

FHS and SISF Configuration Guide

First Published: 2025-09-15

Last Modified: 2026-09-28

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)

© 2026 Cisco Systems, Inc. All rights reserved.



CONTENTS

CHAPTER 1 **IP Source Guard** 1

- Feature history for IP Source Guard 1
- IP Source Guard 1
- IP Source Guard for static hosts 2
- IP Source Guard configuration guidelines 3
- Enable IP Source Guard 3
- Configure IP Source Guard for static hosts on a Layer 2 access port 4
- Monitoring IP Source Guard 6

CHAPTER 2 **Dynamic ARP Inspection** 7

- Feature history for Dynamic ARP Inspection 7
- Understand Dynamic ARP Inspection 7
  - ARP packets in a network 8
  - How Dynamic ARP Inspection works 8
  - Interface trust states and network security 9
  - Rate limit of ARP packets 10
  - Relative priority of ARP ACLs and DHCP snooping entries 10
  - Log dropped packets 11
- Default Dynamic ARP Inspection configuration 11
- Restrictions for Dynamic ARP Inspection 11
- Configure Dynamic ARP Inspection 13
  - Configure ARP ACLs for non-DHCP environments 13
  - Configure Dynamic ARP Inspection in DHCP environments 15
  - Limit the rate of incoming ARP packets 17
  - Perform Dynamic ARP Inspection validation checks 19
- Configuration examples 20

Monitor Dynamic ARP Inspection 24

CHAPTER 3 DHCP Snooping 25

Feature history for DHCP snooping 25

Understand DHCP snooping 25

DHCP snooping and switch stacks 26

DHCP snooping binding database 26

DHCP gleaning 28

Trusted and untrusted sources 28

Default settings for DHCP snooping 29

Prerequisites for DHCP snooping 30

Configure DHCP snooping 30

Enable DHCP Snooping 30

Enable the DHCP snooping binding database agent 32

Configure an interface as a trusted or an untrusted source for DHCP gleaning 33

Monitor DHCP snooping information 34

CHAPTER 4 IPv6 FHS 37

Feature history for IPv6 First Hop Security 37

Understand IPv6 First Hop Security 38

IPv6 Router Advertisement Guard 38

IPv6 DHCP Guard 38

IPv6 source guard 39

IPv6 Prefix Guard 39

Prerequisites for IPv6 First Hop Security 39

Restrictions for IPv6 First Hop Security 40

Configure IPv6 First Hop Security 40

Configure the IPv6 binding table content 40

Configure an IPv6 router advertisement guard policy 41

Attach an IPv6 router advertisement guard policy to an interface 43

Attach an IPv6 router advertisement guard policy to a Layer 2 EtherChannel 44

Attach an IPv6 router advertisement guard policy to VLANs globally 45

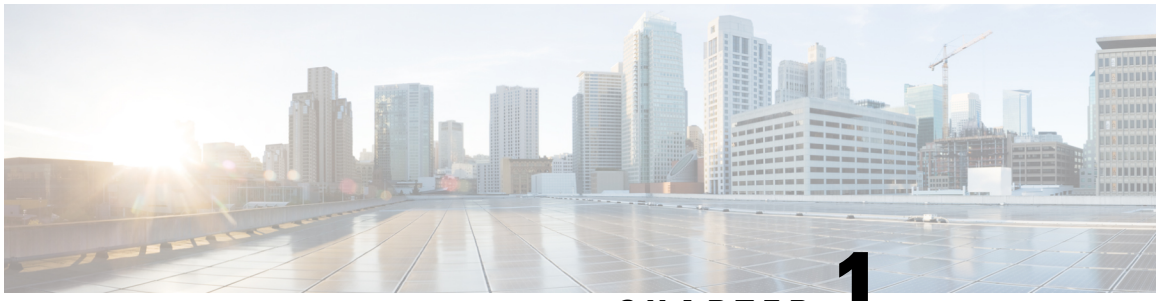
Configure an IPv6 DHCP guard policy 46

Attach an IPv6 DHCP guard policy to an interface 48

Attach an IPv6 DHCP guard policy to a Layer 2 EtherChannel 49
Attach an IPv6 DHCP guard policy to VLANs globally 50
Configure an IPv6 source guard policy 51
Attach an IPv6 source guard policy to an interface 52
Attach an IPv6 source guard policy to a Layer 2 EtherChannel interface 53
Configure an IPv6 Prefix Guard 54
Attach an IPv6 Prefix Guard Policy to an Interface 56
Attach an IPv6 Prefix Guard Policy to a Layer 2 EtherChannel Interface 57

CHAPTER 5 SISF 59
Feature history for SISF 59
Understand SISF 59
The binding table 60
States and lifetime of a binding table entry 61
Binding table sources 63
Actions on a packet and the binding table 65
Device-tracking 65
Device-tracking policy 66
Understand policy parameters 66
Glean versus guard versus inspect 66
Trusted-port and device-role switch 67
Address count limits 74
Tracking 76
Guidelines to create a policy 76
Guidelines to apply a policy 76
Configure SISF 76
Migrate from legacy IPDT and IPv6 Snooping to SISF-based device tracking 78
Apply the default device tracking policy to a target 79
Create a custom device tracking policy with custom settings 80
Attach a device tracking policy to an interface 83
Attach a device tracking policy to a VLAN 84
Enable SISF using an interface template 85
Configuration examples 87
Example: Programatically enable SISF by configuring DHCP Snooping 87

Example: Programatically enable SISF by configuring EVPN on VLAN 87
Example: Programatically enable SISF by configuring LISP 88
Example: Mitigate the IPv4 duplicate address problem 89
Example: Disable IPv6 device tracking on a target 91
Example: Enable IPv6 for SVI on VLAN 91
Example: Configure a multi-switch network to stop creating binding entries from a trunk port 91
Example: Avoid a short device-tracking binding reachable time 91
Example: Detect and prevent spoofing 92



CHAPTER 1

IP Source Guard

- Feature history for IP Source Guard, on page 1
- IP Source Guard, on page 1
- IP Source Guard for static hosts, on page 2
- IP Source Guard configuration guidelines, on page 3
- Enable IP Source Guard, on page 3
- Configure IP Source Guard for static hosts on a Layer 2 access port, on page 4
- Monitoring IP Source Guard, on page 6

Feature history for IP Source Guard

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 26.2.1ea	IP Source Guard: IP Source Guard feature support has been introduced.	Cisco C9550 Series Smart Switches
Cisco IOS XE 17.18.1	IP Source Guard: IP Source Guard is a security feature that restricts IP traffic on nonrouted Layer 2 interfaces by filtering traffic based on the DHCP snooping binding database and on manually configured IP source bindings.	Cisco C9350 Series Smart Switches

IP Source Guard

IP Source Guard is a security feature that restricts IP traffic on nonrouted Layer 2 interfaces by filtering traffic based on the DHCP snooping binding database and on manually configured IP source bindings.

You can enable IP source guard when DHCP snooping is enabled. This ensures that IP traffic with a source IP address in the binding table is allowed, while all other traffic is denied.

The switch utilizes a source IP lookup table enabling both IP and MAC filtering through a combination of source IP and source MAC lookups. This table, known as the IP source binding table, contains bindings that are either learned by DHCP snooping or manually configured as static IP source bindings. Each entry in this table consists of an IP address, its associated MAC address, and its associated VLAN number.

IP source guard is supported on private VLAN access port and etherchannel but not on private VLAN trunk port. You have the capability to configure IP Source Guard with source IP address filtering or with source IP and MAC address filtering.

IP Source Guard for static hosts

Note Do not use IP Source Guard for static hosts on uplink ports or trunk ports.

A static host is a network device in a non-DHCP environment that:

- requires IP addresses to be manually assigned,
- extends IP Source Guard capabilities without relying on DHCP, and
- relies on IP device tracking-table entries to manage port ACLs.

Any traffic received from a host without a valid DHCP binding entry is dropped. This security feature restricts IP traffic on nonrouted Layer 2 interfaces by filtering traffic based on the DHCP snooping binding database and manually configured IP source bindings.

IP Source Guard for static hosts also supports dynamic hosts. If a dynamic host receives a DHCP-assigned IP address in the IP DHCP snooping table, the same entry is learned by the IP device tracking table. In a stacked environment, when the active switch failover occurs, the IP Source Guard entries for static hosts attached to member ports are retained. When you enter the **show device-tracking database** EXEC command, the IP device tracking table displays the entries as ACTIVE.

Note Some IP hosts with multiple network interfaces can inject invalid packets into the network. Invalid packets contain the IP or MAC address of another network interface as the source address. The invalid packets can cause IP Source Guard for static hosts to interact with the host, to learn the invalid IP or MAC address bindings, and to reject the valid bindings. Contact the vendor of the operating system and network interface to stop the host from injecting invalid packets.

IP Source Guard for static hosts initially learns IP or MAC bindings dynamically through an ACL-based snooping mechanism. IP or MAC bindings are learned via ARP and IP packets and stored in the device tracking database. If dynamically learned or statically configured IP addresses on a port reach their maximum, the hardware drops any packet with a new IP address.

To resolve hosts that have moved or gone away for any reason, IP Source Guard for static hosts leverages IP device tracking to age out dynamically learned IP address bindings. This feature can be used with DHCP snooping. Multiple bindings are established on a port that is connected to both DHCP and static hosts. For

example, bindings are stored in both the device tracking database as well as in the DHCP snooping binding database.

IP Source Guard configuration guidelines

- Configure static IP bindings only on nonrouted ports. If you enter the **ip source binding** *mac-address* **vlan** *vlan-id ip-address* **interface** *interface-id* global configuration command on a routed interface, this error message appears:

  ```
  Static IP source binding can only be configured on switch port.
  ```

- Enable DHCP snooping on the access VLAN for the interface when enabling IP Source Guard with source IP address filtering.
- If you are enabling IP Source Guard on a trunk interface with multiple VLANs and DHCP snooping is enabled on all the VLANs, the source IP address filter is applied on all the VLANs.

Note If IP Source Guard is active and you change DHCP snooping on a VLAN on the trunk interface, the switch might not filter traffic properly.

- You can enable this feature when 802.1x port-based authentication is enabled.
- In a switch stack, if IP Source Guard is configured on a stack member interface and you remove the configuration of that switch by entering the **no switch** *stack-member-number* **provision** global configuration command, the interface static bindings are removed from the binding table, but they are not removed from the running configuration. The binding is restored when you re-provision the switch by entering the **switch** *stack-member-number* **provision** command.

  To remove the binding from the running configuration, you must disable IP Source Guard before entering the **no switch provision** command. The configuration is also removed if the switch reloads while the interface is removed from the binding table.

Enable IP Source Guard

Procedure

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

	Command or Action	Purpose
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Specifies the interface to be configured, and enters interface configuration mode.
Step 4	**ip verify source** [**mac-check**] **Example:** `Device(config-if)# ip verify source`	Enables IP Source Guard with source IP address filtering. (Optional) **mac-check**: Enables IP Source Guard with source IP address and MAC address filtering.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**ip source binding** *mac-address* **vlan** *vlan-id ip-address* **interface** *interface-id* **Example:** `Device(config)# ip source binding 0100.0230.0002 vlan 11 10.0.0.4 interface gigabitethernet1/0/1`	Adds a static IP source binding. Enter this command for each static binding.
Step 7	**end** **Example:** `Device(config)# end`	Exits global configuration mode and reutrns to privileged EXEC mode.

Configure IP Source Guard for static hosts on a Layer 2 access port

To ensure IP Source Guard works with static hosts, configure the **ip device tracking maximum** *limit-number* interface configuration command globally. In cases where you only configure this command on a port without enabling IP device tracking globally or by setting an IP device tracking maximum on that interface, IP Source Guard with static hosts will reject all IP traffic from that interface.

Procedure

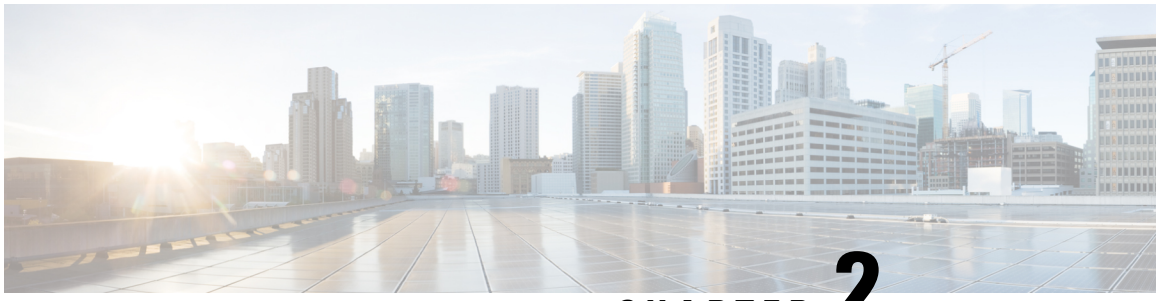
	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode.
Step 4	**device-tracking** **Example:** `Device(config-if)# device-tracking`	Enables SISF-based device tracking and attaches the default policy to the interface.
Step 5	**switchport mode access** **Example:** `Device(config-if)# switchport mode access`	Configures a port as access.
Step 6	**switchport access vlan** *vlan-id* **Example:** `Device(config-if)# switchport access vlan 10`	Configures the VLAN for this port.
Step 7	**ip verify source**[**tracking**] [**mac-check**] **Example:** `Device(config-if)# ip verify source tracking mac-check`	Enables IP Source Guard with source IP address filtering. (Optional) **tracking**: Enables IP Source Guard for static hosts. (Optional) **mac-check**: Enables MAC address filtering. Use the **ip verify source tracking mac-check** command to enable IP Source Guard for static hosts with MAC address filtering.
Step 8	**ip device tracking maximum** *number* **Example:** `Device(config-if)# ip device tracking maximum 8`	Establishes a maximum limit for the number of static IPs that the IP device tracking table allows on the port. The range is 1to 10. The maximum number is 10. **Note** You must configure the **ip device tracking maximum** *limit-number* interface configuration command.
Step 9	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Monitoring IP Source Guard

Table 1: Commands to monitor IP Source Guard

Command	Purpose
show ip verify source [**interface** *interface-id*]	Displays the IP Source Guard configuration on the switch or on a specific interface.
show ip device tracking { **all** | **interface** *interface-id* | **ip** *ip-address* | **mac** *mac-address*}	Displays information about the entries in the IP device tracking table.



CHAPTER 2

Dynamic ARP Inspection

- Feature history for Dynamic ARP Inspection, on page 7
- Understand Dynamic ARP Inspection, on page 7
- Default Dynamic ARP Inspection configuration, on page 11
- Restrictions for Dynamic ARP Inspection, on page 11
- Configure Dynamic ARP Inspection, on page 13
- Configuration examples, on page 20
- Monitor Dynamic ARP Inspection, on page 24

Feature history for Dynamic ARP Inspection

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 26.2.1	Dynamic ARP Inspection: Dynamic ARP Inspection feature support has been introduced.	Cisco C9550 Series Smart Switches
Cisco IOS XE 17.18.1	Dynamic ARP Inspection: Dynamic ARP Inspection is a security feature that validates ARP packets in a network. It intercepts, logs, and discards ARP packets with invalid IP-to-MAC address bindings.	Cisco C9350 Series Smart Switches

Understand Dynamic ARP Inspection

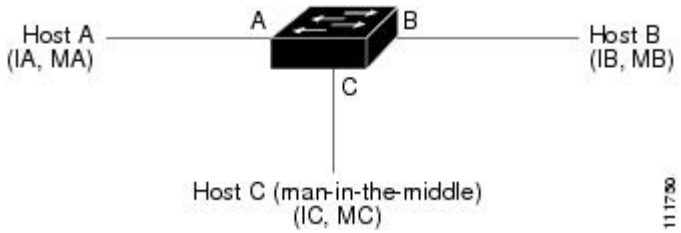
Dynamic ARP inspection is a security feature that validates ARP packets in a network. It intercepts, logs, and discards ARP packets with invalid IP-to-MAC address bindings. This capability protects the network from certain man-in-the-middle attacks.

ARP packets in a network

ARP provides IP communication within a Layer 2 broadcast domain by mapping an IP address to a MAC address. For example, Host B wants to send information to Host A but does not have the MAC address of Host A in its ARP cache. Host B generates a broadcast message for all hosts within the broadcast domain to obtain the MAC address associated with the IP address of Host A. All hosts within the broadcast domain receive the ARP request, and Host A responds with its MAC address. However, because ARP allows a gratuitous reply from a host even if an ARP request was not received, an ARP spoofing attack and the poisoning of ARP caches can occur. After the attack, all traffic from the device under attack flows through the attacker's computer and then to the router, switch, or host.

A malicious user can attack hosts, switches, and routers connected to your Layer 2 network by poisoning the ARP caches of systems connected to the subnet and by intercepting traffic intended for other hosts on the subnet. This figure shows an example of ARP cache poisoning.

