processing path. Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).
Cisco IOS XE 17.15.1	Enhanced Packet Drop Analyzer	Enhanced Packet Drop Analyzer allows you to configure traps to punt dropped packets to a CPU based destination for the purpose of troubleshooting. Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

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 11

Configuring Flexible NetFlow

- Prerequisites for Flexible NetFlow, on page 201
- Restrictions for Flexible NetFlow, on page 202
- Information About Flexible NetFlow, on page 204
- How to Configure Flexible Netflow, on page 221
- Monitoring Flexible NetFlow, on page 235
- Configuration Examples for Flexible NetFlow, on page 235
- Feature History for Flexible NetFlow, on page 239

Prerequisites for Flexible NetFlow

- You are familiar with the Flexible NetFlow key fields as they are defined in the following commands:
  - **match flow**
  - **match interface**
  - **match** {**ipv4** | **ipv6**}
  - **match routing**
  - **match transport**
- You are familiar with the Flexible NetFlow nonkey fields as they are defined in the following commands:
  - **collect counter**
  - **collect flow**
  - **collect interface**
  - **collect**{**ipv4** | **ipv6**}
  - **collect routing**
  - **collect timestamp sys-uptime**
  - **collect transport**
  - **collect policy**

- The networking device must be running a Cisco release that supports Flexible NetFlow.

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 is not supported on the Layer 2 port-channel interface, but is supported on the Layer 2 port-channel member ports.
- Flexible NetFlow is supported on the Layer 3 port-channel interfaces and member ports but not on both at the same time for the same traffic type and direction.
- Multiple Flexible Netflow configurations are supported either on port-channel interface or port-channel member ports but not on both.
- 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.
- Layer 2, VLAN, Layer 3 and SVI interfaces are supported, but the device does not support tunnels.
- 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 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.
- 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.
- For an IPv6 flow monitor, Source Group Tag (SGT) and Destination Group Tag (DGT) fields cannot co-exist with MAC address fields.
- 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.
- Flow exporter exports the flow data only after the template data time out period ends. Configuration changes such as VPN ID modification or VRF deletion will take effect after the time out period ends.
- 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.
- When Flexible NetFlow and Network Address Translation (NAT) are configured on an interface,
    - Flexible NetFlow will display and export the actual flow details; but not the translated flow details. Application-level gateway (ALG) flow details are not part of the actual flow details that are exported.

- If the ALG traffic gets translated through the CPU, Flexible NetFlow will display and export the translated flow details for the ALG traffic.

- When configuring a flow record if you include the **match application name** field, the corresponding flow monitor will not support a flow sampler.

The following restrictions are applicable to the Cisco Catalyst 9600 Series Supervisor 2 Module:

- Flexible NetFlow is configurable only on the ingress interface. There is no support for NetFlow on the egress interface.
- Flexible NetFlow is configurable only with the sampler configuration.
- Flexible Netflow behavior is the same in random and deterministic sampler modes.
- The number of packets to be cached or sampled per flow is determined by the sampling rate.
- You can configure only one sampler rate on the switch at a time. To configure a new sampler rate on the switch, all the monitors on the switch will also have to be reconfigured.
- Flexible NetFlow can only perform a statistical sampling and not an exact sampling.
- Flexible Netflow is supported on Layer 2 and Layer 3 port-channel interface and not on port-channel member ports.
- NetFlow is only software implemented. As Netflow hardware is not supported, flow cache entries per ASIC on NetFlow TCAM cannot be used to troubleshoot.
- NetFlow monitors are not supported for MPLS and VXLAN protocols.
- The active timeout value configured must be greater than or equal to the inactive timeout value.
- **collect routing next-hop address ipv4**, **collect interface output**, and **collect routing** {**source** | **destination**} **as 4-octet** commands are not supported.
- **match flow cts** {**source** | **destination**} **group-tag** command is not supported.

Information About Flexible NetFlow

The following sections provide 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.

Starting with the Cisco IOS XE 16.12.1 release, Source Group Tag (SGT) and Destination Group Tag (DGT) fields over Flexible NetFlow are supported for IPv6 traffic.

Original NetFlow and Benefits of Flexible NetFlow

Flexible NetFlow allows the flow to be user defined. The benefits of Flexible NetFlow include:

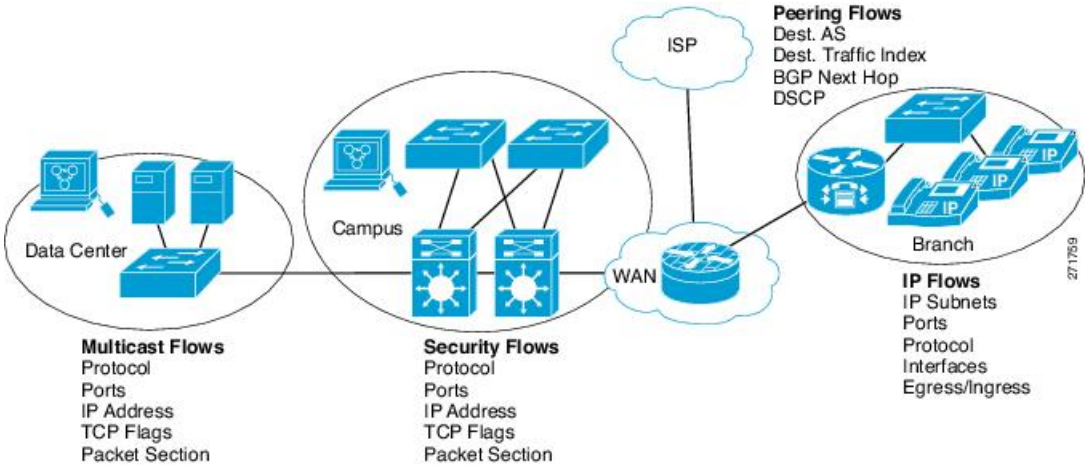
- High-capacity flow recognition, including scalability and aggregation of flow information.
- Enhanced flow infrastructure for security monitoring and dDoS detection and identification.
- New information from packets to adapt flow information to a particular service or operation in the network. The flow information available will be customizable by Flexible NetFlow users.
- Extensive use of Cisco's flexible and extensible NetFlow Version 9.
- 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.

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 10: 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 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 ability to analyze packet fields, enables more detailed traffic monitoring, facilitates the investigation of dDoS attacks, and enables implementation of other security applications such as URL monitoring.

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 11: 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).
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.

Command	Purpose
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.
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.

Flexible NetFlow Collect Parameters

The following table describes the Flexible NetFlow collect parameters.