Figure 1: ARP cache poisoning



Hosts A, B, and C are connected to the switch on interfaces A, B and C, all of which are on the same subnet. Their IP and MAC addresses are shown in parentheses; for example, Host A uses IP address IA and MAC address MA. When Host A needs to communicate to Host B at the IP layer, it broadcasts an ARP request for the MAC address associated with IP address IB. When the switch and Host B receive the ARP request, they populate their ARP caches with an ARP binding for a host with the IP address IA and a MAC address MA; for example, IP address IA is bound to MAC address MA. When Host B responds, the switch and Host A populate their ARP caches with a binding for a host with the IP address IB and the MAC address MB.

Host C can poison the ARP caches of the switch, Host A, and Host B by broadcasting forged ARP responses with bindings for a host with an IP address of IA (or IB) and a MAC address of MC. Hosts with poisoned ARP caches use the MAC address MC as the destination MAC address for traffic intended for IA or IB. This means that Host C intercepts that traffic. Because Host C knows the true MAC addresses associated with IA and IB, it can forward the intercepted traffic to those hosts by using the correct MAC address as the destination. Host C has inserted itself into the traffic stream from Host A to Host B, the classic *man-in-the middle*attack.

How Dynamic ARP Inspection works

Dynamic ARP inspection ensures that only valid ARP requests and responses are relayed. The switch performs these activities:

- Intercepts all ARP requests and responses on untrusted ports.
- Verifies that each of these intercepted packets has a valid IP-to-MAC address binding before updating the local ARP cache or before forwarding the packet to the appropriate destination.
- Drops invalid ARP packets.

Dynamic ARP Inspection determines the validity of an ARP packet based on valid IP-to-MAC address bindings stored in a trusted database, the DHCP snooping binding database. This database is built by DHCP snooping

if DHCP snooping is enabled on the VLANs and on the switch. If the ARP packet is received on a trusted interface, the switch forwards the packet without any checks. On untrusted interfaces, the switch forwards the packet only if it is valid.

In non-DHCP environments, Dynamic ARP Inspection can validate ARP packets against user-configured ARP access control lists (ACLs) for hosts with statically configured IP addresses. You define an ARP ACL by using the **arp access-list** *acl-name* global configuration command.

You can configure Dynamic ARP Inspection to drop ARP packets when the IP addresses in the packets are invalid or when the MAC addresses in the body of the ARP packets do not match the addresses specified in the Ethernet header. Use the **ip arp inspection validate** {[**src-mac**] [**dst-mac**] [**ip**]} global configuration command.

Interface trust states and network security

Dynamic ARP inspection associates a trust state with each interface on the switch. Packets arriving on trusted interfaces bypass all Dynamic ARP Inspection validation checks, and those arriving on untrusted interfaces undergo the Dynamic ARP Inspection validation process.

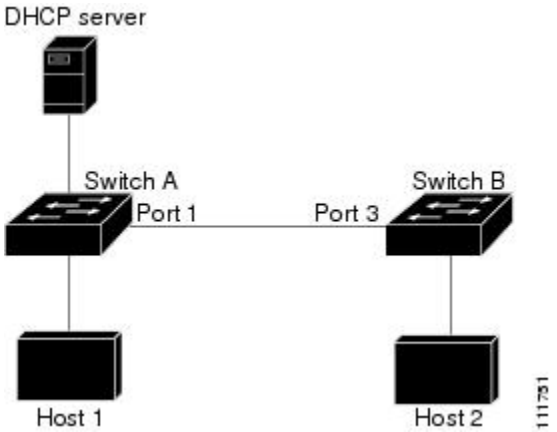
In a typical network configuration, you configure all switch ports connected to host ports as untrusted and configure all switch ports connected to switches as trusted. With this configuration, all ARP packets entering the network from a given switch bypass the security check. No other validation is needed elsewhere in the VLAN or network. You configure the trust setting by using the **ip arp inspection trust interface** configuration command.



Caution Carefully configure the trust state. Configuring interfaces as untrusted when they should be trusted can result in a loss of connectivity.

In the next figure, assume both Switch A and Switch B are running Dynamic ARP Inspection on the VLAN that includes Host 1 and Host 2. When Host 1 and Host 2 acquire their IP addresses from the DHCP server connected to Switch A, only Switch A binds the IP-to-MAC address of Host 1. Therefore, if the interface between Switch A and Switch B is untrusted, the ARP packets from Host 1 are dropped by Switch B. Connectivity between Host 1 and Host 2 is lost.

Figure 2: ARP packet validation on a VLAN enabled for Dynamic ARP Inspection



Trusting interfaces that should be untrusted creates network security gaps. If Switch A is not running Dynamic ARP Inspection, Host 1 can easily poison the ARP cache of Switch B (and Host 2, if the link between the switches is configured as trusted). This condition can occur even though Switch B is running Dynamic ARP Inspection.

Dynamic ARP inspection ensures that hosts connected to untrusted interfaces on a switch running Dynamic ARP Inspection do not poison the ARP caches of other hosts in the network. However, Dynamic ARP Inspection does not prevent hosts in other portions of the network from poisoning the caches of the hosts that are connected to a switch running Dynamic ARP Inspection.

In cases in which some switches in a VLAN run Dynamic ARP Inspection and other switches do not, configure the interfaces connecting such switches as untrusted. However, to validate the bindings of packets from non-Dynamic ARP Inspection switches, configure the switch running Dynamic ARP Inspection with ARP ACLs. When you cannot determine such bindings, at Layer 3, isolate switches running Dynamic ARP Inspection from switches not running Dynamic ARP Inspection switches.



Note It might not be possible to validate a given ARP packet on all switches in the VLAN, depending on the DHCP server and network setup.

Rate limit of ARP packets

The switch CPU performs Dynamic ARP Inspection validation checks, which limit the number of incoming ARP packets to prevent a denial-of-service attack. By default, the rate for untrusted interfaces is 15 packets per second (pps). Trusted interfaces are not rate-limited. Use the **ip arp inspection limit** interface configuration command to change this setting.

The switch places the port in the error-disabled state when the rate of incoming ARP packets exceeds the configured limit. The port remains in that state until you take action. You can use the **errdisable recovery** global configuration command to enable error disable recovery, allowing ports to automatically emerge from this state after a specified timeout period.



Note For an EtherChannel, the rate limit is applied separately to each switch in a stack. For example, if a limit of 20 pps is configured on the EtherChannel, each switch with ports in the EtherChannel can carry up to 20 pps. If any switch exceeds the limit, the entire EtherChannel is placed into the error-disabled state.

Relative priority of ARP ACLs and DHCP snooping entries

Dynamic ARP inspection uses the DHCP snooping binding database for the list of valid IP-to-MAC address bindings.

ARP ACLs take priority over DHCP snooping entries. The switch uses ACLs only if you configure them by using the **ip arp inspection filter vlan** global configuration command. The switch first compares ARP packets to user-configured ARP ACLs. If the ARP ACL denies the ARP packet, the switch will deny the packet even if a valid binding exists in the database populated by DHCP snooping.

Log dropped packets

When the switch drops a packet, it generates system messages on a rate-controlled basis after placing an entry in the log buffer. After the message is generated, the switch clears the entry from the log buffer. Each log entry contains flow information, such as receiving VLAN, port number, source IP address, destination IP address, source MAC address, and destination MAC address.

Use the **ip arp inspection log-buffer** global configuration command to configure the number of entries in the buffer and the number of entries needed in the specified interval to generate system messages. Use the **ip arp inspection vlan logging** global configuration command to specify the type of packets that are logged.

Default Dynamic ARP Inspection configuration

Feature	Default Settings
Dynamic ARP Inspection	Disabled on all VLANs.
Interface trust state	All interfaces are untrusted.
Rate limit of incoming ARP packets	The rate is set at 15 pps on untrusted interfaces, assuming network switching with hosts connecting to up to 15 new hosts per second. The rate is unlimited on all trusted interfaces. The burst interval is 1 second.
ARP ACLs for non-DHCP environments	No ARP ACLs are defined.
Validation checks	No checks are performed.
Log buffer	When Dynamic ARP Inspection is enabled, all denied or dropped ARP packets are logged. The number of entries in the log is 32. The number of system messages is limited to 5 per second. The logging-rate interval is 1 second.
Per-VLAN logging	All denied or dropped ARP packets are logged.

Restrictions for Dynamic ARP Inspection

This section lists the restrictions and guidelines for configuring Dynamic Address Resolution Protocol (ARP) Inspection on the switch.

- Dynamic ARP Inspection is an ingress security feature. It does not perform any egress checking.
- Dynamic ARP Inspection is not effective for hosts connected to switches that do not support Dynamic ARP Inspection or that do not have this feature enabled. Man-in-the-middle attacks are limited to a single Layer 2 broadcast domain.

For security, separate the domain with Dynamic ARP Inspection checks from ones without checking. This action secures the ARP caches of hosts in the domain enabled for Dynamic ARP Inspection.

- Dynamic ARP Inspection verifies IP-to-MAC address bindings by relying on entries in the DHCP snooping binding database for incoming ARP requests and ARP responses.

  Make sure to enable DHCP snooping to permit ARP packets that have dynamically assigned IP addresses. Use ARP ACLs to permit or deny packets when DHCP snooping is disabled or in non-DHCP environments.

- Dynamic ARP Inspection supports access ports, trunk ports, and EtherChannel ports only.



Note Do not enable Dynamic ARP Inspection on RSPAN VLANs. Configuring Dynamic ARP Inspection on RSPAN VLANs can obstruct packet delivery to the RSPAN destination port.

- A physical port can join an EtherChannel port channel only if the trust state of the physical port and the channel port match. Otherwise, the physical port remains suspended in the port channel.

  A port channel inherits its trust state from the first physical port that joins the channel. Consequently, the trust state of the first physical port need not match the trust state of the channel.

  Conversely, when you change the trust state on the port channel, the switch configures a new trust state on all the physical ports that comprise the channel.

- The rate limit is calculated separately on each switch in a switch stack. For a cross-stack EtherChannel, this means that the actual rate limit might be higher than the configured value.

  For example, if you set the rate limit to 30 pps on an EtherChannel that has one port on switch 1 and one port on switch 2, each port can receive packets at 29 pps without causing the EtherChannel to become error-disabled.

- The operating rate for the port channel is cumulative across all the physical ports within the channel. For example, if you configure the port channel with an ARP rate-limit of 400 pps, all the interfaces combined on the channel receive an aggregate 400 pps. The rate of incoming ARP packets on EtherChannel ports is equal to the sum of the incoming rate of packets from all the channel members. Configure the rate limit for EtherChannel ports only after examining the rate of incoming ARP packets on the channel-port members.

  The rate of incoming packets on a physical port is checked against the port-channel configuration rather than the physical-ports configuration. The rate-limit configuration on a port channel is independent of the configuration on its physical ports.

  If the EtherChannel receives more ARP packets than the configured rate, the switch places the channel (including all physical ports) in the error-disabled state.

- Make sure to limit the rate of ARP packets on incoming trunk ports. Configure higher rates for trunk ports to reflect their aggregation and handle packets across multiple VLANs with Dynamic ARP Inspection enabled. You also can use the **ip arp inspection limit none** interface configuration command to make the rate unlimited. A high rate-limit on one VLAN can cause a denial-of-service attack to other VLANs when the software places the port in the error-disabled state.

- When Dynamic ARP Inspection (DAI) is enabled on the switch, policers that were configured to police ARP traffic are no longer effective. The result is that all ARP traffic is sent to the CPU.

Configure Dynamic ARP Inspection

This section provides information about the various tasks to configure Dynamic ARP Inspection.

Configure ARP ACLs for non-DHCP environments

This procedure shows how to configure Dynamic ARP Inspection when Switch B shown in Figure 2 does not support Dynamic ARP Inspection or DHCP snooping.

If you configure port 1 on Switch A as trusted, a security hole is created because both Switch A and Host 1 could be attacked by either Switch B or Host 2. To prevent this possibility, you must configure port 1 on Switch A as untrusted. To permit ARP packets from Host 2, you must set up an ARP ACL and apply it to VLAN 1. If Host 2's IP address is dynamic, separate Switch A from Switch B at Layer 3 and use a router to route packets between them.

Follow these steps to configure an ARP ACL on Switch A. This procedure is required in non-DHCP environments.

Procedure

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 **arp access-list** *acl-name*

Example:

```
Device(config)# arp access-list arpacl22
```

Defines an ARP ACL, and enters ARP access-list configuration mode. By default, no ARP access lists are defined.

Note
At the end of the ARP access list, there is an implicit **deny ip any mac any** command.

Step 4 **permit ip host** *sender-ip* **mac host** *sender-mac*

Example:

```
Device(config-arp-nacl))# permit ip host 10.2.2.2 mac host 0018.bad8.3fbd
```

Permits ARP packets from the specified host (Host 2).

- For *sender-ip*, enter the IP address of Host 2.
- For *sender-mac*, enter the MAC address of Host 2.

Step 5 **exit**

Example:

```
Device(config-arp-nacl)# exit
```

Exits ARP access-list configuration mode and returns to global configuration mode.

Step 6 **ip arp inspection filter** *arp-acl-name* **vlan** *vlan-range* [**static**]

Example:

```
Device(config)# ip arp inspection filter arpacl22 vlan 1-2
```

Applies ARP ACL to the VLAN. By default, no defined ARP ACLs are applied to any VLAN.

- For *arp-acl-name*, specify the name of the ACL created in Step 2.
- For *vlan-range*, specify the VLAN that the switches and hosts are in. You can specify a single VLAN identified by VLAN ID number, a range of VLANs separated by a hyphen, or a series of VLANs separated by a comma. The range is 1 to 4094.
- (Optional) Specify **static** to treat implicit denies in the ARP ACL as explicit denies and to drop packets that do not match any previous clauses in the ACL. DHCP bindings are not used.

  If you do not specify this keyword, it means that there is no explicit deny in the ACL that denies the packet, and DHCP bindings determine whether a packet is permitted or denied if the packet does not match any clauses in the ACL.

ARP packets containing only IP-to-MAC address bindings are compared against the ACL. Packets are permitted only if the access list permits them.

Step 7 **interface** *interface-type interface-number*

Example:

```
Device(config)# interface gigabitethernt 0/1/1
```

Specifies Switch A interface that is connected to Switch B, and enters interface configuration mode.

Step 8 **no ip arp inspection trust**

Example:

```
Device(config-if)# no ip arp inspection trust
```

Configures Switch A interface that is connected to Switch B as untrusted.

By default, all interfaces are untrusted.

For untrusted interfaces, the switch intercepts all ARP requests and responses. It verifies that the intercepted packets have valid IP-to-MAC address bindings before updating the local cache and before forwarding the packet to the appropriate destination. The switch drops invalid packets and logs them in the log buffer according to the logging configuration specified with the **ip arp inspection vlan logging** global configuration command.

Step 9 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

Step 10 **show arp access-list** *acl-name*

Example:

```
Device# show arp access-list arpacl22
```

Displays information about the named ACLs.

Step 11 **show ip arp inspection vlan** *vlan-range*

Example:

```
Device# show ip arp inspection vlan 1-2
```

Displays the statistics for the selected range of VLANs.

Step 12 **show ip arp inspection interfaces**

Example:

```
Device# show ip arp inspection interfaces
```

Displays the trust state and the rate limit of ARP packets for the provided interface.

Configure Dynamic ARP Inspection in DHCP environments

Before you begin

This procedure shows how to configure Dynamic ARP Inspection when two switches support this feature. Host 1 is connected to Switch A, and Host 2 is connected to Switch B. Both switches are running Dynamic ARP Inspection on VLAN 1 where the hosts are located. A DHCP server is connected to Switch A. Both hosts acquire their IP addresses from the same DHCP server. Therefore, Switch A has the bindings for Host 1 and Host 2, and Switch B has the binding for Host 2.



Note Dynamic ARP inspection verifies IP-to-MAC address bindings by relying on entries in the DHCP snooping binding database for incoming ARP requests and ARP responses. Enable DHCP snooping, which permits ARP packets with dynamically assigned IP addresses.

Perform this procedure on both switches to configure Dynamic ARP Inspection.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **show cdp neighbors**

Example:

```
Device# show cdp neighbors
```

Verify the connection between the switches.

Step 3 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 4 **ip arp inspection vlan** *vlan-range*

Example:

```
Device(config)# ip arp inspection vlan 1
```

Enable Dynamic ARP Inspection on a per-VLAN basis; it is disabled on all VLANs by default.

For vlan-range, specify a single VLAN identified by VLAN ID number, a range of VLANs separated by a hyphen, or a series of VLANs separated by a comma. The range is 1 to 4094. Specify the same VLAN ID for both switches.

Step 5 **Interface** *type number*

Example:

```
Device(config)# interface gigabitethernet 1/0/1
```

Specifies the interface connected to the other switch, and enter interface configuration mode.

Step 6 **ip arp inspection trust**

Example:

```
Device(config-if)# ip arp inspection trust
```

Configures the connection between the switches as trusted. By default, all interfaces are untrusted.

The switch does not check ARP packets that it receives from the other switch on the trusted interface. It simply forwards the packets.