Table 12: 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	• **packets**—packet fields • **rcp**—RCP fields **Note** The above commands are not supported on the Cisco Catalyst 9600X Series Switches. • **tcp**—TCP fields
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 flow {**sampler** |**username**}	Collects the following flow values: • **sampler**— Configures a flow sampler ID as a non-key field for the record. This field contains the ID of the flow sampler used to monitor the flow. **Note** This command is not supported on the Cisco Catalyst 9600X Series Switches. • **username**— Configures the username associated with the flow.

Command	Purpose
collect ipv4 {**source**|**destination**}	Collects the following IPv4 fields: • **source**— Configures the IPv4 source as a non-key field for a flow record. • **destination**— Configures the IPv4 destination as a non-key field for a flow record. **Note** This command is not supported on the Cisco Catalyst 9600X Series Switches.
collect ipv6 {**source**|**destination**}	Collects the following IPv6 information: • **source**— Configures the IPv6 source as a non-key field for a flow record. • **destination**— Configures the IPv6 destination as a non-key field for a flow record.
collect routing{ **next-hop address** |**destination**|**source**|**forwarding-status**|**multicast**	Collects the following routing values: • **next-hop address**— Information regarding the next hop from the flows. • **destination**— Destination routing attributes. • **source**— Source routing attributes. • **forwarding status**— Forwarding status of the packet. • **multicast**— Multicast routing attributes. **Note** This command is not supported on the Cisco Catalyst 9600X Series Switches.
collect policy firewall event	Configures the collect policy firewall event as a non-key field and enables collecting information on SGACL denied or permitted traffic based on the traffic pattern. **Note** This command is not supported on the Cisco Catalyst 9600X Series Switches.

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.

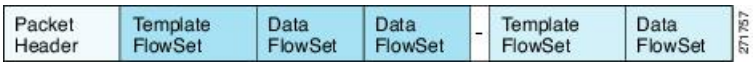
NetFlow Data Export Format Version 9

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.

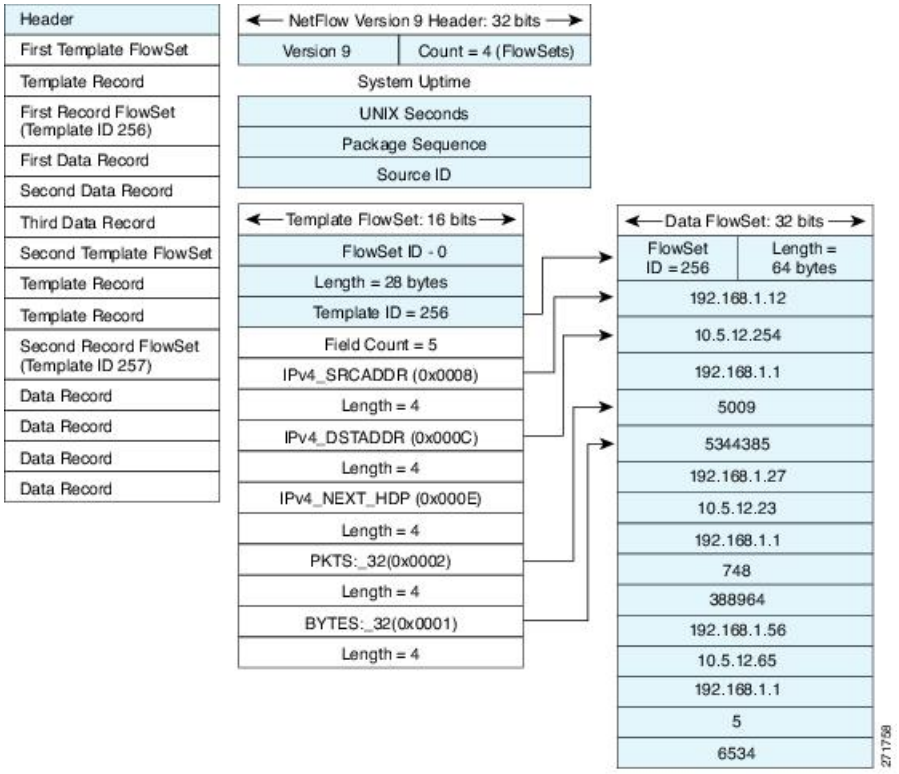
The Version 9 export format consists of a packet header followed by one or more template flow or data flow sets. A template flow set provides a description of the fields that will be present in future data flow sets. These data flow sets may occur later within the same export packet or in subsequent export packets. Template flow and data flow sets can be intermingled within a single export packet, as illustrated in the figure below.

Figure 11: 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 12: Detailed Example of the NetFlow Version 9 Export Format



For more information on the Version 9 export format, refer to the white paper titled Cisco IOS NetFlow Version 9 Flow-Record Format, available at this URL: http://www.cisco.com/en/US/tech/tk648/tk362/technologies_white_paper09186a00800a3db9.shtml.

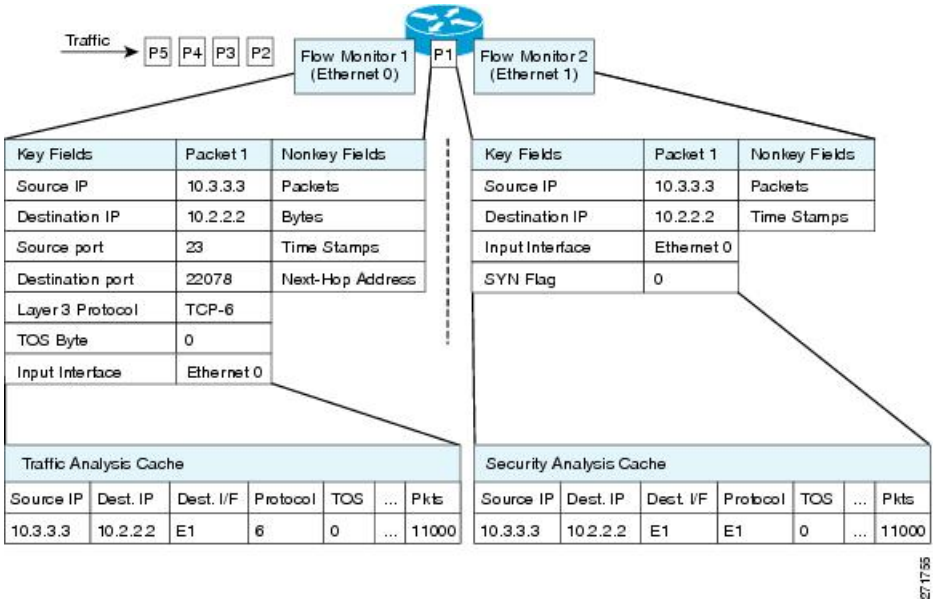
Flow Monitors

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.

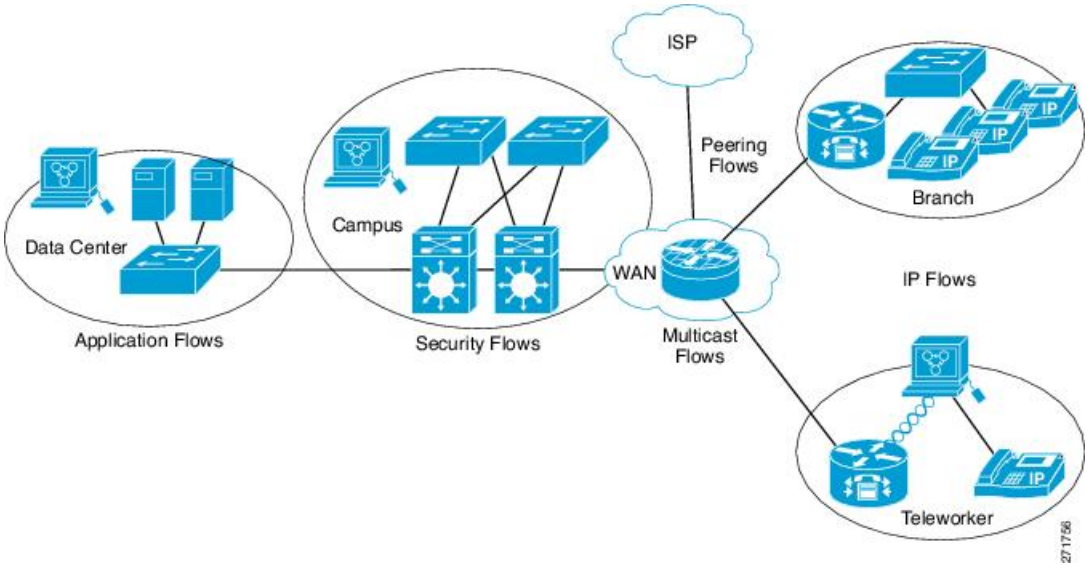
Flexible NetFlow can be used to perform different types of analysis on the same traffic. In the figure below, packet 1 is analyzed using a record designed for standard traffic analysis on the input interface and a record designed for security analysis on the output interface.

Figure 13: 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 14: Complex Example of Using Multiple Types of Flow Monitors with Custom Records



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.

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 TTL	—	—	Yes	Yes	Yes	Yes	Same as IPv4 TTL.
IPv4 protocol	—	—	Yes	Yes	Yes	Yes	Same as IPv4 protocol. Must use if any of src/dest port, ICMP code/type, IGMP type or TCP flags are used.
IPv4 source address	—	—	Yes	Yes	—	—	

Field	Layer 2 In	Layer 2 Out	IPv4 In	IP v4 Out	IPv6 In	IPv6 Out	Notes
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							
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	

Field	Layer 2 In	Layer 2 Out	IPv4 In	IP v4 Out	IPv6 In	IPv6 Out	Notes
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							
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 13: 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.

Flexible Netflow—Egress VRF Support Overview

The Flexible Netflow—Egress VRF Support feature enables collecting the VRF ID from outgoing packets on a device by applying an output flow monitor having a flow record that collects the VRF ID as a key field.

Autonomous System Number

The Autonomous System number space is a 32 bit field with 4,294,967,296 unique values, which are available for use to support the Internet's public inter-domain routing system.

An Autonomous System Number (AS number) is a special number assigned by IANA, used primarily with Border Gateway Protocol. It uniquely identifies a network under a single technical administration that has a unique routing policy, or is multi-homed to the public internet. This autonomous system number is required to run BGP and peer with your internet service provider, between internet service providers at peering points, and Internet Exchanges (IX). The AS number must be globally unique so that IP address blocks appear to come from a unique location that BGP can find and route to. BGP uses Prefixes and Autonomous System Paths (AS Paths) to determine the shortest path to a destination where a prefix is located.

NetFlow V9 and IPFIX export types support 32 bit AS number. NetFlow V5 export protocol does not support this 32 AS field, as it follows fixed 16 bit source and destination AS format.

You can export the below BGP parameters in Netflow:

- BGP source origin or peer AS number
- BGP destination origin or peer AS number

Configuration

Use the below command to configure AS number system:

[no] collect routing { **`destination | source`** } as **[[4-octet] peer] [4-octet]**

Ingress Egress Flexible NetFlow on MPLS Overview

- Ingress Flexible Netflow on MPLS (IP level): This feature allows the capture of Internet Protocol (IP) flow information for packets undergoing MPLS label imposition that are entering the MPLS network. These packets arrive on a router as IP packets and are transmitted as MPLS packets. This feature can be enabled by configuring an ingress flow monitor for IPv4 and IPv6 traffic at the CE facing side of the PE node.
- Egress Flexible NetFlow on MPLS (IP level): This feature allows the capture of Internet Protocol (IP) flow information for packets undergoing MPLS label imposition that are exiting the MPLS network. These packets arrive on a router as MPLS packets and are transmitted as IP packets. The feature can be enabled by configuring an egress flow monitor for IPv4 and IPv6 traffic at the CE facing side of PE node.

Configuring VPN ID in Flexible NetFlow

Multiple VPNs from the same private network can use same private source and destination IPs in the data traffic. This can make it difficult to identify the IP address to which the data belongs. A VPN-ID can be used to solve this issue. A VPN-ID is a global unique Virtual Private Network Identifier. It is used to identity a VPN across autonomous Systems (AS). If VPN-ID is exported in NetFlow exported packets – the collector in another AS will be able to associate and segregate the flows based on the VPN to which the data belongs. VPN-ID is a system level property similar to VRF-ID and it can be exported in a similar fashion.

Components of the VPN ID

Each VPN ID consists of the following elements:

- An Organizational Unique Identifier (OUI), a three-octet hex number. The IEEE Registration Authority assigns OUIs to any company that manufactures components under the ISO/IEC 8802 standard. The OUI is used to generate universal LAN MAC addresses and protocol identifiers for use in local and metropolitan area network applications. For example, an OUI for Cisco Systems is 00-03-6B (hex).
- A VPN index, a four-octet hex number, which identifies the VPN within the company.