For untrusted interfaces, the switch intercepts all ARP requests and responses. It verifies that intercepted packets have valid IP-to-MAC address bindings before updating the local cache and forwarding the packet to the destination. The switch drops invalid packets and logs them in the log buffer according to the logging configuration specified with the **ip arp inspection vlan logging** global configuration command.

Step 7 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

Step 8 **show ip arp inspection interfaces**

Example:

```
Device# show ip arp inspection interfaces
```

Verifies the Dynamic ARP Inspection configuration on interfaces.

Step 9 **show ip arp inspection vlan** *vlan-range*

Example:

```
Device# show ip arp inspection vlan 1
```

Verifies the Dynamic ARP Inspection configuration on VLAN.

Step 10 **show ip dhcp snooping binding**

Example:

```
Device# show ip dhcp snooping binding
```

Verifies the DHCP bindings.

Step 11 **show ip arp inspection statistics vlan** *vlan-range*

Example:

```
Device# show ip arp inspection statistics vlan 1
```

Checks the Dynamic ARP Inspection statistics on VLAN.

Limit the rate of incoming ARP packets

The switch CPU runs Dynamic ARP Inspection checks. Incoming ARP packets are limited to prevent service disruption.

If incoming ARP packets exceed the limit, the switch disables the port, which remains in that state until error-disabled recovery is enabled so that ports automatically emerge from this state after a specified timeout period.

Note If you do not configure a rate limit, changing the interface's trust state resets its rate limit to the default value. After you configure the rate limit, the interface retains the rate limit even when its trust state is changed. If you enter the **no ip arp inspection limit** interface configuration command, the interface reverts to its default rate limit.

Follow these steps to limit the rate of incoming ARP packets.

Procedure

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 0/1/2
```

Specifies the interface to be rate-limited, and enters interface configuration mode.

Step 4 **ip arp inspection limit** {**rate** *pps* [**burst interval** *seconds*] | **none**}

Limits the rate of incoming ARP requests and responses on the interface. The default rate is 15 pps on untrusted interfaces and unlimited on trusted interfaces. The burst interval is 1 second.

The keywords have these meanings:

- For **rate** *pps*, specify an upper limit for the number of incoming packets processed per second. The range is 0 to 2048 pps.
- (Optional) For **burst interval** *seconds*, specify the consecutive interval in seconds, over which the interface is monitored for a high rate of ARP packets. The range is 1 to 15.
- For **rate none**, specify no upper limit for the rate of incoming ARP packets that can be processed.

Step 5 **exit**

Example:

```
Device(config-if)# exit
```

Exits interface configuration mode and returns to global configuration mode.

Step 6 Use the following commands:

- **errdisable detect cause arp-inspection**
- **errdisable recovery cause arp-inspection**
- **errdisable recovery interval** *interval*

Example:

```
Device(config)# errdisable recovery cause arp-inspection
```

(Optional) Enables error recovery from the Dynamic ARP Inspection error-disabled state, and configures the Dynamic ARP Inspection recover mechanism variables.

By default, recovery is disabled, and the recovery interval is 300 seconds.

For **interval** *interval*, specify the time in seconds to recover from the error-disabled state. The range is 30 to 86400.

Step 7 **exit**

Example:

```
Device(config)# exit
```

Exits global configuration mode and returns to privileged EXEC mode.

Perform Dynamic ARP Inspection validation checks

Dynamic ARP inspection intercepts and logs ARP packets. It discards packets with invalid IP-to-MAC address bindings. You can configure the switch to perform additional checks on the destination MAC address, the sender and target IP addresses, and the source MAC address.

Follow these steps to perform specific checks on incoming ARP packets. This procedure is optional.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **ip arp inspection validate {[src-mac] [dst-mac] [ip]}**

Example:

```
Device(config)# ip inspection validate ip
```

Performs a specific check on incoming ARP packets. By default, no checks are performed.

The keywords have these meanings:

- For **src-mac**, check the source MAC address in the Ethernet header against the sender MAC address in the ARP body. This check is performed on both ARP requests and responses. When enabled, packets with different MAC addresses are classified as invalid and are dropped.
- For **dst-mac**, check the destination MAC address in the Ethernet header against the target MAC address in ARP body. This check is performed for ARP responses. When enabled, packets with different MAC addresses are classified as invalid and are dropped.
- For **ip**, check the ARP body for invalid and unexpected IP addresses. Addresses include 0.0.0.0, 255.255.255.255, and all IP multicast addresses. Sender IP addresses are checked in all ARP requests and responses, and target IP addresses are checked only in ARP responses.

You must specify at least one of the keywords. Each command overrides the configuration of the previous command; that is, if a command enables src and dst mac validations, and a second command enables IP validation only, the src and dst mac validations are disabled as a result of the second command.

Step 4 **exit**

Example:

```
Device(config)# exit
```

Exits global configuration mode and returns to privileged EXEC mode.

Step 5 **show ip arp inspection vlan** *vlan-range*

Example:

```
Device# show ip arp insepction vlan 1-2
```

Displays the statistics for the selected range of VLANs.

Configuration examples

In this example, the given ARP ACL is **permit any any**. Any incoming packet is inspected by ARP inspection enabled on VLAN 100 and permitted by this ACL rule. Hence, the ACL permit counter is incremented as shown below (count=1) because of the permit ACL rule hit. If the rule was **deny ip any any**, then the ACL drop counter would have incremented as any incoming packet will hit the deny any rule.

```
Device# show ip arp inspection

Source Mac Validation      : Disabled
Destination Mac Validation : Disabled
IP Address Validation      : Disabled

 Vlan     Configuration    Operation   ACL Match          Static ACL
 ----     -------------    ---------   ---------          ----------
  103     Enabled          Active      dai-2              No

 Vlan     ACL Logging      DHCP Logging      Probe Logging
 ----     -----------      ------------      -------------
  103     Deny             Deny              Off

 Vlan      Forwarded        Dropped     DHCP Drops      ACL Drops
 ----      ---------        -------     ----------      ---------
  103              0              0              0              0

 Vlan   DHCP Permits    ACL Permits  Probe Permits   Source MAC Failures
 ----   ------------    -----------  -------------   -------------------
  103              0              0              0                     0

 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------

 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------
  103                   0                        0                      0

Device# show arp access-list

ARP access list dai-2
    permit ip any mac any

Device# show arp access-list

ARP access list dai-2
    permit ip any mac any

Inspecting pkt from Gi2/0/12
ACL match : NACL Permit
Packet permitted by acl match.intf Gi2/0/12, linktype 1, da 00-00-5E-90-10-01
 Enqueued packet in dai software queuesending packet to PI for processing with SMAC =
```

```
00-00-5E-90-10-00{mac} and SRC_ADDR = 10.5.6.7{ipv4}
 0 : sec-cnt : 2, bi : 0, tot : 2
(Gi2/0/12/103)Src: 00-00-5E-90-10-00, Dst: 00-00-5E-90-10-00, SM: 00-00-5E-90-10-00, SI:
10.5.6.7, TM: 00-00-5E-90-10-00, TI: 10.2.3.4
```

```
Device# show ip arp inspection

Source Mac Validation      : Disabled
Destination Mac Validation : Disabled
IP Address Validation      : Disabled

 Vlan     Configuration    Operation   ACL Match          Static ACL
 ----     -------------    ---------   ---------          ----------
  103     Enabled          Active      dai-2              No

 Vlan     ACL Logging      DHCP Logging      Probe Logging
 ----     -----------      ------------      -------------
  103     Deny             Deny              Off

 Vlan      Forwarded        Dropped     DHCP Drops      ACL Drops
 ----      ---------        -------     ----------      ---------
  103              1              0              0              0

 Vlan   DHCP Permits    ACL Permits  Probe Permits   Source MAC Failures
 ----   ------------    -----------  -------------   -------------------
  103              0              1              0                     0

 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------

 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------
  103                   0                        0                     0
```

This is an example of a case where DHCP packet is permitted. ARP inspection is enabled on the incoming VLAN 100.

The received DHCP packet is forwarded as the programmed DHCP binding table entry contains source MAC address (00-00-5E-90-10-22) that matches the incoming ARP packet's source MAC address (00-00-5E-90-10-22). Hence, ARP inspection forwards the incoming ARP packet and the forward count is reflected under *DHCP Permits*.

```
Device# show ip dhcp snooping bin

MacAddress          IpAddress        Lease(sec)  Type           VLAN  Interface
------------------  ---------------  ----------  -------------  ----  --------------------
00-00-5E-90-10-22   10.10.10.15      84239       dhcp-snooping  100   GigabitEthernet1/0/4
Total number of bindings: 1
```

```
Device# show ip arp inspection

Source Mac Validation      : Enabled
Destination Mac Validation : Enabled
IP Address Validation      : Enabled

 Vlan     Configuration    Operation   ACL Match          Static ACL
 ----     -------------    ---------   ---------          ----------
  100     Enabled          Active

 Vlan     ACL Logging      DHCP Logging      Probe Logging
 ----     -----------      ------------      -------------
  100     Deny             Deny              Off

 Vlan      Forwarded        Dropped     DHCP Drops      ACL Drops
 ----      ---------        -------     ----------      ---------
```

```
 100             0               0            0              0

 Vlan   DHCP Permits    ACL Permits  Probe Permits   Source MAC Failures
 ----   ------------    -----------  -------------   -------------------
  100             0             0              0                    0

 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------

 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------
  100                   0                        0                       0

Inspecting pkt from Gi1/0/4
 Enqueued packet in dai software queue
 DAI processing: SMAC = 00-00-5E-90-10-22{mac} and SRC_ADDR = 10.10.10.15{ipv4} DMAC =
00-00-5E-90-10-44{mac} and DST_ADDR = 10.10.10.1{ipv4}vlan: 100, if_input: Gi1/0/4
 0 : sec-cnt : 2, bi : 0, tot : 2
 (Gi1/0/4/100)Src: 00-00-5E-90-10-22, Dst: 00-00-5E-90-10-44, SM: 00-00-5E-90-10-22, SI:
10.10.10.15, TM: 00-00-5E-90-10-44, TI: 10.10.10.1

Device# show ip arp inspection

Source Mac Validation      : Enabled
Destination Mac Validation : Enabled
IP Address Validation      : Enabled

 Vlan     Configuration    Operation   ACL Match          Static ACL
 ----     -------------    ---------   ---------          ----------
  100     Enabled          Active

 Vlan     ACL Logging      DHCP Logging      Probe Logging
 ----     -----------      ------------      -------------
  100     Deny             Deny              Off

 Vlan      Forwarded        Dropped     DHCP Drops      ACL Drops
 ----      ---------        -------     ----------      ---------
  100              1              0              0              0

 Vlan   DHCP Permits    ACL Permits  Probe Permits   Source MAC Failures
 ----   ------------    -----------  -------------   -------------------
  100             1             0              0                    0

 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------

 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------
  100                   0                        0                       0
```

Below is an example of a case where an incoming ARP packet is dropped. ARP inspection is enabled on the incoming VLAN 100.

The received ARP packet is dropped as the programmed DHCP binding table entry contains source MAC address (00-00-5E-90-10-22) that does not match the incoming ARP packet's source MAC address (00-00-5E-90-10-33). Hence, ARP inspection drops the incoming ARP packet and the drop counter increment is reflected under *DHCP Drops*.

```
Device# show ip dhcp snooping bin

MacAddress          IpAddress        Lease(sec)  Type           VLAN  Interface
------------------  ---------------  ----------  -------------  ----  --------------------
00:00:5E:90:10:22   10.10.10.15      85920       dhcp-snooping  100   GigabitEthernet1/0/4
Total number of bindings: 1
```

```
Device# show ip arp inspection

Source Mac Validation      : Enabled
Destination Mac Validation : Enabled
IP Address Validation      : Enabled

 Vlan     Configuration    Operation   ACL Match          Static ACL
 ----     -------------    ---------   ---------          ----------
  100     Enabled          Active

 Vlan     ACL Logging      DHCP Logging      Probe Logging
 ----     -----------      ------------      -------------
  100     Deny             Deny              Off

 Vlan      Forwarded        Dropped     DHCP Drops      ACL Drops
 ----      ---------        -------     ----------      ---------
  100             0           0               0              0

 Vlan   DHCP Permits    ACL Permits  Probe Permits   Source MAC Failures
 ----   ------------    -----------  -------------   -------------------
  100             0              0              0                     0

 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------

 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------
  100                  0                        0                      0

Inspecting pkt from Gi1/0/16
Packet marked for log by DHCP bindings.
DHCP snooping binding failure - Dropping packet
%SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Gi1/0/16, vlan
100.([00-00-5E-90-10-33/10.10.10.2/00-00-5E-90-10-44/10.10.10.1/01:04:51 UTC Wed Jul 30
2025])

Device# show ip arp inspection

Source Mac Validation      : Enabled
Destination Mac Validation : Enabled
IP Address Validation      : Enabled

 Vlan     Configuration    Operation   ACL Match          Static ACL
 ----     -------------    ---------   ---------          ----------
  100     Enabled          Active

 Vlan     ACL Logging      DHCP Logging      Probe Logging
 ----     -----------      ------------      -------------
  100     Deny             Deny              Off

 Vlan      Forwarded        Dropped     DHCP Drops      ACL Drops
 ----      ---------        -------     ----------      ---------
  100             0           1               1              0

 Vlan   DHCP Permits    ACL Permits  Probe Permits   Source MAC Failures
 ----   ------------    -----------  -------------   -------------------
  100             0              0              0                     0

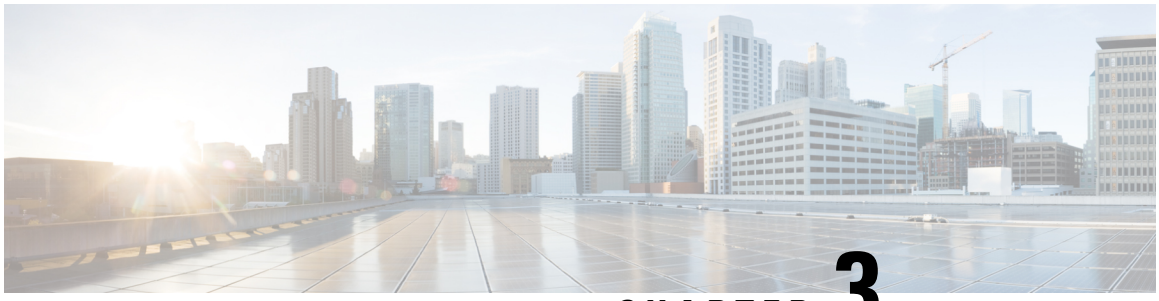
 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------