You can configure the VPN ID by using the **vpn id** command in VRF definition configuration mode. Specify the VPN ID in the following format:

vpn id oui:vpn-index

Once the VPN ID has been configured you can use the **option vrf-attributes** command in flow-exporter configuration mode to configure the VPN ID.

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 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:** `Device(config-flow-record)# description Used for basic traffic analysis`	(Optional) Creates a description for the flow record.
Step 5	**match** {**ip** | **ipv6**} {**destination** | **source**} **address** **Example:** `Device(config-flow-record)# match ipv4 destination address`	**Note** This example configures the IPv4 destination address as a key field for the record.
Step 6	Repeat Step 5 as required to configure additional key fields for the record.	—
Step 7	**end** **Example:** `Device(config-flow-record)# end`	Exits Flexible NetFlow flow record configuration mode and returns to privileged EXEC mode.
Step 8	**show flow record** *record-name* **Example:** `Device# show flow record FLOW_RECORD-1`	(Optional) Displays the current status of the specified flow record.
Step 9	**show running-config flow record** *record-name* **Example:**	(Optional) Displays the configuration of the specified flow record.

	Command or Action	Purpose
	`Device# show running-config flow record FLOW_RECORD-1`	

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	**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	**flow exporter** *name* **Example:** `Device(config)# flow exporter ExportTest`	Creates a flow exporter and enters flow exporter configuration mode.
Step 4	**description** *string* **Example:** `Device(config-flow-exporter)# description ExportV9`	(Optional) Describes this flow record as a maximum 63-character string.
Step 5	**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.

	Command or Action	Purpose
Step 6	**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 7	**source** *interface type interface number* **Example:** `Device(config-flow-exporter)# source gigabitEthernet1/0/1`	(Optional) Specifies the interface to use to reach the NetFlow collector at the configured destination. **Note** Flow Exporter does not support unnumbered IP interface as source interface. The following interfaces can be configured as source: • **Auto Template**—Auto-Template interface • **Capwap**—CAPWAP tunnel interface • **GigabitEthernet**—Gigabit Ethernet IEEE 802 • **GroupVI**—Group virtual interface • **Internal Interface**—Internal interface • **Loopback**—Loopback interface • **Null**—Null interface • **Port-channel**—Ethernet Channel of interface • **TenGigabitEthernet**—10-Gigabit Ethernet • **Tunnel**—Tunnel interface • **Vlan**—Catalyst VLANs
Step 8	**transport udp** *number* **Example:** `Device(config-flow-exporter)# transport udp 200`	(Optional) Specifies the UDP port to use to reach the NetFlow collector.
Step 9	**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 10	**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 11	**end** **Example:** `Device(config-flow-record)# end`	Returns to privileged EXEC mode.
Step 12	**show flow exporter** [**name** *record-name*] **Example:** `Device# show flow exporter ExportTest`	(Optional) Displays information about NetFlow flow exporters.
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

Define a flow monitor based on the flow record and flow exporter.

Configuring a Flow Exporter for Flexible NetFlow V5

Perform this required task to configure the flow exporter for Flexible Netflow version 5 (V5).

Flexible Netflow NetFlow V5 Export Protocol feature enables sending export packets using the Version 5 export protocol. Netflow version 5 is a simpler export protocol than version 9 and IPFIX. It does not have templates due to a fixed record format, therefore, there is no configuration for template handling.

Following limitations for V5 export protocol are applicable when configured:

- Mandatory 5-tuple match fields.
  - `match ipv4 protocol`
  - `match ipv4 source address`
  - `match ipv4 destination address`
  - `match transport source-port`
  - `match transport destination-port`
- Any other fields are optional as long as they are part of the Version 5 subset.

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 exporter** *exporter-name* **Example:** `Device(config)# flow exporter EXPORTER-1`	Creates the flow exporter and enters Flexible NetFlow flow exporter configuration mode. • This command also allows you to modify an existing flow exporter.
Step 4	**description** *description* **Example:** `Device(config-flow-exporter)# description Exports to the datacenter`	(Optional) Configures a description to the exporter that will appear in the configuration and the display of the **show flow exporter** command.
Step 5	**export-protocol netflow-v5**	Enables support for netflow version v5 in the flow exporter.
Step 6	**destination** {*ip-address* | *hostname*} [**vrf** *vrf-name*] **Example:** `Device(config-flow-exporter)# destination 172.16.10.2`	Specifies the IP address or hostname of the destination system for the exporter. **Note** You can export to a destination using either an IPv4 or IPv6 address.
Step 7	**export-protocol** {**netflow-v5** | **netflow-v9** | **ipfix**} **Example:** `Device(config-flow-exporter)# export-protocol netflow-v9`	Specifies the version of the NetFlow export protocol used by the exporter. The export of extracted fields from NBAR is supported only over IPFIX. • Default: **netflow-v9**.
Step 8	**dscp** *dscp* **Example:** `Device(config-flow-exporter)# dscp 63`	(Optional) Configures differentiated services code point (DSCP) parameters for datagrams sent by the exporter. • The range for the *dscp* argument is from 0 to 63. Default: 0.

	Command or Action	Purpose
Step 9	**source** *interface-type interface-number* **Example:** `Device(config-flow-exporter)# source ethernet 0/0`	(Optional) Specifies the local interface from which the exporter will use the IP address as the source IP address for exported datagrams.
Step 10	**option** {**exporter-stats** | **interface-table** | **sampler-table** | **vrf-table**} [**timeout** *seconds*] **Example:** `Device(config-flow-exporter)# option exporter-stats timeout 120`	(Optional) Configures options data parameters for the exporter. • You can configure all three options concurrently. • The range for the *seconds* argument is 1 to 86,400. Default: 600.
Step 11	**output-features** **Example:** `Device(config-flow-exporter)# output-features`	(Optional) Enables sending export packets using quality of service (QoS) and encryption.
Step 12	**template data timeout** *seconds* **Example:** `Device(config-flow-exporter)# template data timeout 120`	(Optional) Configures resending of templates based on a timeout. • The range for the *seconds* argument is 1 to 86400 (86400 seconds = 24 hours).
Step 13	**transport udp** *udp-port* **Example:** `Device(config-flow-exporter)# transport udp 650`	Specifies the UDP port on which the destination system is listening for exported datagrams. • The range for the *udp-port* argument is from 1 to 65536.
Step 14	**ttl** *seconds* **Example:** `Device(config-flow-exporter)# ttl 15`	(Optional) Configures the time-to-live (TTL) value for datagrams sent by the exporter. • The range for the *seconds* argument is from 1 to 255.
Step 15	**end** **Example:** `Device(config-flow-exporter)# end`	Exits flow exporter configuration mode and returns to privileged EXEC mode.
Step 16	**show flow exporter** *exporter-name* **Example:** `Device# show flow exporter FLOW_EXPORTER-1`	(Optional) Displays the current status of the specified flow exporter.