 Vlan   Dest MAC Failures   IP Validation Failures   Invalid Protocol Data
 ----   -----------------   ----------------------   ---------------------
  100                  0                        0                      0
```

Monitor Dynamic ARP Inspection

Use these commands to monitor Dynamic ARP Inspection configuration.

Command	Description
clear ip arp inspection statistics	Clears statistics for Dynamic ARP Inspection.
show ip arp inspection	Displays the configuration and the operating state of Dynamic ARP Inspection.
show ip arp inspection statistics [**vlan** *vlan-range*]	Displays statistics for forwarded packets, dropped packets, MAC validation failures, IP validation failures, ACL permitted and denied packets, and DHCP permitted and denied packets for the specified VLAN. If no VLANs are specified, or if a range is specified, it displays information only for VLANs with Dynamic ARP Inspection enabled (active).
clear ip arp inspection log	Clears the log buffer for Dynamic ARP Inspection.
show ip arp inspection log	Displays the configuration and contents of the Dynamic ARP Inspection log buffer.
show arp access-list [*acl-name*]	Displays detailed information about ARP ACLs.
show ip arp inspection interfaces [*interface-id*]	Displays the trust state and the rate limit of ARP packets for the specified interface or all interfaces.
show ip arp inspection vlan *vlan-range*	Displays the configuration and the operating state of Dynamic ARP Inspection for the specified VLAN. If no VLANs are specified or if a range is specified, displays information only for VLANs with Dynamic ARP Inspection enabled (active).



CHAPTER 3

DHCP Snooping

- Feature history for DHCP snooping, on page 25
- Understand DHCP snooping, on page 25
- Prerequisites for DHCP snooping, on page 30
- Configure DHCP snooping, on page 30
- Monitor DHCP snooping information, on page 34

Feature history for DHCP snooping

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 26.2.1ea	DHCP snooping: DHCP snooping feature support has been introduced.	Cisco C9550 Series Smart Switches
Cisco IOS XE 17.18.2	DHCP snooping: DHCP snooping acts like a firewall between untrusted hosts and trusted DHCP servers.	Cisco C9610 Series Smart Switches
Cisco IOS XE 17.18.1	DHCP snooping: DHCP snooping acts like a firewall between untrusted hosts and trusted DHCP servers.	Cisco C9350 Series Smart Switches

Understand DHCP snooping

DHCP snooping acts like a firewall between untrusted hosts and trusted DHCP servers. DHCP snooping performs the following activities:

- Validates DHCP messages received from untrusted sources and filters out invalid messages.

- Builds and maintains the DHCP snooping binding database, which contains information about untrusted hosts with leased IP addresses.
- Uses the DHCP snooping binding database to validate subsequent requests from untrusted hosts.

Other security features, such as dynamic Address Resolution Protocol (ARP) inspection (DAI), also uses information stored in the DHCP snooping binding database.

DHCP snooping is enabled on a per-VLAN basis. By default, the feature is inactive on all VLANs. You can enable the feature on a single VLAN or on a range of VLANs.



Note

- For DHCP snooping to function properly, all DHCP servers must be connected to the switch through trusted interfaces, as untrusted DHCP messages will be forwarded only to trusted interfaces.
- DAI is not supported on Cisco C9550 Series Smart Switches and Cisco C9610 Series Smart Switches.
- IP source guard is not supported on Cisco C9610 Series Smart Switches.

DHCP snooping and switch stacks

DHCP snooping is managed on the active switch. When you connect a new switch to the stack, it receives the DHCP snooping configuration. When a member leaves the stack, all DHCP snooping address bindings associated with the switch expires.

All snooping statistics are generated on the active switch. If a new active switch is elected, the statistics counters reset.

During a stack merge, if the switch is no more active, all DHCP snooping bindings are lost. In a stack partition, the active switch remains unchanged and the bindings belonging to partitioned switches expire. The new active switch of the partitioned stack begins processing the new incoming DHCP packets.

DHCP snooping binding database

A DHCP snooping binding database is a record system that

- stores information about untrusted interfaces when DHCP snooping is enabled and
- uses a checksum to verify entries for data integrity.

Each database entry (binding) has an IP address, an associated MAC address, the lease time, the interface to which the binding applies, and the VLAN to which the interface belongs. The database agent stores the bindings in a file at a configured location. At the end of each entry is a checksum that accounts for all the bytes from the start of the file through all the bytes associated with the entry. Each entry is 77 bytes, followed by a space, a checksum value, and the EOL symbol.

Use the DHCP snooping database agent to retain bindings when the switch reloads. If the agent is disabled, dynamic ARP inspection or IP source guard is enabled, and the DHCP snooping binding database has dynamic bindings, the switch loses its connectivity. If the agent is disabled and only DHCP snooping is enabled, the switch does not lose its connectivity, but DHCP snooping might not prevent DHCP spoofing attacks.

When the switch reloads, it reads the binding file to build the DHCP snooping binding database. The switch updates the file when the database changes.

When a switch learns of new bindings or when it loses bindings, the switch immediately updates the entries in the database. The switch also updates the entries in the binding file. The switch uses the DHCP snooping binding database to store information about untrusted interfaces when DHCP snooping is enabled. If the file is not updated in a specified time (set by the write-delay and abort-timeout values), the update stops.

Binding file format

This is the format of the file with bindings:

```
<initial-checksum>
TYPE DHCP-SNOOPING
VERSION 1
BEGIN
<entry-1> <checksum-1>
<entry-2> <checksum-1-2>
...
...
<entry-n> <checksum-1-2-..-n>
END
```

Each entry in the file is tagged with a checksum value, which the switch uses to verify the entries when it reads the file. The initial-checksum entry on the first line distinguishes entries associated with the latest file update from entries associated with a previous file update.

Binding file example

This is an example of a binding file:

```
3ebe1518
TYPE DHCP-SNOOPING
VERSION 1
BEGIN
10.1.1.1 512 001.0001.0005 3EBE2881 Gi1/1                                   e5e1e733
10.1.1.1 512 001.0001.0002 3EBE2881 Gi1/1                                   4b3486ec
10.1.1.1 1536 001.0001.0004 3EBE2881 Gi1/1                                  f0e02872
10.1.1.1 1024 001.0001.0003 3EBE2881 Gi1/1                                  ac41adf9
10.1.1.1 1 001.0001.0001 3EBE2881 Gi1/1                                     34b3273e
END
```

When the switch starts, and the calculated checksum value equals the stored checksum value, the switch reads entries from the binding file and adds the bindings to its DHCP snooping binding database. The switch ignores an entry when one of these situations occurs:

- The switch reads the entry and the calculated checksum value does not equal the stored checksum value. The entry and the ones following it are ignored.
- An entry has an expired lease time, although the switch may not remove the entry when the lease time expires.
- The interface in the entry no longer exists on the system.
- The interface is a routed interface or a DHCP snooping-trusted interface.

DHCP gleaning

DHCP gleaning is a read-only DHCP snooping functionality that:

- extracts location information from DHCP messages,
- differentiates an untrusted device port connected to an end user from a trusted port connected to a DHCP server, and
- registers and gleans only DHCP version 4 packets.

Gleaning helps extract location information from DHCP messages when messages are forwarded by a DHCP relay agent. The process does not block or modify DHCP packets, making it passive. DHCP gleaning is supported only on Layer 2 ports. By default, gleaning is disabled, though enabling a device sensor automatically activates it. It functions on all active interfaces where DHCP snooping is deactivated. For private VLAN interaction, ensure gleaning is active on the secondary VLAN, even if snooping is disabled on the primary VLAN.

Enabling DHCP snooping also internally creates DT-PROGRAMMATIC device tracking policy. This policy gleans ND, ARP, DHCPv4, and DHCPv6.



Note On Cisco C9350 Series Smart Switches, private VLAN destination lookup forwarding is based on the primary VLAN.

Trusted and untrusted sources

A trusted source is an entity or component that is considered reliable for network communications, provides authenticated data transmissions, and is verified to be part of the network infrastructure.

An untrusted DHCP message is a message that is received through an untrusted interface. By default, the switch considers all interfaces untrusted. So, the switch must be configured to trust some interfaces to use DHCP Snooping. When you use DHCP snooping in a service-provider environment, an untrusted message is sent from a device that is not in the service-provider network, such as a customer's switch. Messages from unknown devices are untrusted because they can be sources of traffic attacks.

The DHCP snooping binding database has the MAC address, the IP address, the lease time, the binding type, the VLAN number, and the interface information that corresponds to the local untrusted interfaces of a switch. It does not have information regarding hosts interconnected with a trusted interface.

In a service-provider network, an example of an interface you might configure as trusted is one connected to a port on a device in the same network. An example of an untrusted interface is one that is connected to an untrusted interface in the network or to an interface on a device that is not in the network.

When a switch receives a packet on an untrusted interface and the interface belongs to a VLAN in which DHCP snooping is enabled, the switch compares the source MAC address and the DHCP client hardware address. If the addresses match (the default), the switch forwards the packet. If the addresses do not match, the switch drops the packet.

The switch drops a DHCP packet when one of these situations occurs:

- A packet from a DHCP server, such as a DHCPOFFER, DHCPACK, DHCPNAK, or DHCPLEASEQUERY packet, is received from outside the network or firewall.

- A packet that is received on an untrusted interface, has a source MAC address and a DHCP client hardware address that do not match.
- The switch receives a DHCPRELEASE or DHCPDECLINE broadcast message that has a MAC address in the DHCP snooping binding database, but the interface information in the binding database does not match the interface on which the message was received.
- A DHCP relay agent forwards a DHCP packet that includes a relay-agent IP address that is not 0.0.0.0, or the relay agent forwards a packet that includes option-82 information to an untrusted port.
- DHCP snooping exceeds the queue size limit of 1000.

When an aggregation switch with DHCP snooping connects to an edge switch inserting DHCP option-82, packets with option-82 are dropped if received on an untrusted interface. If DHCP Snooping is enabled and packets are received on a trusted port, the aggregation switch does not learn the DHCP snooping bindings for connected devices and cannot build a complete DHCP snooping binding database.

When an aggregation switch can be connected to an edge switch through an untrusted interface and you enter the **ip dhcp snooping information option allow-untrusted** global configuration command, the aggregation switch accepts packets with option-82 information from the edge switch. The aggregation switch learns the bindings for hosts connected through an untrusted switch interface. The DHCP security features, such as Dynamic ARP Inspection or IP Source Guard, can still be enabled on the aggregation switch while the switch receives packets with option-82 information on untrusted input interfaces to which hosts are connected. The port on the edge switch that connects to the aggregation switch must be configured as a trusted interface.

When DHCP snooping is enabled on a VLAN where relay is configured, the SVI generates Layer 3 unicast packets destined for the DHCP server. In this scenario, it is not necessary to configure DHCP snooping trust on the server-facing port because the packets are unicast at Layer 3.

Default settings for DHCP snooping

Table 2: Default DHCP snooping configuration

Feature	Default Setting
DHCP snooping	Disabled
DHCP snooping information option	Enabled
DHCP snooping option to accept packets on untrusted input interfaces[1]	Disabled
DHCP snooping limit rate	None configured
DHCP snooping trust	Untrusted
DHCP snooping VLAN	Disabled
DHCP snooping MAC address verification	Enabled
DHCP snooping binding database agent	Enabled in Cisco IOS software, requires configuration. This feature is operational only when a destination is configured.

[1] Use this feature when the switch is an aggregation switch that receives packets with option-82 information from an edge switch.

Prerequisites for DHCP snooping

These prerequisites apply to DHCP snooping:

- You must be familiar with DHCP before you configure DHCP snooping.
- Before globally enabling DHCP snooping on the switch, make sure that the devices acting as the DHCP server and the DHCP relay agent are configured and enabled.
- For DHCP snooping to function properly, all DHCP servers must be connected to the switch through trusted interfaces, as untrusted DHCP messages will be forwarded only to trusted interfaces. In a service-provider network, a trusted interface is connected to a port on a device in the same network.
- You must configure the switch to use the Cisco IOS DHCP server binding database to use it for DHCP snooping.
- To use the DHCP snooping option of accepting packets on untrusted inputs, the switch must be an aggregation switch that receives packets with option-82 information from an edge switch.
- The following prerequisites apply to DHCP snooping binding database configuration:
  - You must configure a destination on the DHCP snooping binding database to use the switch for DHCP snooping.
  - Because both NVRAM and the flash memory have limited storage capacity, we recommend that the binding file be stored on a TFTP server.
  - You must create an empty file at the configured network-based URLs (such as TFTP and FTP) before the switch can write bindings to the binding file at that URL. See the documentation for your TFTP server to determine whether you must first create an empty file on the server; some TFTP servers cannot be configured this way.
  - To ensure the accuracy of the lease time in the database, we recommend that you enable and configure Network Time Protocol (NTP).
  - If NTP is configured, the switch writes binding changes to the binding file only when the switch system clock is synchronized with NTP.

Configure DHCP snooping

This section provides information about the various tasks to configure DHCP snooping.

Enable DHCP Snooping

Beginning in privileged EXEC mode, follow these steps to enable DHCP snooping on the 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# configure terminal`	Enters global configuration mode.
Step 3	**ip dhcp snooping** [**acl** | **database** | **glean** | **information** | **track** | **verify** | **vlan** | **wireless**] **Example:** `Device(config)# ip dhcp snooping`	Enables DHCP snooping globally. The **no** option disables DHCP snooping. Additionally, you can enable these optional keywords: • **acl**: Enables DHCP snooping on the specified ACL. • **database**: Enables DHCP snooping for the database agent or the binding file • **glean**: Enables read–only DHCP snooping function. • **information**: Enables the insertion and removal of Option 82 information for DHCP packets. • **track**: Enables DHCP server tracking when multiple DHCP servers are in the network. • **verify**: Verifies packets received on client associated address or gateway address which are part of DHCP snoop lookup. • **vlan**: Enables DHCP snooping on the specified VLANs. • **wireless**: Enables bootstrap protocol broadcast support in wireless DHCP.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Enable the DHCP snooping binding database agent

Beginning in privileged EXEC mode, follow these steps to enable and configure the DHCP snooping binding database agent on the 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# configure terminal`	Enters global configuration mode.
Step 3	**ip dhcp snooping database** {**flash** [*number*] **:** */filename* | **ftp:**//*user* **:** *password* **@** *host* */filename* | **http:**//[[*username* **:** *password*] **@**] {*hostname* | *host-ip*} [*/directory*] */image-name***.tar** | **rcp:**//*user* **@** *host*/*filename* | **scp:**//*user***@***host*/*filename* | **tftp:**//*hostfilename*} **Example:** `Device(config)# ip dhcp snooping database tftp://10.90.90.90/snooping-rp2`	Specifies the URL for the database agent or the binding file by using one of these forms: • **flash**[*number*]**:**/*filename* • **ftp:**//*user***:***password***@***host*/*filename* • **http:**//[[*username***:***password*]**@**]{*hostname* | *host-ip*}[/*directory*] /*image-name***.tar** • **rcp:**//*user***@***host*/*filename* • **scp:**//*user***@***host*/*filename* **Note** Before you configure SCP, you need to set the line console 0 transport output to *ssh* or *all*. • **tftp:**//*host*/*filename*
Step 4	**ip dhcp snooping database timeout** *seconds* **Example:** `Device(config)# ip dhcp snooping database timeout 300`	Specifies (in seconds) how long to wait for the database transfer process to finish before stopping the process. The default is 300 seconds. The range is 0 to 86400. Use 0 to define an infinite duration, which means to continue trying the transfer indefinitely.
Step 5	**ip dhcp snooping database write-delay** *seconds* **Example:**	Specifies the duration for which the transfer should be delayed after the binding database changes. The range is from 15 to 86400 seconds. The default is 300 seconds (5 minutes).

	Command or Action	Purpose
	`Device(config)# ip dhcp snooping database write-delay 15`	
Step 6	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 7	**ip dhcp snooping binding** *mac-address* **vlan** *vlan-id ip-address* **interface** *interface-id* **expiry** *seconds* **Example:** `Device# ip dhcp snooping binding 0001.1234.1234 vlan 1 172.20.50.5 interface gigabitethernet 1/1/0 expiry 1000`	(Optional) Adds binding entries to the DHCP snooping binding database. The *vlan-id* range is from 1 to 4904. The *seconds* range is from 1 to 4294967295. Enter this command for each entry that you add. Use this command when you are testing or debugging the switch.
Step 8	**show ip dhcp snooping database** [**detail**] **Example:** `Device# show ip dhcp snooping database detail`	Displays the status and statistics of the DHCP snooping binding database agent.

Configure an interface as a trusted or an untrusted source for DHCP gleaning

You can enable or disable DHCP gleaning on a device. You can configure an interface as a trusted or untrusted source of DHCP messages. Verify that no DHCP packets are dropped when DHCP gleaning is enabled on an untrusted interface or on a device port.

Note By default, DHCP gleaning is disabled, and all interfaces are untrusted.

You can configure DHCP trust on the following types of interfaces:

- Layer 2 Ethernet interfaces
- Layer 2 port-channel interfaces
- Layer 2 access and trunk interfaces

Procedure

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

	Command or Action	Purpose
	`Device> enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip dhcp snooping glean** **Example:** `Device(config)# ip dhcp snooping glean`	Enables DHCP gleaning.
Step 4	**interface** *type number* **Example:** `Device(config)# interface gigabitEthernet 1/0/1`	Enters interface configuration mode, where *type number* is the Layer 2 Ethernet interface which you want to configure as trusted or untrusted for DHCP snooping.
Step 5	[**no**] **ip dhcp snooping trust** **Example:** `Device(config-if)# ip dhcp snooping trust`	Configures the interface as a trusted interface for DHCP snooping. The **no** option configures the port as an untrusted interface.
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 7	**show ip dhcp snooping statistics** **Example:** `Device# show ip dhcp snooping statistics`	Displays packets that were dropped on the device port configured as an untrusted interface.
Step 8	**show ip dhcp snooping** **Example:** `Device# show ip dhcp snooping`	Displays DHCP snooping configuration information, including information about DHCP gleaning.

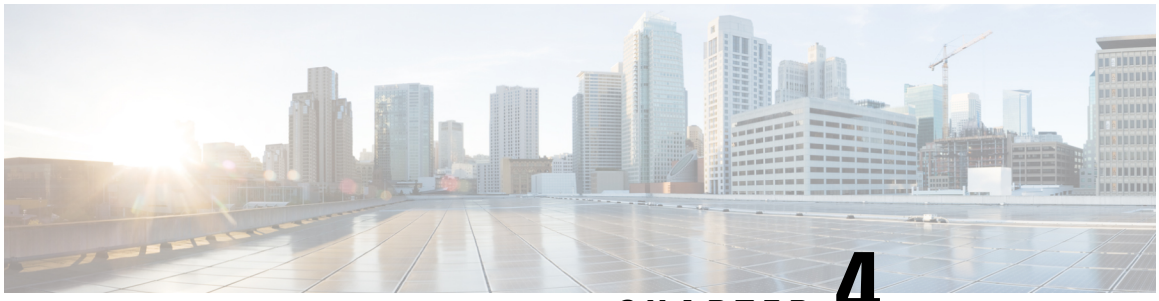
Monitor DHCP snooping information

Table 3: Commands for displaying DHCP information

Command	Purpose
show ip dhcp snooping	Displays general information about DHCP snooping.
show ip dhcp snooping binding	Displays only the dynamically configured bindings in the DHCP snooping binding database, also referred to as a binding table.
show ip dhcp snooping database	Displays the DHCP snooping binding database status and statistics.
show ip dhcp snooping statistics	Displays the DHCP snooping statistics in summary or detail form.

Command	Purpose
show ip source binding	Display the dynamically and statically configured bindings. **Note** This command is not supported on Cisco C9550 Series Smart Switches and Cisco C9610 Series Smart Switches.

Note If DHCP snooping is enabled and an interface changes to the down state, the switch does not delete the statically configured bindings.



CHAPTER 4

IPv6 FHS

- Feature history for IPv6 First Hop Security, on page 37
- Understand IPv6 First Hop Security, on page 38
- Prerequisites for IPv6 First Hop Security, on page 39
- Restrictions for IPv6 First Hop Security, on page 40
- Configure IPv6 First Hop Security, on page 40

Feature history for IPv6 First Hop Security

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 26.2.1	IPv6 Prefix Guard: This feature operates within the IPv6 source guard framework to ensure that only traffic from topologically correct source addresses is permitted.	Cisco C9350 Series Smart Switches Cisco C9550 Series Smart Switches
Cisco IOS XE 26.2.1ea	IPv6 First Hop Security: IPv6 First Hop Security feature support has been introduced.	Cisco C9550 Series Smart Switches
Cisco IOS XE 26.1.1a	IPv6 Source Guard: This feature validates the source of IPv6 traffic to prevent source address spoofing.	Cisco C9350 Series Smart Switches
Cisco IOS XE 17.18.1	IPv6 First Hop Security: IPv6 First Hop Security is a set of IPv6 security features that protects networks by mitigating such security breaches.	Cisco C9350 Series Smart Switches

Understand IPv6 First Hop Security

IPv6 networks face security threats and breaches in the form of router impersonation (man-in-the-middle attacks), address theft, address spoofing, misconfigurations errors, and so on. The First Hop Security in IPv6 (IPv6 FHS) is a set of IPv6 security features that protects networks by mitigating such security breaches. It does this by establishing security at the first switch connecting the end-hosts. The first hop for a host is very often a Layer 2 switch.

IPv6 FHS consists of the IPv6 Router Advertisement Guard and IPv6 DHCP Guard security features. Each of these security features addresses a different aspect of first hop security. To use a security feature, configure the corresponding policy.

Policies specify a particular behavior and must be attached to a target, which can be a physical interface, an EtherChannel interface, or a VLAN. An IPv6 software policy database service stores and accesses these policies. When a policy is configured or modified, the attributes of the policy are stored or updated in the software policy database, and applied as specified.

In addition to the security features, the IPv6 FHS Binding Table contains IPv6 neighbors connected to the device. A binding entry includes: IP and MAC address of the host, interface, VLAN, state of the entry, etc. This database or binding table is used by other features (such as IPv6 ND Inspection) to validate the link-layer address (LLA), the IPv4 or IPv6 address, and the prefix binding of neighbors, to prevent spoofing and redirect attacks. The binding table updates via the IPv6 Snooping feature and manually added static binding entries.



Note The IPv6 FHS Binding Table is supported through the Switch Integrated Security Feature (SISF) feature. For more information, refer the *Switch Integrated Security Features* chapter.

IPv6 Router Advertisement Guard

This feature enables the network administrator to block or reject unwanted or rogue Router Advertisement (RA) guard messages that arrive at the network device platform. RAs are used by devices to announce themselves on the link. The RA Guard feature processes the RAs and filters out invalid RAs sent by unauthorized devices. In host mode, all router-advertisement and router-redirect messages are disallowed on the port. The RA Guard feature compares the configuration data on the Layer 2 device with the incoming RA frame information. Once the Layer 2 device has validated the content of the RA frame and router redirect frame against the configuration, it forwards the RA to its unicast or multicast destination. If the RA frame content is not validated, the RA is dropped.

The SISF-based device-tracking mechanism operates by forwarding router solicitation packets on interfaces configured with RA guard policies and designated as router-facing. If no such interface exists, the router solicitation messages are dropped, which might delay the router discovery for onboarding hosts as they will be unable to discover the router until it sends a periodic unsolicited router advertisement.

IPv6 DHCP Guard

The IPv6 DHCP Guard feature blocks reply and advertisement messages that come from unauthorized DHCPv6 servers and relay agents. IPv6 DHCP guard can prevent forged messages from being entered in the binding table and block DHCPv6 server messages when they are received on ports that are not explicitly configured as facing a DHCPv6 server or DHCP relay.

To debug DHCP guard packets, use the **debug ipv6 snooping dhcp-guard** privileged EXEC command.

IPv6 source guard

The IPv6 Source Guard feature validates the source of IPv6 traffic to prevent source address spoofing. It deals exclusively with data packet traffic. Use this feature to deny traffic from unknown sources or from sources not assigned by a DHCP server.

A hardware-programmed (TCAM table) filter allows or denies traffic based on the source address. Ensure the binding table includes an entry for the source address to allow the filter to work properly. If the source address is in the binding table, the filter allows the packet into the network. If the address is not in the binding table, the filter denies entry and drops the packet. If you remove an entry from the binding table, the filter is removed and packets with that source address are dropped.

The IPv6 source guard feature requires device-tracking (SISF) to work. Device-tracking database supports static bindings for IPv4/IPv6, VLAN, interface, and optional MAC addresses. IP device tracking is used to obtain and maintain a list of devices that are connected to the device using an IP address. Device tracking listens to Neighbor Discovery Protocol (NDP) messages like router solicitation, neighbor solicitation, or DHCPv6 packets and builds a binding table, mapping the IPv6 address, MAC address, VLAN, and port of the device. IPV6 source guard relies on this binding table information to generate the IP source guard table to permit, punt, and drop packets.

The IPv6 source guard feature supports 4,000 entries per device.

When configuring this feature, consider these:

- An IPv6 source guard policy cannot be attached to a VLAN. It is supported only at the interface level.
- When you configure IPv4 and IPv6 source guard together on an interface, use the **ip verify source mac-check** command instead of the **ip verify source tracking mac-check** command. IPv4 connectivity on a given port might be interrupted due to two different sets of filtering rules: one for IPv4 (IP-filter) and one for IPv6 (IP-MAC filter).
- When IPv6 source guard is enabled on a switch port, NDP or DHCP snooping must be enabled on the interface to which the switch port belongs. Otherwise, all data traffic from this port will be blocked.

To use this feature, you must configure an IPv6 source guard policy and attach it to a target.

To debug source-guard packets, use the **debug ipv6 snooping source-guard** privileged EXEC command.

IPv6 Prefix Guard

The IPv6 prefix guard feature works within the IPv6 source guard feature, to enable the device to deny traffic originated from non-topologically correct addresses.

IPv6 prefix guard is often used when IPv6 prefixes are delegated to devices (for example, home gateways) using DHCP prefix delegation. The feature discovers ranges of addresses assigned to the link and blocks any traffic sourced with an address outside this range.

Prerequisites for IPv6 First Hop Security

Configure the necessary IPv6 enabled SDM template.

Restrictions for IPv6 First Hop Security

Legacy deprecated configurations from older platforms or releases will not work on newer platforms or releases. We recommend you convert legacy commands to the SISF-based device tracking CLI commands. For more information, refer *Switch Integrated Security Features* chapter.

These restrictions apply when applying FHS policies to EtherChannel interfaces (Port Channels):

- A physical port with an FHS policy attached cannot join an EtherChannel group.
- An FHS policy cannot be attached to a physical port when it is a member of an EtherChannel group.
- If device-tracking is enabled, DHCPv6 server lease time should be at least 2 hours.

Configure IPv6 First Hop Security

This section provides information about the various tasks to configure IPv6 first hop security.

Configure the IPv6 binding table content

Follow these steps to configure IPv6 binding table content:

Procedure

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**] **ipv6 neighbor binding** [**vlan** *vlan-id* {*ipv6-address* **interface** *interface_type stack/module/port hw_address* [**reachable-lifetimevalue** [*seconds* | **default** | **infinite**] | [**tracking**{ [**default** | **disable**] [**reachable-lifetimevalue** [*seconds* | **default** | **infinite**] | [**enable** [**reachable-lifetimevalue** [*seconds* | **default** | **infinite**] | [**retry-interval** {*seconds* | **default** [**reachable-lifetimevalue** [*seconds* | **default** | **infinite**]}]

Example:

```
Device(config)# ipv6 neighbor binding
```

Adds a static entry to the binding table database.

Step 4 [**no**] **ipv6 neighbor binding max-entries** *number* [**mac-limit** *number* | **port-limit** *number* [**mac-limit** *number*] | **vlan-limit** *number* [[**mac-limit** *number*] | [**port-limit** *number* [**mac-limit***number*]]]

Example:

```
Device(config)# ipv6 neighbor binding max-entries 30000
```

Specifies the maximum number of entries that are allowed to be inserted in the binding table cache.

Step 5 **ipv6 neighbor binding logging**

Example:

```
Device(config)# ipv6 neighbor binding logging
```

Enables the logging of binding table main events.

Step 6 **exit**

Example:

```
Device(config)# exit
```

Exits global configuration mode and returns to privileged EXEC mode.

Step 7 **show ipv6 neighbor binding**

Example:

```
Device# show ipv6 neighbor binding
```

Displays contents of a binding table.

Configure an IPv6 router advertisement guard policy

Follow these steps to configure an IPv6 router advertisement policy:

Procedure

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 **ipv6 nd raguard policy** *policy-name*

Example:

```
Device(config)# ipv6 nd raguard policy example_policy
```

Specifies the RA guard policy name and enters RA guard policy configuration mode.

Step 4 [**no**]**device-role** {**host** | **monitor** | **router** | **switch**}

Example:

```
Device(config-nd-raguard)# device-role switch
```

Specifies the role of the device attached to the port. The default is **host**.

Note
For a network with both host-facing ports and router-facing ports, along with a RA guard policy configured with **device-role host** on host-facing ports or vlan, it is mandatory to configure a RA guard policy with **device-role router** on router-facing ports to allow the RA Guard feature to work properly.

Step 5 **hop-limit** {**maximum** | **minimum**} *value*

Example:

```
Device(config-nd-raguard)# hop-limit maximum 33
```

Enables filtering of Router Advertisement messages by the Hop Limit value. A rogue RA message may have a low Hop Limit value (equivalent to the IPv4 Time to Live) that when accepted by the host, prevents the host from generating traffic to destinations beyond the rogue RA message generator. An RA message with an unspecified Hop Limit value is blocked.

The range for Maximum and Minimum Hop Limit values is 1 to 255.

If not configured, this filter is disabled. Configure **minimum** to block RA messages with Hop Limit values lower than the value you specify. Configure **maximum** to block RA messages with Hop Limit values greater than the value you specify.

Step 6 **managed-config-flag** {**off** | **on**}

Example:

```
Device(config-nd-raguard)# managed-config-flag on
```

Enables filtering of Router Advertisement messages by the managed address configuration, or "M" flag field. A rouge RA message with an M field of 1 can cause a host to use a rogue DHCPv6 server. If not configured, this filter is disabled.

- **On**: Accepts and forwards RA messages with an M value of 1, blocks those with 0.
- **Off**: Accepts and forwards RA messages with an M value of 0, blocks those with 1.

Step 7 **match** {**ipv6 access-list** *list* | **ra prefix-list** *list*}

Example:

```
Device(config-nd-raguard)# match ipv6 access-list example_list
```

Matches a specified prefix list or access list.

Step 8 **other-config-flag** {**on** | **off**}

Example:

```
Device(config-nd-raguard)# other-config-flag on
```

Enables filtering of Router Advertisement messages by the Other Configuration, or "O" flag field. A rouge RA message with an O field of 1 can cause a host to use a rogue DHCPv6 server. If not configured, this filter is disabled.

- **On**: Accepts and forwards RA messages with an O value of 1, blocks those with 0.
- **Off**: Accepts and forwards RA messages with an O value of 0, blocks those with 1.

Step 9 [**no**]**router-preference maximum** {**high** | **medium** | **low**}

Example:

```
Device(config-nd-raguard)# router-preference maximum high
```

Enables filtering of Router Advertisement messages by the router preference flag. If not configured, this filter is disabled.

- **high**: Accepts RA messages with the router preference set to high, medium, or low.
- **medium**: Blocks RA messages with the router preference set to high.
- **low**: Blocks RA messages with the router preference set to medium and high.

Step 10 **trusted-port**

Example:

```
Device(config-nd-raguard)# trusted-port
```

When configured as a trusted port, all attached devices are trusted, and no further message verification is performed.

Step 11 **default** {**device-role** | **hop-limit** {**maximum** | **minimum**} | **managed-config-flag** | **match** {**ipv6 access-list** | **ra prefix-list**} | **other-config-flag** | **router-preference maximum** | **trusted-port**}

Example:

```
Device(config-nd-raguard)# default hop-limit
```

Restores a command to its default value.

Step 12 **end**

Example:

```
Device(config-nd-raguard)# end
```

Exits RA Guard policy configuration mode and returns to privileged EXEC mode.

Step 13 **show ipv6 nd raguard policy** *policy_name*

Example:

```
Device# show ipv6 nd raguard policy example_policy
```

(Optional) Displays the ND guard policy configuration.

Attach an IPv6 router advertisement guard policy to an interface

Follow these steps to attach an IPv6 router advertisement policy to an interface or to VLANs on the interface:

Procedure

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 1/1/4
```

Specifies an interface type and identifier; enters the interface configuration mode.

Step 4 **ipv6 nd raguard** [**attach-policy** *policy_name* [**vlan** {*vlan_ids* | **add** *vlan_ids* | **except** *vlan_ids* | **none** | **remove** *vlan_ids* | **all**}] | **vlan** [{*vlan_ids* | **add** *vlan_ids* | **except** *vlan_ids* | **none** | **remove** *vlan_ids* | **all**}]

Example:

```
Device(config-if)# ipv6 nd raguard attach-policy example_policy
Device(config-if)# ipv6 nd raguard attach-policy example_policy vlan 222,223,224
Device(config-if)# ipv6 nd raguard vlan 222, 223,224
```

Attaches the Neighbor Discovery Inspection policy to the interface or the specified VLANs on that interface. The default policy is attached if the **attach-policy** option is not used.

Step 5 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

Attach an IPv6 router advertisement guard policy to a Layer 2 EtherChannel

Follow these steps to attach an IPv6 router advertisement guard policy on an EtherChannel interface or VLAN:

Procedure

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 range** *type number*

Example:

```
Device(config)# interface range Port-channel 11
```

Specifies the port-channel interface name assigned when the EtherChannel was created. Enters interface range configuration mode.

Tip
Enter the **show interfaces summary** command in privileged EXEC mode for quick reference to interface names and types.

Step 4 **ipv6 nd raguard** [**attach-policy** *policy_name* [**vlan** {*vlan_ids* | **add** *vlan_ids* | **except** *vlan_ids* | **none** | **remove** *vlan_ids* | **all**}] | **vlan** [{*vlan_ids* | **add** *vlan_ids* | **except** *vlan_ids* | **none** | **remove** *vlan_ids* | **all**}]

Example:

```
Device(config-if-range)# ipv6 nd raguard attach-policy example_policy
Device(config-if-range)# ipv6 nd raguard attach-policy example_policy vlan 222,223,224
Device(config-if-range)# ipv6 nd raguard vlan 222, 223,224
```

Attaches the RA Guard policy to the interface or the specified VLANs on that interface. The default policy is attached if the **attach-policy** option is not used.

Step 5 **end**

Example:

```
Device(config-if-range)# end
```

Exits interface range configuration mode and returns to privileged EXEC mode.

Attach an IPv6 router advertisement guard policy to VLANs globally

Follow these steps to attach an IPv6 router advertisement policy to VLANs regardless of interface:

Procedure

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 configuration** *vlan_list*

Example:

```
Device(config)# vlan configuration 335
```

Specifies the VLANs to which the IPv6 RA Guard policy will be attached, and enters VLAN interface configuration mode.

Step 4 **ipv6 dhcp guard** [**attach-policy** *policy_name*]

Example:

```
Device(config-vlan-config)# ipv6 nd raguard attach-policy example_policy
```

Attaches the IPv6 RA Guard policy to the specified VLANs across all switch and stack interfaces. The default policy is attached if the **attach-policy** option is not used.

Step 5 **end**

Example:

```
Device(config-vlan-config)# end
```

Exits VLAN interface configuration mode and returns to privileged EXEC mode.

Configure an IPv6 DHCP guard policy

Follow these steps to configure an IPv6 DHCP (DHCPv6) guard policy:

Procedure

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 **ipv6 dhcp guard policy** *policy-name*

Example:

```
Device(config)# ipv6 dhcp guard policy example_policy
```

Specifies the DHCPv6 Guard policy name and enters DHCPv6 Guard Policy configuration mode.

Step 4 **device-role** {**client** | **server**}

Example:

```
Device(config-dhcp-guard)# device-role server
```

(Optional) Filters out DHCPv6 replies and DHCPv6 advertisements on the port that are not from a device of the specified role. Default is **client**.

- **client**: Default value, specifies that the attached device is a client. Server messages are dropped on this port.
- **server**: Specifies that the attached device is a DHCPv6 server. Server messages are allowed on this port.