	Command or Action	Purpose
Step 17	**show running-config flow exporter** *exporter-name* **Example:** `Device# show running-config flow exporter FLOW_EXPORTER-1`	(Optional) Displays the configuration of the specified 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.

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.

	Command or Action	Purpose
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.	—
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.

	Command or Action	Purpose
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.

	Command or Action	Purpose
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.
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 supported on the L3 port-channel interfaces and member ports but not on both at the same time.
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.
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`	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`startup-config`	

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	**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	**vlan** [**configuration**] *vlan-id*
	Example:
	`Device(config)# vlan configuration 30`
	`Device(config-vlan-config)#`
		Enters VLAN or VLAN configuration mode.

Step 4	**ip flow monitor** *monitor name* [**sampler** *sampler name*] {**input** } **Example:** `Device(config-vlan-config)# ip flow monitor MonitorTest input`	Associates a flow monitor and an optional sampler to the VLAN for input packets.
Step 5	**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	**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	**flow record** *name*
	Example:
	`Device(config)# flow record L2_record`
	`Device(config-flow-record)#`
		Enters flow record configuration mode.

Step 4	**match datalink** {**dot1q** |**ethertype** | **mac** | **vlan**} **Example:** `Device(config-flow-record)# match datalink ethertype`	Specifies the Layer 2 attribute as a key.
Step 5	**end** **Example:** `Device(config-flow-record)# end`	Returns to privileged EXEC mode.
Step 6	**show flow record** [*name*] **Example:** `Device# show flow record`	(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.

Monitoring Flexible NetFlow

The commands in the following table can be used to monitor Flexible NetFlow.

Table 14: 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)# 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
```

Example: Configuring Flexible NetFlow for Egress VRF Support

The following example configures the collection of the VRF ID from outgoing packets on a device by applying an output 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 intput
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 output
Device(config-if)# end
```

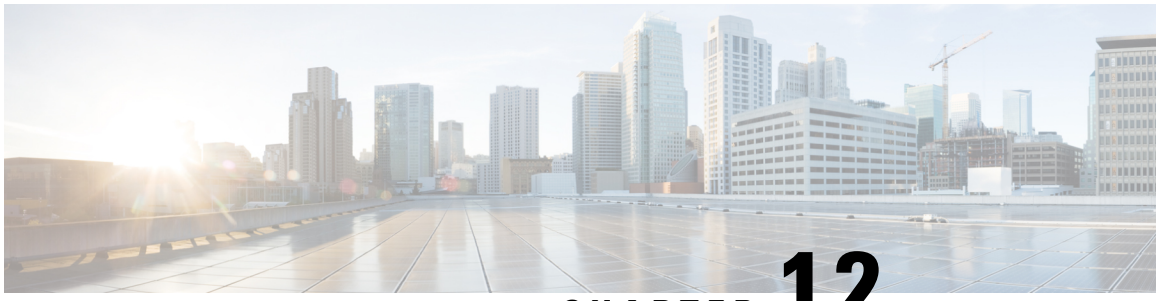
Feature History for Flexible NetFlow

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 notedotherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	Flexible NetFlow	Flexible NetFlow uses flows to provide statistics for accounting, network monitoring, and network planning.
Cisco IOS XE Gibraltar 16.12.1	Source Group Tag (SGT), Destination Group Tag (DGT) over Flexible NetFlow for IPv6 traffic	Support was introduced for Source Group Tag (SGT), Destination Group Tag (DGT) fields over Flexible NetFlow, for IPv6 traffic.
Cisco IOS XE Amsterdam 17.1.1	Ingress and Egress Flexible Netflow on MPLS	Allows capture of IP flow information for packets undergoing Multiprotocol Label Switching (MPLS) label imposition when entering an MPLS network. These packets arrive on a device as IP packets and are transmitted as MPLS packets.
Cisco IOS XE Amsterdam 17.2.1	VPN ID in NetFlow	Supports Virtual Private Network Identifier (VPN-ID) configuration in Flexible NetFlow. A VPN-ID is global and unique. It is used to identify a VPN across autonomous systems (ASes).
Cisco IOS XE Bengaluru 17.5.1	Flexible NetFlow for Egress VRF support	Support was introduced for configuring Flexible NetFlow for Egress VRF support.\
Cisco IOS XE Cupertino 17.7.1	Flexible NetFlow	This feature was implemented on the Cisco Catalyst 9600 Series Supervisor 2 Module (C9600X-SUP-2).

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 12

Configuring IP-aware Netflow for VRF Ingress

- Restrictions for IP-aware Netflow for VRF Ingress, on page 241
- Information About IP-aware Netflow for VRF Ingress, on page 241
- How to Configure IP-aware Netflow for VRF Ingress, on page 242
- Configuration Examples for IP-aware Netflow for VRF Ingress, on page 246
- Feature History for IP-aware Netflow for VRF Ingress, on page 248

Restrictions for IP-aware Netflow for VRF Ingress

- IP-aware VRF ingress Netflow is supported with IPv4, IPv6 and MVPNv4 as CE facing interface
- Supported only on layer 3 interface
- Supported only for ingress traffic on the VRF interface
- Supported only for MPLS L3 VPN VRF interface
- IP aware VRF ingress Netflow on MVPNv6 as CE facing interface is not supported
- Not supported on portchannel, SVI as CE facing interface
- Not supported for egress traffic on the VRF interface
- Not supported on MPLS L2VPN Attachment circuit interface

Information About IP-aware Netflow for VRF Ingress

This feature enables collecting the virtual routing and forwarding (VRF) ID from incoming packets on a router by applying an input flow monitor having a flow record that collects the VRF ID as a key or a non-key field.

Table 15: Scale Numbers

Platform	SDM Template	Max IPv4 Flows	Max IPv6 Flows
9300	Access	16K	8K
9400	Distribution	32K	16K

Platform	SDM Template	Max IPv4 Flows	Max IPv6 Flows
9500	Access	32K	16K
9600	Core	32K	32K

How to Configure IP-aware Netflow for VRF Ingress

This section provides the configuration steps for configuring IP-aware Netflow for VRF Ingress:

Creating a Flow Record

Perform the following task to create a flow record.

Step 1

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	**flow record** *flow_record_name* **Example:** `Device(config)# flow record flow-record-1`	Enters flow record configuration mode.
Step 4	**description** *description* **Example:** `Device(config-flow-record)# description flow-record-1`	(Optional) Creates a description for the flow record.
Step 5	**match ipv4 version** **Example:** `Device (config-flow-record)# match ipv4 version`	Specifies a match to the IP version from the IPv4 header.
Step 6	**match ipv4** {**source** | **destination**} *address*	Specifies a match to the IPv4 source and destination address.

	Command or Action	Purpose
Step 7	**match ipv4 protocol** **Example:** `Device (config-flow-record)# match ipv4 protocol`	Specifies a match to the IPv4 protocol.
Step 8	**match transport {source-port | destination-port}**	Configures source-port or destination port as a key field for the flow record.
Step 9	**match ipv4 tos** **Example:** `Device (config-flow-record)# match ipv4 tos`	Configures IPv4 ToS as a key field for the flow record.
Step 10	**match ipv4 ttl** **Example:** `Device (config-flow-record)# match ipv4 ttl`	Configures IPv4 TTL as a key field for the flow record.
Step 11	**match flow direction** **Example:** `Device (config-flow-record)# match flow direction`	Specifies a match to the flow identifying fields.
Step 12	**collect counter packets long** **Example:** `Device (config-flow-record)# collect flow direction`	Configures the number of packets seen in a flow as a non-key field and enables collecting the total number of packets from the flow.
Step 13	**collect counter bytes long** **Example:** `Device (config-flow-record)# collect counter bytes long`	Configures the number of bytes seen in a flow as a non-key field and enables collecting the total number of bytes from the flow.
Step 14	**end** **Example:** `Device(config-flow-record)# end`	Returns to privileged EXEC mode.
Step 15	**show flow record** **Example:** `Device # show flow record`	Displays information about all the flow records.

Creating a Flow Exporter

You can create a flow exporter to define the export parameters for a flow.

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	**flow exporter** *flow_exporter_name* **Example:** `Device(config)# flow exporter flow-exporter-1`	Enters flow exporter configuration mode.
Step 4	**description** *description* **Example:** `Device(config-flow-exporter)# description flow-exporter-1`	(Optional) Creates a description for the flow exporter.
Step 5	**destination** { *hostname* | *ipv4-address* | *ipv6-address* } **Example:** `Device (config-flow-exporter)# destination 10.10.1.1`	Specifies the hostname, IPv4 or IPv6 address of the system to which the exporter sends data.
Step 6	**source** *interface-type interface-name* **Example:** `Device (config-flow-exporter)# destination 10.10.1.1`	Specifies the local interface from which the exporter will use the IP address as the source IP address for exported datagrams.
Step 7	**end** **Example:** `Device(config-flow-record)# end`	Returns to privileged EXEC mode.
Step 8	**show flow exporter** **Example:** `Device # show flow exporter`	Displays information about all the flow exporters.

Creating a Flow Monitor

You can create a flow monitor and associate it with a flow record.

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	**flow monitor** *monitor-name* **Example:** `Device (config)# flow monitor flow-monitor-1`	Creates a flow monitor and enters flow monitor configuration mode.
Step 4	**description** *description* **Example:** `Device (config-flow-monitor)# description flow-monitor-1`	(Optional) Creates a description for the flow monitor.
Step 5	**record** *record-name* **Example:** `Device (config-flow-monitor)# record flow-record-1`	Specifies the name of a record that was created previously.
Step 6	**exporter** *exporter-name* **Example:** `Device (config-flow-monitor)# exporter flow-exporter-1`	Specifies the name of an exporter that was created previously.
Step 7	**cache type normal** {**timeout** | **active** | **inactive**} | **type normal**	(Optional) Specifies to configure flow cache parameters.
Step 8	**end** **Example:** `Device(config-flow-record)# end`	Returns to privileged EXEC mode.
Step 9	**show flow monitor** **Example:** `Device # show flow monitor`	Displays information about all the flow monitors.

Applying Flow Monitor 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** *interface-type interface-name*	Specifies an interface and enters interface configuration mode.
Step 4	**no switchport** **Example:** `Device(config-if)# description no switchport`	For physical ports only, enters Layer 3 mode.
Step 5	**vrf forwarding** *vrf-name*	Associates the VRF with the Layer 3 interface.
Step 6	{**ip** | **ipv6**} **flow-monitor** *monitor-name* **input**	Associates a flow monitor to the interface for input packets.
Step 7	**end** **Example:** `Device(config-flow-record)# end`	Returns to privileged EXEC mode.
Step 8	**show flow interface** **Example:** `Device# show flow interface`	Displays the status of NetFlow (enabled or disabled) on the specified interface.

Configuration Examples for IP-aware Netflow for VRF Ingress

The **show flow interface** command displays information about Netflow on the specified interface. :

```
Interface TenGigabitEthernet1/0/36
FNF:  monitor: v4vrfingress
 direction: Input
 traffic(ip): on
FNF: monitor: v6vrfingress
```

```
 direction: Input
 traffic(ipv6): on
```

The **show flow monitor** *flow-monitor-name* **cache** command displays the contents of the cache for the flow monitor.

```
  Cache type:                               Normal (Platform cache)
  Cache size:                                10000
  Current entries:                             100

  Flows added:                                 100
  Flows aged:                                    0

IPV4 SOURCE ADDRESS:       108.3.20.100
IPV4 DESTINATION ADDRESS:  108.2.20.100
TRNS SOURCE PORT:          0
TRNS DESTINATION PORT:     0
FLOW DIRECTION:            Input
IP VERSION:                4
IP TOS:                    0x20
IP PROTOCOL:               255
IP TTL:                    64
counter bytes long:        2956000
counter packets long:      2000
```

The **show flow exporter** command displays information about all the flow exporters. :

```
Flow Exporter v4vrfingress:
  Description:              User defined
  Export protocol:          NetFlow Version 9
  Transport Configuration:
    Destination type:       IP
    Destination IP address: 15.15.15.16
    Source IP address:      15.15.15.15
    Source Interface:       TenGigabitEthernet1/0/1
    Transport Protocol:     UDP
    Destination Port:       9995
    Source Port:            52319
    DSCP:                   0x0
    TTL:                    255
    Output Features:        Used
Flow Exporter v6vrfingress:
  Description:              User defined
  Export protocol:          NetFlow Version 9
  Transport Configuration:
    Destination type:       IP
    Destination IP address: 15.15.15.16
    Source IP address:      15.15.15.15
    Source Interface:       TenGigabitEthernet1/0/1
    Transport Protocol:     UDP
    Destination Port:       9995
    Source Port:            50881
    DSCP:                   0x0
```

```
    TTL:                    255
    Output Features:        Used
```