Step 5 **match server access-list** *ipv6-access-list-name*

Example:

```
;;Assume a preconfigured IPv6 Access List as follows:
Device(config)# ipv6 access-list my_acls
Device(config-ipv6-acl)# permit host 2001:BD8:::1 any

;;configure DCHPv6 Guard to match approved access list.
Device(config-dhcp-guard)# match server access-list my_acls
```

(Optional). Enables verification that the advertised DHCPv6 server or relay address is from an authorized server access list (The destination address in the access list is 'any'). If not configured, this check will be bypassed. An empty access list is treated as a permit all.

Step 6 **match reply prefix-list** *ipv6-prefix-list-name*

Example:

```
;;Assume a preconfigured IPv6 prefix list as follows:
Device(config)# ipv6 prefix-list my_prefix permit 2001:DB8::/64 le 128

;; Configure DCHPv6 Guard to match prefix
Device(config-dhcp-guard)#  match reply prefix-list my_prefix
```

(Optional) Enables verification of the advertised prefixes in DHCPv6 reply messages from the configured authorized prefix list. If not configured, this check will be bypassed. An empty prefix list is treated as a permit.

Step 7 **preference** {**max** *limit* | **min** *limit*}

Example:

```
Device(config-dhcp-guard)# preference max 250
Device(config-dhcp-guard)#preference min 150
```

Configure **max** and **min** when **device-role** is **server** to filter DCHPv6 server advertisements by the server preference value. The defaults permit all advertisements.

- **max** *limit*: (0 to 255) (Optional) Enables verification that the advertised preference (in preference option) is less than the specified limit. Default is 255. If not specified, this check will be bypassed.
- **min** *limit*: (0 to 255) (Optional) Enables verification that the advertised preference (in preference option) is greater than the specified limit. Default is 0. If not specified, this check will be bypassed.

Step 8 **trusted-port**

Example:

```
Device(config-dhcp-guard)# trusted-port
```

(Optional) **trusted-port**: Sets the port to a trusted mode. No further policing takes place on the port.

Note
If you configure a trusted port then the device-role option is not available.

Step 9 **default** {**device-role** | **trusted-port**}

Example:

```
Device(config-dhcp-guard)# default device-role
```

(Optional) **default**: Sets a command to its defaults.

Step 10 **end**

Example:

```
Device(config-dhcp-guard)# end
```

Exits DHCPv6 Guard Policy configuration mode and returns to privileged EXEC mode.

Step 11 **show ipv6 dhcp guard policy** *policy_name*

Example:

```
Device# show ipv6 dhcp guard policy example_policy
```

(Optional) Displays the configuration of the IPv6 DHCP guard policy. Omitting the *policy_name* variable displays all DHCPv6 policies.

Attach an IPv6 DHCP guard policy to an interface

Follow these steps to to attach an IPv6 DHCP guard policy to an interface or a VLAN:

Procedure

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 1/1/4
```

Specifies an interface type and identifier, and enters interface configuration mode.

Step 4 **ipv6 dhcp guard** [**attach-policy** *policy_name* [**vlan** {*vlan_ids* | **add** *vlan_ids* | **except** *vlan_ids* | **none** | **remove** *vlan_ids* | **all**}] | **vlan** [{*vlan_ids* | **add** *vlan_ids* | **except** *vlan_ids* | **none** | **remove** *vlan_ids* | **all**}]

Example:

```
Device(config-if)# ipv6 dhcp guard attach-policy example_policy
Device(config-if)# ipv6 dhcp guard attach-policy example_policy vlan 222,223,224
Device(config-if)# ipv6 dhcp guard vlan 222, 223,224
```

Attaches the DHCP Guard policy to the interface or the specified VLANs on that interface. The default policy is attached if the **attach-policy** option is not used.

Step 5 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

Attach an IPv6 DHCP guard policy to a Layer 2 EtherChannel

Follow these steps to attach an IPv6 DHCP guard policy on an EtherChannel interface or VLAN:

Procedure

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 range** *interface_name*

Example:

```
Device(config)# interface range Port-channel 11
```

Specify the port-channel interface name assigned when the EtherChannel was created. Enters interface range configuration mode.

Tip
Enter the **show interfaces summary** command in privileged EXEC mode for quick reference to interface names and types.

Step 4 **ipv6 dhcp guard** [**attach-policy** *policy_name* [**vlan** {*vlan_ids* | **add** *vlan_ids* | **except** *vlan_ids* | **none** | **remove** *vlan_ids* | **all**}] | **vlan** [{*vlan_ids* | **add** *vlan_ids* | **except** *vlan_ids* | **none** | **remove** *vlan_ids* | **all**}]

Example:

```
Device(config-if-range)# ipv6 dhcp guard attach-policy example_policy
Device(config-if-range)# ipv6 dhcp guard attach-policy example_policy vlan 222,223,224
Device(config-if-range)# ipv6 dhcp guard vlan 222, 223,224
```

Attaches the DHCP Guard policy to the interface or the specified VLANs on that interface. The default policy is attached if the **attach-policy** option is not used.

Step 5 **end**

Example:

```
Device(config-if-range)# end
```

Exits interface range configuration mode and returns to privileged EXEC mode.

Attach an IPv6 DHCP guard policy to VLANs globally

Follow these steps to attach an IPv6 DHCP guard policy to VLANs across multiple interfaces:

Procedure

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 configuration** *vlan_list*

Example:

```
Device(config)# vlan configuration 334
```

Specifies the VLANs to which the IPv6 Snooping policy will be attached, and enters VLAN interface configuration mode.

Step 4 **ipv6 dhcp guard** [**attach-policy** *policy_name*]

Example:

```
Device(config-vlan-config)# ipv6 dhcp guard attach-policy example_policy
```

Attaches the IPv6 Neighbor Discovery policy to the specified VLANs across all switch and stack interfaces. The default policy is attached if the **attach-policy** option is not used.

Step 5 **end**

Example:

```
Device(config-vlan-config)# end
```

Exits VLAN interface configuration mode and returns to privileged EXEC mode.

Configure an IPv6 source guard policy

Follow these steps to configure an IPv6 source guard policy:

Procedure

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 **device-tracking binding vlan** *vlan_id* {*ipv4_add* | *ipv6_add*} [**interface** *inteface_type_no*] [*48-bit-hardware-address*]

Example:

```
Device(config)# device-tracking binding vlan 100 192.0.2.1 interface tengigabitethernet1/0/1
 00:00:5e:00:53:af
```

Creates a static device-tracking binding entry.

Step 4 **ipv6 source-guard policy** *policy_name*

Example:

```
Device(config)# ipv6 source-guard policy example_policy
```

Specifies the IPv6 source guard policy name and enters IPv6 source guard policy configuration mode.

Step 5 [**deny global-autoconf**] [**permit link-local**] [**default**{. . . }] [**exit**] [**no**{. . . }]

Example:

```
Device(config-sisf-sourceguard)# deny global-autoconf
```

(Optional) Defines the IPv6 Source Guard policy.

- **deny global-autoconf**: Denies data traffic from auto-configured global addresses. This is useful when all global addresses on a link are DHCP-assigned and the administrator wants to block hosts with self-configured addresses to send traffic.
- **permit link-local**: Allows all data traffic that is sourced by a link-local address.

Note
Trusted option under source guard policy is not supported.

Step 6 **validate** {**address** | **prefix**}

Example:

```
Device(config-sisf-sourceguard)# validate address
```

(Optional) Validates against addresses or prefixes.

- **address**: Validates against addresses learned in NDP/DHCP traffic.
- **prefix**: Validates against prefixes learned in RA or DHCP-PD traffic.

Step 7 **end**

Example:

```
Device(config-sisf-sourceguard)# end
```

Exits IPv6 Source Guard policy configuration mode and returns to privileged EXEC mode.

Step 8 **show ipv6 source-guard policy** *policy_name*

Example:

```
Device# show ipv6 source-guard policy example_policy
```

Shows the policy configuration and all the interfaces where the policy is applied.

Attach an IPv6 source guard policy to an interface

Before you begin

IPv6 source guard learns entries from device-tracking database, and so enable device-tracking on the interface for IPv6 source guard to work properly.

Follow these steps to to attach an IPv6 source guard policy to an interface:

Procedure

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 1/1/4
```

Specifies an interface type and identifier; enters interface configuration mode.

Step 4 **ipv6 source-guard** [**attach-policy** <*policy_name*>]

Example:

```
Device(config-if)# ipv6 source-guard attach-policy example_policy
```

Attaches the IPv6 source guard policy to the interface. The default policy is attached if the **attach-policy** option is not used.

Step 5 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

Step 6 **show ipv6 source-guard policy** *policy_name*

Example:

```
Device#(config)# show ipv6 source-guard policy example_policy
```

Shows the policy configuration and all the interfaces where the policy is applied.

Attach an IPv6 source guard policy to a Layer 2 EtherChannel interface

Before you begin

IPv6 source guard learns entries from device-tracking database, and so enable device-tracking on the interface for IPv6 source guard to work properly.

Follow these steps to attach an IPv6 source guard policy to a Layer 2 EtherChannel interface:

Procedure

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 port-channel** *port-channel-number*

Example:

```
Device(config)# interface Port-channel 4
```

Specifies an interface type and port number and places the switch in the port channel configuration mode.

Step 4 **ipv6 source-guard** [**attach-policy** <*policy_name*>]

Example:

```
Device(config-if)# ipv6 source-guard attach-policy example_policy
```

Attaches the IPv6 source guard policy to the interface. The default policy is attached if the **attach-policy** option is not used.

Step 5 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

Step 6 **show ipv6 source-guard policy** *policy_name*

Example:

```
Device# show ipv6 source-guard policy example_policy
```

Shows the policy configuration and all the interfaces where the policy is applied.

Configure an IPv6 Prefix Guard

Follow these steps to configure an IPv6 source guard policy:

Note To allow routing protocol control packets sourced by a link-local address when prefix guard is applied, enable the **permit link-local** command in the source-guard policy configuration mode.

Procedure

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 **ipv6 source-guard policy** *source-guard-policy*

Example:

```
Device(config)# ipv6 source-guard policy my_snooping_policy
```

Defines an IPv6 source-guard policy name and enters switch integrated security features source-guard policy configuration mode.

Step 4 **no validate address**

Example:

```
Device(config-sisf-sourceguard)# no validate address
```

Disables the validate address feature and enables the IPv6 prefix guard feature to be configured.

Step 5 **validate prefix**

Example:

```
Device(config-sisf-sourceguard)# validate prefix
```

Enables IPv6 source guard to perform the IPv6 prefix-guard operation.

Step 6 **exit**

Example:

```
Device(config-sisf-sourceguard)# exit
```

Exits switch integrated security features source-guard policy configuration mode and returns to privileged EXEC mode.

Step 7 **show ipv6 source-guard policy** [*source-guard-policy*]

Example:

```
Device# show ipv6 source-guard policy policy1
```

Displays the IPv6 source-guard policy configuration.

Attach an IPv6 Prefix Guard Policy to an Interface

Follow these steps to attach an IPv6 prefix guard policy to an interface:

Procedure

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 1/1/4
```

Specifies an interface type and identifier, and enters interface configuration mode.

Step 4 **ipv6 source-guard attach-policy** *policy_name*

Example:

```
Device(config-if)# ipv6 source-guard attach-policy example_policy
```

Attaches the IPv6 Source Guard policy to the interface. The default policy is attached if the attach-policy option is not used.

Step 5 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

Step 6 **show ipv6 source-guard policy** *policy_name*

Example:

```
Device(config-if)# show ipv6 source-guard policy example_policy
```

Shows the policy configuration and all the interfaces where the policy is applied.

Attach an IPv6 Prefix Guard Policy to a Layer 2 EtherChannel Interface

Follow these steps to attach an IPv6 prefix guard policy to a Layer 2 EtherChannel interface:

Procedure

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 port-channel** *port-channel-number*

Example:

```
Device(config)# interface Port-channel 4
```

Specifies an interface type and port number and places the switch in the port channel configuration mode.

Step 4 **ipv6 source-guard** [**attach-policy** <*policy_name*>]

Example:

```
Device(config-if)# ipv6 source-guard attach-policy example_policy
```

Attaches the IPv6 Source Guard policy to the interface. The default policy is attached if the attach-policy option is not used.

Step 5 **end**

Example:

```
Device(config-if)# end
```

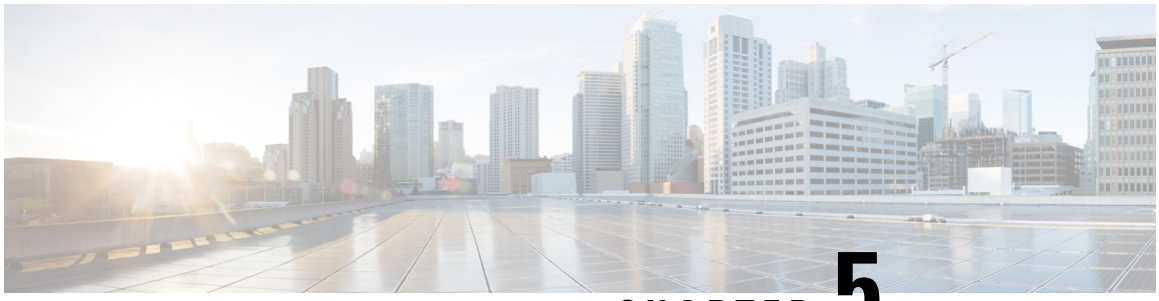
Exits interface configuration mode and returns to privileged EXEC mode.

Step 6 **show ipv6 source-guard policy** *policy_name*

Example:

```
Device(config)# show ipv6 source-guard policy example_policy
```

Shows the policy configuration and all the interfaces where the policy is applied.



CHAPTER 5

SISF

- Feature history for SISF, on page 59
- Understand SISF, on page 59
- Guidelines to create a policy, on page 76
- Guidelines to apply a policy, on page 76
- Configure SISF, on page 76
- Configuration examples, on page 87

Feature history for SISF

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 26.2.1ea	SISF: SISF feature support has been introduced.	Cisco C9550 Series Smart Switches
Cisco IOS XE 17.18.1	SISF: SISF is a framework for optimizing security in Layer 2 domains. It merges the IPDT and certain IPv6 FHS functionality, to simplify the migration from IPv4 to IPv6 stack or a dual-stack.	Cisco C9350 Series Smart Switches

Understand SISF

Switch Integrated Security Features (SISF) is a framework for optimizing security in Layer 2 domains. It merges the IP Device Tracking (IPDT) and *certain* IPv6 first-hop security (FHS) functionality[2], to simplify the migration from IPv4 to IPv6 stack or a dual-stack.

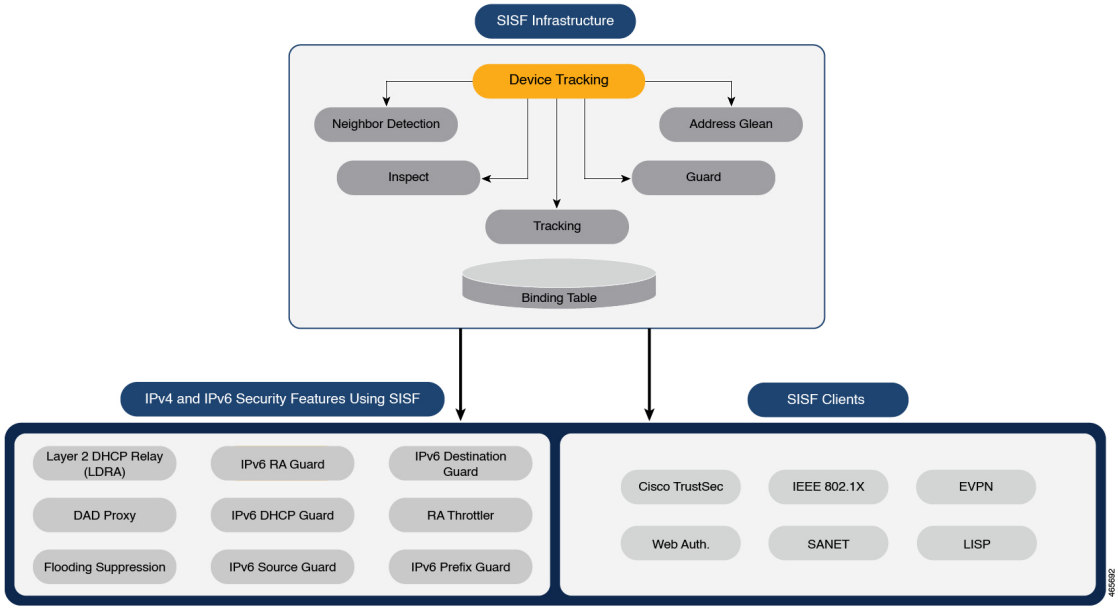
The SISF infrastructure provides a unified database that is used by:

[2] IPv6 Snooping Policy, IPv6 FHS Binding Table Content, and IPv6 Neighbor Discovery Inspection

- IPv6 FHS features: IPv6 Router Advertisement (RA) Guard, IPv6 DHCP Guard, Layer 2 DHCP Relay, IPv6 Duplicate Address Detection (DAD) Proxy, and Flooding Suppression.
- Cisco TrustSec, IEEE 802.1X, Locator ID Separation Protocol (LISP), Ethernet VPN (EVPN), and Web Authentication act as clients for SISF.

The figure illustrates the SISF Framework.

Figure 3: SISF Framework



Note The terms *SISF*, *device-tracking*, and *SISF-based device-tracking* are used interchangeably in this document and refer to the same feature. Neither term is used to mean or should be confused with the legacy IPDT or IPv6 Snooping features.

The binding table

The SISF infrastructure is built around the binding table. The binding table contains information about the hosts that are connected to the ports of a switch and the IP and MAC address of these hosts. This action creates a physical map of all the hosts connected to a switch.

Each entry in a binding table provides the following information about a connected host:

- IPv4 or IPv6 address of the host.
- MAC address of the host. The same MAC address may be linked to an IPv4 and IPv6 address.
- The interface or port on the switch that the host is connected to, and the associated VLAN.
- The state of the entry, which indicates the reachability of the entry.

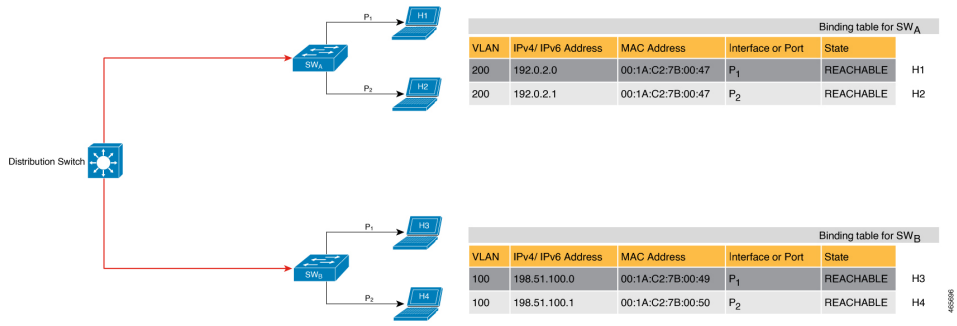
The figure shows a simple network topology and a representative binding table for each access switch in the network. SW_A and SW_B are the two access switches in the network. The two access switches are connected to the same distribution switch. H1, H2, H3, H4 are the hosts.

This example shows a distributed binding table where each access switch in the network has its table. An alternative setup could be one centralized binding table on the distribution switch containing entries of SW_A *and* SW_B.

Having a distributed or a centralized binding table is a key design choice in the process of implementing SISF in your network and is covered in greater detail in the Understand policy parameters, on page 66 section.

Figure 4: Binding table

(Click on the image to see the details more clearly.)



States and lifetime of a binding table entry

The state of an entry shows whether the host is reachable. The binding table entry can be in stable states such as REACHABLE, DOWN, and STALE. When changing from one state to another, an entry may have other temporary or transitional states such as: VERIFY, INCOMPLETE, and TENTATIVE.

The duration an entry stays in a state depends on its lifetime and successful validation. The lifetime of an entry can be policy-driven or configured globally.

To configure the REACHABLE, DOWN, and STALE lifetimes, enter the command in global configuration mode.

device-tracking binding { **reachable-lifetime** { *seconds* | **infinite** } | **stale-lifetime** { *seconds* | **infinite** } | **down-lifetime** { *seconds* | **infinite** } }

Note For DHCP-originated entries, the above configuration does not apply as the DHCP lease time overrides the SISF stale-lifetime and down-lifetime configuration that are globally defined or specified by the policy.

State: Reachable

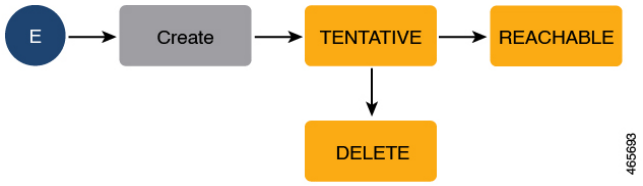
If an entry is in this state, it means the host (IP and MAC address) is verified and valid. A reachable entry has a default lifetime of five minutes. You can also configure a duration. By configuring a reachable-lifetime, you specify how long a host can remain in a REACHABLE state, after the last incoming control packet from that host.

If an event is detected before the entry's reachable lifetime expires, then the reachable lifetime is reset.

To qualify for the REACHABLE state, a new entry goes through the process illustrated in the figure below. The switch detects an event (E), such as an incoming control packet from a connected host and creates an entry. Various events cause the creation of an entry, and these are described in the Binding table sources, on page 63 section. After creating an entry, it goes through transient states like TENTATIVE or INCOMPLETE. While in a transitional state, the switch validates and confirms the integrity of the binding entry. If the entry is found to be valid, then the state changes to REACHABLE.

But if an address theft or similar event is detected, then the entry is regarded as invalid and is deleted. For example, if an attacker sends unsolicited neighbor advertisement messages with the same IP as the target IP and their own MAC address to redirect traffic.

Figure 5: Creation of a reachable entry

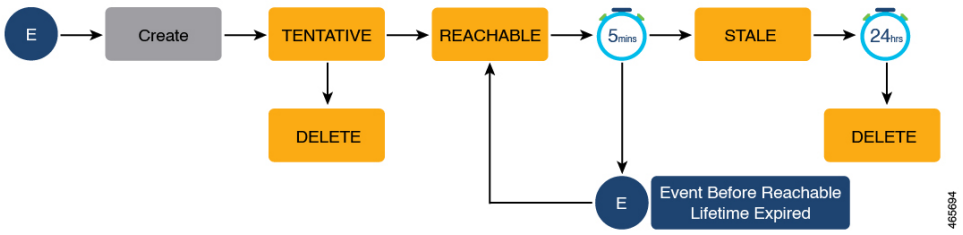


State: Stale

If an entry is in this state it means that the entry's reachable lifetime has expired and the corresponding host is still silent (no incoming packets from the host). A stale entry has a default lifetime of 24 hours. You can also configure a duration. The system deletes an entry if it stays in the STALE state beyond its stale lifetime.

This is illustrated in the figure below which depicts the lifecycle of an entry.

Figure 6: Lifecycle of an entry



State: Down

If an entry is in this state, it means that the host's connecting interface is down. A down entry has a default lifetime of 24 hours. Additionally, you can configure a duration. An entry is deleted if it remains in the DOWN state beyond its down lifetime.

Poll a host and update the binding table entry

Polling involves checking whether the host is connected and communicating. In addition to determining an entry's state, you can use polling to reconfirm an entry's state.

You can enable polling with the **device-tracking tracking** command in global configuration mode. After you do, you still have the flexibility to turn polling on or off for a particular interface or VLAN. For this, configure the **tracking enable** or **tracking disable** keywords in the policy (the device-tracking configuration mode).

When polling is enabled, the switch polls the host at the specified interval, thus reconfirming its reachability for the duration of its reachable lifetime.

When polling is enabled, the switch sends up to three polling requests after the reachable lifetime expires at intervals determined by the system. You can also configure this interval with the **device-tracking tracking retry-interval** *seconds* command in global configuration mode.

The figure below depicts the lifecycle of an entry where the host is polled. Default reachable and stale lifetimes, and retry intervals are used in figure:

When an event (E) is detected, a REACHABLE entry is created.

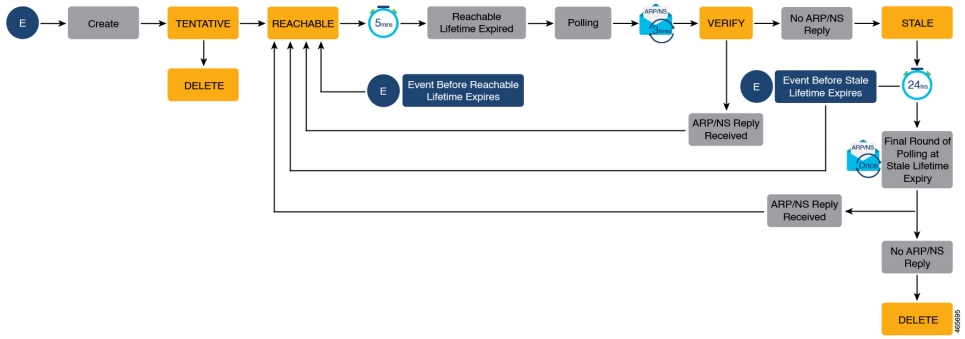
If an event is detected *during* the reachable lifetime, the reachable lifetime timer is reset.

The switch sends a polling request after the reachable lifetime expires. The switch polls the host up to three times at fixed, system-determined intervals. The polling request may be in the form of a unicast Address Resolution Protocol (ARP) probe or a Neighbor Solicitation message. During this time, the system changes the entry state to VERIFY. If a polling response is received (thus confirming reachability of the host), the state of the entry changes back to REACHABLE.

If the switch does not receive a polling response after three attempts, the entry changes to the STALE state. It remains in this state for 24 hours. If an event is detected during the stale lifetime, the state of the entry is changed back to REACHABLE. At expiry of the stale lifetime, the device sends one final polling to ascertain reachability. If this final polling attempt receives a reply, the state of the entry is changed back to REACHABLE. If the final polling attempt does not receive a response, the entry is deleted.

Figure 7: Lifecycle of an entry where the host is polled

(Click on the image to see the details more clearly.)



The DHCP lease time determines how long a DHCP-originated binding remains valid in the table. When the DHCP lease time expires, the entry is removed from the table without additional reachability probes.

Binding table sources

The following are the sources of information and events that cause the creation and update of a binding table entry:

- Learning events that dynamically populate the binding table:
  - Dynamic Host Configuration Protocol (DHCP) negotiation (DHCP REQUEST, and DHCP REPLY). This includes DHCPv4 and DHCPv6.
  - Address Resolution Protocol (ARP) packets.

ARP packets are throttled to mitigate high CPU utilization scenarios. Within a five-second window, a maximum of 50 ARP packets from the same source are processed by SISF. Note that the limit of 50 in five seconds is for each binding entry, that is, for each source IP.

All ARP (ARP REQUEST and ARP REPLY) packets are dropped if

- the limit is reached and
- the security level of the device tracking policy is set to guard.

- Neighbor Discovery Protocol (NDP) packets.
- Multiple Identity Association-Nontemporary Address (IA_NA) and Identity Association-Prefix Delegation (IA_PD).

  In some cases, a network device can request and receive more than one IPv6 address from the DHCP server. This may be done to provide addresses to multiple clients of the device, such as when a residential gateway requests addresses to distribute to network clients. When the device sends a DHCPv6 packet, it includes all of the addresses that have been assigned to the device.

  When SISF analyzes a DHCPv6 packet, it examines the IA_NA (Identity Association-Nontemporary Address) and IA_PD (Identity Association-Prefix Delegation) components of the packet and extracts each IPv6 address contained in the packet. SISF adds each extracted address to the binding table.

Entries created through learning events are called dynamic entries. These entries, shown in the device-tracking database details output, are prefixed with abbreviations that indicate the type of learning event, such as "ARP" for ARP packets.

- Configuring static binding entries.

  If there are silent but reachable hosts in the Layer 2 domain, you can create static binding entries to retain binding information even if the host becomes silent.

  Manually add a static binding entry to the binding table using this command in global configuration mode:

  device-tracking binding vlan *vlan_id* { *ipv4_add ipv6_add ipv6_prefix* } [**interface** *inteface_type_no*] [*48-bit-hardware-address*] [**reachable-lifetime** { *seconds* | **default** | **infinite** } | **tracking** { **default** | **disable** | **enable** [**retry-interval** { *seconds* | **default** }] } [**reachable-lifetime** { *seconds* | **default** | **infinite** }]]

  Static entries in the **show device-tracking database details** output are prefixed with the letter "S".

  You can configure a reachable lifetime for a static entry. The stale and down lifetime timer is fixed by the system as **infinite** (for an entry in the STALE or DOWN state, the output of the **show device-tracking database** command displays the `Time Left` column as “N/A”). When a static entry enters the STALE or DOWN state, it remains in this state indefinitely in the binding table.

  A static entry can be removed from the binding table only by the actions listed below. It cannot be deleted from the binding table by using **clear** commands or by any other event:

  - Remove the entry by configuring the **no** form of the above command.
  - A local entry replaces the static entry.

    A local entry is an entry that is automatically created by the system when you configure an SVI on the device. When configuring the SVI, if you use the same IP address as the static entry then the static entry is replaced with the local entry, because the local entry has a higher priority.

In the output of the **show device-tracking database details** privileged EXEC command, local entries are prefixed with the letter "L".

For more information about static binding entries, see the **device-tracking binding** command in the command reference.

Note A specific scenario allows a ping to result in a device-tracking entry. If a sender's ARP cache or IPv6 neighbor table does not have the target's IP address yet, then a ping triggers an ARP packet for IPv4, or ND packet for IPv6. This can result in a device-tracking entry.

But if the target IP is already in the ARP cache or IPv6 neighbour table, no ARP or ND packet is generated when you ping, in which case SISF cannot learn the IP address.

Actions on a packet and the binding table

The distinction between system behaviour in the context of a binding table entry and system behaviour in the context of a packet (from which the binding information is extracted) is an important one, because the available actions are exclusive to each context.

SISF actions determine if any features can access, use, or forward a packet.

- Stop: Means the packet is not available to any client or feature.

  A packet may be stopped while the binding integrity of a possible entry is being verified. From a system perspective, this action is equivalent to dropping a packet. However, from a SISF perspective, the packet might not be considered malicious, as SISF attempts to extract binding information from it.

- Forward: Means the packet is allowed to enter the network and is sent on, unchanged.
- Drop: Means the packet is not allowed to enter the network.

SISF actions on the binding table include only the items listed here.

- Create or update: Means the packet or other source is used to create or update an entry in the binding table.
- Ignore: Means the packet or other source of information is disregarded and there is no change or update in the binding table.

Multiple actions can be performed on a packet. If both the *stop* and *ignore* actions occur, a drop may also happen. If *stop* and *update* actions are observed, the packet is allowed to enter the network and is sent on, unchanged.

Device-tracking

SISF-based device-tracking is disabled by default. You can enable the feature on an interface or VLAN.

When you enable the feature, the binding table is created, followed by subsequent maintenance of the binding table.

The events listed in the Binding table sources, on page 63 section trigger SISF-based device-tracking to monitor the presence, location, and movement of hosts in the network to populate and maintain the binding

table. For example, if information about a host is learned via an ARP or ND packet, each subsequent ARP or ND packet from the same host alerts SISF-based device-tracking to refresh the entry in the binding table, indicating whether the host remains in the same location or has moved.

The switch continuously snoops packets, extracts device identities (MAC and IP addresses), and stores them in the binding table. This process ensures binding integrity and maintains the reachability status of hosts.

Device-tracking policy