The **show platform software fed switch active fnf monitors-dump** displays Netflow monitors dump.

```
FNF Monitors
============
Monitor (0x7f4afc031748):
    profile_id(c461d4fe) ref_ct(1) wdavc_monitor(0)
wdavc_monitor_create_requested(False)
wdavc_remote_monitoring_remote_caching(0) flags(0x0000) is_wireless(No)
is_etta_over_fnf No ettaOrBaseProfile(00000000) etta_refcnt(0)
field(113) size(16) param(0) flags(1) offset(0)
field(114) size(16) param(0) flags(1) offset(16)
field(118) size(2) param(0) flags(1) offset(32)
field(119) size(2) param(0) flags(1) offset(34)
field(156) size(1) param(0) flags(1) offset(36)
field(181) size(8) param(0) flags(0) offset(37)
field(42) size(1) param(0) flags(1) offset(45)
field(46) size(1) param(0) flags(1) offset(46)
field(43) size(1) param(0) flags(1) offset(47)
field(47) size(1) param(0) flags(1) offset(48)
Monitor (0x7f4afc029338):
    profile_id(74c02ab0) ref_ct(1) wdavc_monitor(0)
wdavc_monitor_create_requested(False)
wdavc_remote_monitoring_remote_caching(0) flags(0x0000) is_wireless(No)
is_etta_over_fnf No ettaOrBaseProfile(00000000) etta_refcnt(0)
field(93) size(4) param(0) flags(1) offset(0)
field(94) size(4) param(0) flags(1) offset(4)
field(118) size(2) param(0) flags(1) offset(8)
field(119) size(2) param(0) flags(1) offset(10)
field(156) size(1) param(0) flags(1) offset(12)
field(177) size(8) param(0) flags(0) offset(13)
field(181) size(8) param(0) flags(0) offset(21)
field(42) size(1) param(0) flags(1) offset(29)
field(43) size(1) param(0) flags(1) offset(30)
field(46) size(1) param(0) flags(1) offset(31)
field(47) size(1) param(0) flags(1) offset(32)
```

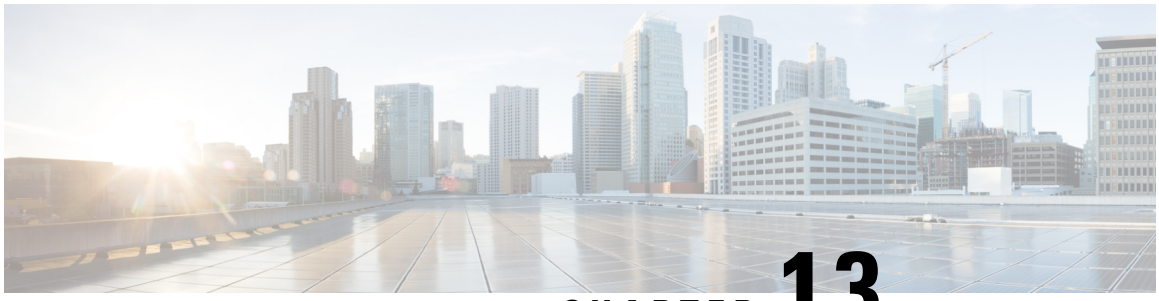
Feature History for IP-aware Netflow for VRF Ingress

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 notedotherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	IP-aware Netflow for VRF Ingress	This feature enables collecting the virtual routing and forwarding (VRF) ID from incoming packets on a router by applying an input flow monitor having a flow record that collects the VRF ID as a key or a non-key field.
Cisco IOS XE Cupertino 17.7.1	IP-aware Netflow for VRF Ingress	Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module.

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 13

Top-N Reports

- Information About Top-N Reports, on page 251
- How to use Top-N Reports, on page 252
- Examples : Top-N Reports, on page 254
- Feature History for Top-N Reports, on page 255

Information About Top-N Reports

Top-N Reports Overview

Top-N reports allow you to collect and analyze data for each physical port on a switch. When Top-N reports start, they obtain statistics from the appropriate hardware counters and then go into sleep mode for a user-specified interval. When the interval ends, the reports obtain the current statistics from the same hardware counters, compare the current statistics from the earlier statistics, and store the difference. Top-N reports feature is supported only the Cisco Catalyst 9500 High Performance Series Switches. The statistics for each port are sorted by one of the statistic types that are listed below:

- broadcast — Number of input/output broadcast packets
- bytes — Number of input/output bytes
- errors — Number of input errors
- multicast — Number of input/output multicast packets
- overflow — Number of buffer overflows
- packets — Number of input/output packets
- utilization — Utilization

Note When calculating the port utilization, Top-N reports bundles the Tx and Rx lines into the same counter and also looks at the full-duplex bandwidth when calculating the percentage of utilization. For example, a Gigabit Ethernet port would be 2000-Mbps full duplex.

Top-N Reports Operation

When you enter the collect top command, processing begins and the system prompt reappears immediately. When processing completes, the reports are not displayed immediately on the screen; the reports are saved for later viewing. The Top-N reports notify you when the reports are complete by sending a syslog message to the screen.

How to use Top-N Reports

The following sections provide information on how to use Top-N Reports.

Enabling Top-N Reports

To enable Top-N reports creation, perform this task:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**collect top** [*number_of_ports*] **counters interface** {*type* | **all** | **layer-2** | **layer-3**} [**sort-by** *statistic_type*] [**interval** *seconds*] **Example:** `Device# collect top 4 counters interface all sort-by utilization interval 76`	Enables Top-N reports creation. • *type* — type of interface — FastEthernet, GigabitEthernet, TenGigabitEthernet, FortyGigabitEthernet, TwentyFiveGigabitEthernet, HundredGigabitEthernet, Port-channel When enabling Top-N reports creation, note the following information: • You can specify the number of busiest ports for which to create reports (the default is 20). • You can specify the statistic type by which ports are determined to be the busiest (the default is utilization). The supported values for statistic_type are **broadcast**, **bytes**, **errors**, **multicast**, **overflow**, **packets**, and **utilization**. • You can specify the interval over which statistics are collected (range: 0 through 999; the default is 30 seconds). • Except for a utilization report (configured with the **sort-by utilization** keywords), you can specify an interval of zero to

	Command or Action	Purpose
		create a report that displays the current counter values instead of a report that displays the difference between the start-of-interval counter values and the end-of-interval counter values.

Displaying Top-N Reports

To display Top-N reports, perform this task:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**show top counters interface report** [*report_num*] **Example:** `Device# show top counters interface report 1`	Displays Top-N reports. **Note** To display information about all the reports, do not enter a *report_num* value. Top-N reports statistics are not displayed in these situations: • f a port is not present during the first poll. • If a port is not present during the second poll. • If a port's speed or duplex changes during the polling interval. • If a port's type changes from Layer 2 to Layer 3 during the polling interval. • If a port's type changes from Layer 3 to Layer 2 during the polling interval.

Clearing Top-N Reports

To clear Top-N reports, perform one of these tasks:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**clear top counters interface report** [*report_num*] **Example:** `Device# clear top counters interface report 4`	Clears all the Top-N reports that have a status of done. • *report_num* — Specifies the report number that must be cleared regardless of the status

Examples : Top-N Reports

Enabling Top-N Reports

This example shows how to enable Top-N reports creation for an interval of 76 seconds for the four ports with the highest utilization:

```
Device# collect top 4 counters interface all sort-by utilization interval 76
TopN collection started.
```

Displaying Top-N Reports

This example shows how to display information about all the Top-N reports:

Note Reports for which statistics are still being obtained are shown with a status of pending.

```
# show top counters interface report

Id Start Time Int N Sort-By Status Owner
-- --------------------------- --- --- --------- -------
----------------------
1 08:18:25 UTC Tue Nov 23 2004 76 20 util done console
2 08:19:54 UTC Tue Nov 23 2004 76 20 util done console
3 08:21:34 UTC Tue Nov 23 2004 76 20 util done console
4 08:26:50 UTC Tue Nov 23 2004 90 20 util done console
```

This example shows how to display a specific Top-N report:

```
# show top counters interface report 1

Started By : console
Start Time : 08:18:25 UTC Tue Nov 23 2004
End Time : 08:19:42 UTC Tue Nov 23 2004
```

```
Port Type : All
Sort By : util
Interval : 76 seconds
Port Band Util Bytes Packets Broadcast Multicast In- Buf-
width (Tx + Rx) (Tx + Rx) (Tx + Rx) (Tx + Rx) err ovflw
------- ----- ---- ----------- ----------- ---------- ---------- ----
-----

Gi2/5 100 50 726047564 11344488 11344487 1 0 0
Gi2/48 100 35 508018905 7937789 0 43 0 0
Gi2/46 100 25 362860697 5669693 0 43 0 0
Gi2/47 100 22 323852889 4762539 4762495 43 0 0
```

Clearing Top-N Reports

This example shows how to remove all reports that have a status of done:

```
# clear top counters interface report

04:00:06: %TOPN_COUNTERS-5-DELETED: TopN report 1 deleted by the console
04:00:06: %TOPN_COUNTERS-5-DELETED: TopN report 2 deleted by the console
04:00:06: %TOPN_COUNTERS-5-DELETED: TopN report 3 deleted by the console
04:00:06: %TOPN_COUNTERS-5-DELETED: TopN report 4 deleted by the console
```

This example shows how to remove a report number 4:

```
# clear top counters interface report 4

04:52:12: %TOPN_COUNTERS-5-KILLED: TopN report 4 killed by the console
```

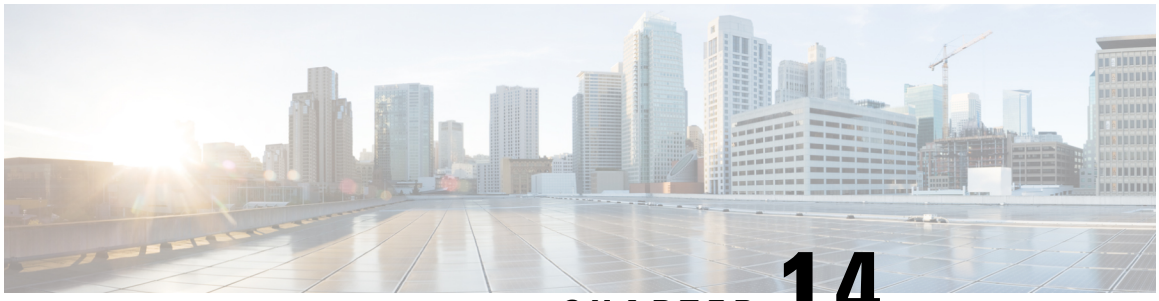
Feature History for Top-N Reports

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 notedotherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	Top-N Reports	Top-N reports allow you to collect and analyze data for each physical port on a switch.
Cisco IOS XE Cupertino 17.7.1	Top-N Reports	Support for this feature was introduced on the Cisco Catalyst 9600 Series Supervisor 2 Module.

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 14

Troubleshooting Network Management

- Overview, on page 257
- Support Articles, on page 257
- Feedback Request, on page 258
- Disclaimer and Caution, on page 258

Overview

This chapter provides links to documents authored by Cisco subject matter experts (SMEs). They aim to help you resolve technical issues without requiring a support ticket. If these documents are unable to resolve your issue, we recommend visiting the applicable Cisco Community. There is a wealth of information and advice available from fellow Cisco customers who may have experienced this issue already and provided a solution. If you are not able to find a resolution on the Community, it may be best that you raise a support ticket at Cisco Support. In cases where a support ticket has to be raised, these documents provide guidance about the data that should be collected and added to the support ticket. Specify the support document you referred, and TAC can create an improvement request with the document owner.

Support Articles

The documents in this section were created using specific software and hardware listed in the Components Used section of each article. However, this does not mean that they are limited to what is listed in Components Used, and generally remain relevant for later versions of software and hardware. Note that there could be some changes in the software or hardware that can cause commands to stop working, the syntax to change, or GUIs and CLIs to look different from one release to another.

The following are the support articles associated with this technology:

Document	Description
Verify SPAN and ERSPAN on Catalyst 9000 Series Switches	This document describes how to verify SPAN and ERSPAN on Catalyst 9000 Series Switches.
Configure and Verify Netflow, AVC, and ETA on Catalyst 9000 Series Switches	This document describes how to configure and validate NetFlow, Application Visibility and Control (AVC), and Encrypted Traffic Analytics (ETA).

Feedback Request

Your input helps. A key aspect to improving these support documents is customer feedback. Note that these documents are owned and maintained by multiple teams within Cisco. If you find an issue specific to the document (unclear, confusing, information missing, etc):

- Provide feedback using the **Feedback** button located at the right panel of the corresponding article. The document owner will be notified, and will either update the article, or flag it for removal.
- Include information regarding the section, area, or issue you had with the document and what could be improved. Provide as much detail as possible.

Disclaimer and Caution

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.