A device-tracking policy is a set of rules that SISF-based device-tracking follows. The policy outlines which events to monitor, whether a host is probed, the time to wait before probing the host. These rules are referred to as policy parameters.

Note The policy must be attached to an interface or VLAN. Only then is the binding table for that interface or VLAN populated, in accordance with policy parameters.

To display policy settings, use the **show device-tracking policy** *policy_name* command in privileged EXEC mode.

Understand policy parameters

Policy parameters are keywords for configuring the device-tracking mode. Each parameter enhances network security.

This section explains the purpose of *some* of the important policy parameters so you can configure your policy to better suit your requirements.

For detailed information about parameters in the device-tracking configuration mode, refer to the command reference document for the release.

Glean versus guard versus inspect

SISF extracts the IP and MAC addresses from a packet entering the network and dictates the subsequent action according to the security-level configured in the policy.

You can choose one of these options for the security-level parameter: glean, guard, or inspect. Glean is the least secure, followed by Inspect, and Guard is the most secure option.

To configure this parameter in a policy, enter the **security-level** keyword in the device-tracking configuration mode.

Glean

When the security-level is set to **glean**, SISF extracts the IP and MAC address and enters them into the binding table, without any verification. This option therefore does not ensure binding integrity. It may for example, be suited to a set-up where client applications such as IEEE 802.1X or SANET want to only learn about the host and not rely on SISF for authentication.

The address count limit is the only factor affecting the addition of a binding entry for this security-level. There are separate limits for the maximum number of IPs per port, IPv4 per MAC, and IPv6 per MAC. Entries are rejected once a limit is reached. For more information about this parameter, refer Address count limits, on page 74.

Guard

This is the default value for the security-level parameter.

When the security-level is set to **guard**, SISF extracts and verifies the IP and MAC address of packets entering the network. The outcome of the verification determines whether a binding entry is added, updated, or if the packet is dropped and the client rejected.

Verification begins by searching for a matching entry in the database. The database may be centralised or distributed. If a matching entry is not found, a new entry is added.

If a matching entry is found and the points of attachment (MAC, VLAN, or interface) are found to be the same, only the timestamp is updated. If not, the scope of verification is extended to include validation of address ownership. This may include host polling to determine if the change in the point of attachment (a different MAC, or VLAN) is valid. If the change is valid the entry is updated, or if it is a case of theft, the entry is not added to the binding table.

If a binding entry is added or updated, the corresponding client is granted access to the network. If an entry does not pass verification, the corresponding client is rejected.

Note The verification process affects the binding entry and the corresponding incoming packet.

Inspect

Even though security-level **inspect** is available on the CLI, we recommend not using it. The **glean** and **guard** options described above address most use cases and network requirements.

Security level and SISF action on a packet

The Actions on a packet and the binding table , on page 65 section clarifies the difference between SISF actions on packet and those on the binding table.

The security level policy parameter can affect actions on ND and ARP packets: If the security level is set to **guard** and the packet does not pass verification, a drop action follows. If its glean, the packet is not dropped. Verification failure (with the **guard** security level) can be because of various reasons including invalid binding information, reaching the address count limit, and so on.

Trusted-port and device-role switch

The **device-role switch** and **trusted-port** options help you design an efficient and scalable secure zone. When used together, these two parameters help you achieve an efficient distribution of the creation of entries in the binding table. This keeps the size of the binding tables under control.

The **trusted-port** option: Disables the guard function on configured targets. Bindings learned through a trusted-port have preference over bindings learned through any other port. A trusted port is also given preference in case of a collision while making an entry in the table.

The **device-role** option: Indicates the type of device that is facing the port and this can be a node or a switch. To allow the creation of binding entries for a port, you configure the device as a node. To stop the creation of binding entries, you configure the device as switch.

Configuring the device as a switch is suitable for multi-switch setups, where there is a high possibility of large device-tracking tables. Here, a port facing a device (an uplink trunk port) can be configured to stop creating

binding entries, and the traffic arriving at such a port can be trusted, because the switch on the other side of the trunk port will have device-tracking enabled and that will have checked the validity of the binding entry.

Note In most cases, configure both the **trusted-port** and **device-role switch** options on the port - the examples below explain this in detail. Possible scenarios where only either one of these options is suited or required have also been described, at the end of this section.

To configure these parameters in a policy, use the **trusted-port** and **device-role** keywords in the device-tracking configuration mode.

Trusted-port and device-role switch options in a multi-switch set-up

This example explains how the **device-role switch** and **trusted-port** options help to design an efficient and scalable "secure zone".

In figure *Multi-switch set-ups without trusted-port and device-role switch options* below, SW_A, SW_B, and SW_C are three access switches. They are all connected to a common distribution switch. The only required configuration on the distribution switch in this scenario is to ensure that traffic of any kind is *not* blocked.

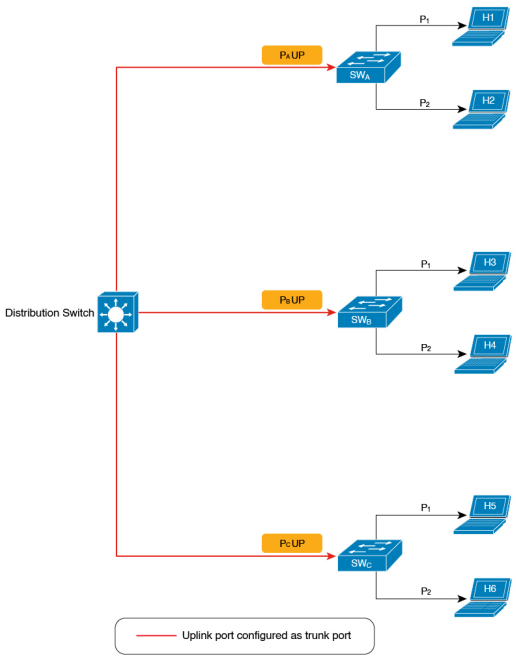
H1, H2, …H6 are the hosts. Each switch has two directly connected hosts. All hosts are communicating with each other, that is, control packets are being transmitted. All hosts are also within the same VLAN boundary. Each switch is receiving control packets from hosts that are directly connected to it, and also from hosts that are connected to other switches. This means SW_A is receiving control packets from H1, H2, …H6 similarly with SW_B and SW_C.

For each switch, the entries of directly connected hosts have interface or port P_1 and P_2 in the binding table. Entries originating from hosts that are connected to other switches have interface or port name P_xUP, to show that they have been learned through the uplink port (x represents the corresponding uplink port for each switch. For example, the entries that SW_A learns through its uplink port have interface or port name P_AUP and for SW_B it is P_BUP, and so forth.

The end result is that each switch learns and creates binding entries for all hosts in the set-up.

This scenario displays an inefficient use of the binding table, because each host is being validated multiple times, which does not make it more secure than if just one switch validates host. Secondly, entries for the same host in multiple binding tables could mean that the address count limit is reached sooner. After the limit is reached, any further entries are rejected and required entries may be missed this way.

Figure 8: Multi-switch set-ups without trusted-port and device-role switch options



Binding table for SW_A

VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
100	192.0.2.1	00:1A:C2:7B:00:47	P_1	REACHABLE	H1
100	192.0.2.2	00:1A:C2:7B:00:47	P_2	REACHABLE	H2
200	192.0.2.3	00:1A:C2:7B:00:48	P_AUP	REACHABLE	H3
200	192.0.2.4	00:1A:C2:7B:00:48	P_AUP	REACHABLE	H4
300	192.0.2.5	00:1A:C2:7B:00:49	P_AUP	REACHABLE	H5
300	192.0.2.6	00:1A:C2:7B:00:49	P_AUP	REACHABLE	H6

Binding table for SW_B

VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
200	192.0.2.3	00:1A:C2:7B:00:48	P_1	REACHABLE	H3
200	192.0.2.4	00:1A:C2:7B:00:48	P_2	REACHABLE	H4
100	192.0.2.1	00:1A:C2:7B:00:47	P_BUP	REACHABLE	H1
100	192.0.2.2	00:1A:C2:7B:00:47	P_BUP	REACHABLE	H2
300	192.0.2.5	00:1A:C2:7B:00:49	P_BUP	REACHABLE	H5
300	192.0.2.6	00:1A:C2:7B:00:49	P_BUP	REACHABLE	H6

Binding table for SW_C

VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
300	192.0.2.5	00:1A:C2:7B:00:49	P_1	REACHABLE	H5
300	192.0.2.6	00:1A:C2:7B:00:49	P_2	REACHABLE	H6
100	192.0.2.1	00:1A:C2:7B:00:47	P_CUP	REACHABLE	H1
100	192.0.2.2	00:1A:C2:7B:00:47	P_CUP	REACHABLE	H2
200	192.0.2.3	00:1A:C2:7B:00:48	P_CUP	REACHABLE	H3
200	192.0.2.4	00:1A:C2:7B:00:48	P_CUP	REACHABLE	H4

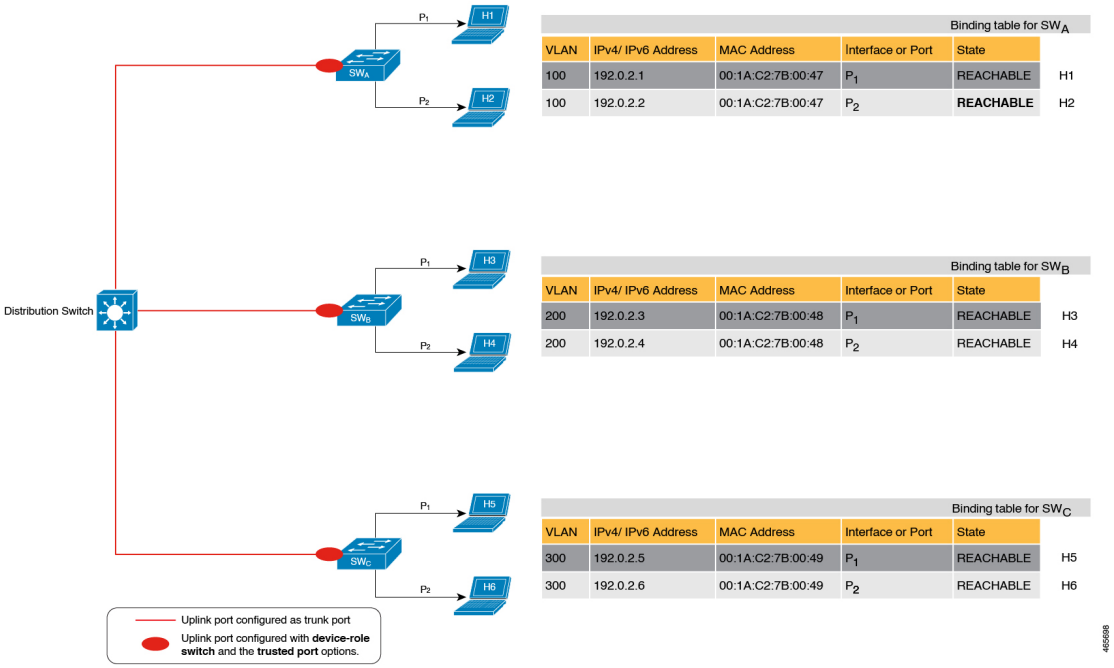
455697

By contrast, see figure *Multi-switch set-ups with trusted-port and device-role switch options* below. Here, when SW_A intercepts the packet of a host that is not attached to it (say H3 which is directly attached to SW_B), it does not create an entry because it detects that H3 is attached to a device that is configured as a switch (**device-role switch** option) and the uplink port of the switch (where the packet came from) is a trusted port (**trusted-port** option).

By creating binding entries only on switches where the host appears on an access port (port P_1 and P_2 of each switch), and not creating entries for a host that appears over an uplink port or trusted port (P_x UP), each switch in the set-up validates and makes only the required entries, thus achieving an efficient distribution of the creation of binding table entries.

A second advantage of configuring **device-role switch** and **trusted-port** options in a multi-switch scenario is the prevention of duplicate entries when a host, such as H1, moves from one switch to another H1's IP and MAC binding in the earlier location (let's say SW_A) continues to remain there until it reaches the STALE state. But if H1 moves and connects to a second switch, say SW_C, then SW_A receives a duplicate binding entry through the uplink port. In such a situation, if the uplink port of the second switch (SW_C) is configured as a trusted port, SW_A deletes its stale entry. Further, it doesn't create another new binding entry because the SW_C will already have the latest entry and this entry is trusted.

Figure 9: Multi-switch set-ups with trusted-port and device-role switch options



Binding table for SW_A

VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
100	192.0.2.1	00:1A:C2:7B:00:47	P_1	REACHABLE	H1
100	192.0.2.2	00:1A:C2:7B:00:47	P_2	**REACHABLE**	H2

Binding table for SW_B

VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
200	192.0.2.3	00:1A:C2:7B:00:48	P_1	REACHABLE	H3
200	192.0.2.4	00:1A:C2:7B:00:48	P_2	REACHABLE	H4

Binding table for SW_C

VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
300	192.0.2.5	00:1A:C2:7B:00:49	P_1	REACHABLE	H5
300	192.0.2.6	00:1A:C2:7B:00:49	P_2	REACHABLE	H6

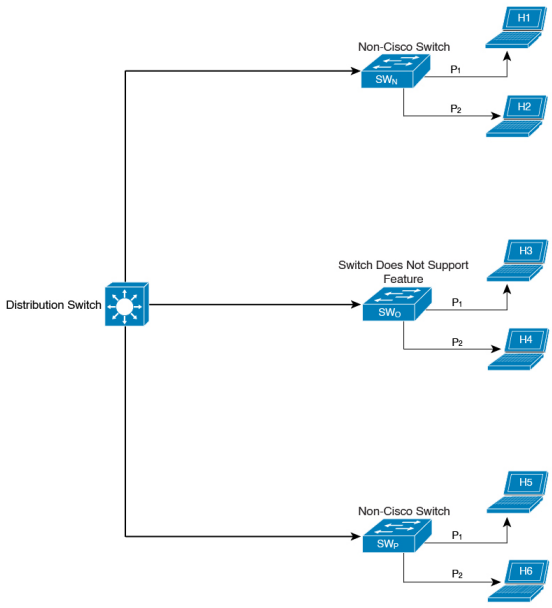
When not to use trusted-port and device-role switch options

While the previous example clarifies how a multi-switch set-up with distributed binding tables stands to benefit from the **device-role switch** and **trusted-port** options, it may not suit networks of the following kinds:

- Networks where non-Cisco switches are being used
- Networks where the switch does not support the SISF-based device-tracking feature.

In both cases, we recommended that you do not configure the **device-role switch** and **trusted-port** options. Further, we recommended that you maintain a centralized binding table - on the distribution switch. When you do, all the binding entries for all the hosts connected to non-Cisco switches and switches that do not support the feature, are validated by the distribution switch and still secure your network. The figure below illustrates the same.

Figure 10: Centralised binding table



Binding table for Distribution Switch

VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
100	192.0.2.1	00:1A:C2:7B:00:47	P_NUP	REACHABLE	H1
100	192.0.2.2	00:1A:C2:7B:00:47	P_NUP	REACHABLE	H2
200	192.0.2.3	00:1A:C2:7B:00:48	P_OUP	REACHABLE	H3
200	192.0.2.4	00:1A:C2:7B:00:48	P_OUP	REACHABLE	H4
300	192.0.2.5	00:1A:C2:7B:00:49	P_PUP	REACHABLE	H5
300	192.0.2.6	00:1A:C2:7B:00:49	P_PUP	REACHABLE	H6

When to use only trusted-port or device-role switch option

Configure only the **device-role** switch when you want to listen but not learn entries. For example, for Duplicate Address Detection (DAD) or when you want to send IPv6 or Neighbor Solicitation (NS) messages on a switch-facing port.

When you configure this option on a switch port (or interface), SISF-based device-tracking treats the port as a trunk port, implying that the port is connected to other switches. It is irrelevant whether the port is a trunk port. Therefore, when NS packets or queries are sent to switches in the network for new entry validation, only the secure ports (ports where the **device-role switch** is configured) receive the packet or query. This configuration safeguards the network. If the command is not configured on any port, a general broadcast of the query is sent.

Configure only the **trusted-port** for situations where an access port should be a trusted port. If an access port is connected to a DHCP server or a similar service that the switch is consuming, configuring an access port as a trusted port ensures that the service is not disrupted because traffic from such a port is trusted. This setup also widens the secure zone to include the access port.

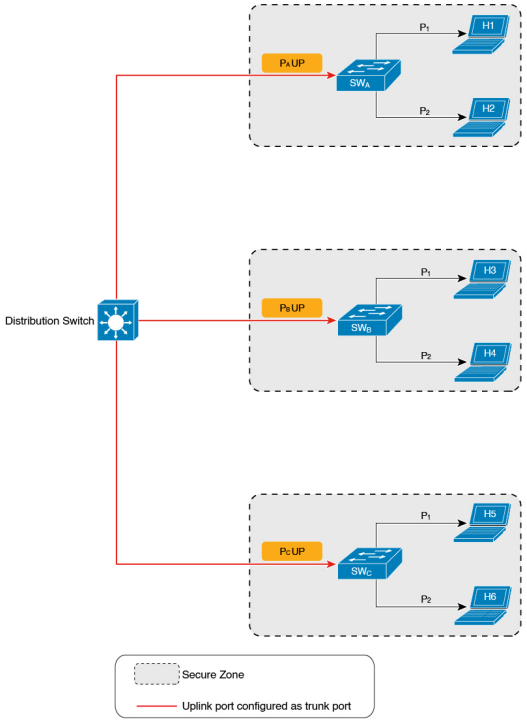
Efficient and scalable secure zone

Use the **trusted-port** and **device-role switch** options in appropriate networks to create an efficient and scalable secure zone.

Secure zones 1, 2, and 3 represent three setups, each demonstrating how the secure zone is established.

Secure zone:	Secure zone 1 - Inefficient and unscalable secure zone	Secure zone 2 - Efficient and scalable secure zone when binding tables are decentralized	Secure zone 3: Efficient secure zone when binding table is centralized
Scalability:	Unscalable; each switch has entries of all the hosts in the network	Scalable; each switch as entries of only directly connected hosts	Unscalable; the distribution switch has entries of all hosts in the network
Polling and its effect on the network: n = number of hosts m = number of switches total number of polling requests: = n X m	18 polling requests are being sent (6 hosts x 3 switches). Each host is polled by all the switches in the network (in the absence of the **trusted-port** and **device-role switch** options). Network load is very high.	6 polling requests are being sent (2 hosts x 1 switch for *each* switch). Minimal network load. (Polling requests are sent by the local access switches to directly connected hosts, each polling request passes through fewer points in the network.)	6 polling requests are being sent (6 hosts x 1 switch) Network load is higher than secure zone 2, but not as high as secure zone 1. (Polling requests come from the distribution switch and go through the access switch before reaching the host.)
Efficiency:	The binding table is inefficient because it is duplicated on each switch.	Efficient binding table, because each host's binding information is entered only once, and in one binding table and this the binding table of the directly connected switch.	Efficient binding table, because the binding information for each host is entered only once, and this is in the central binding table, which is on the distribution switch.
Recommended action:	Reapply suitable policies to make the secure zone like secure zone 2	None; this is an efficient and scalable secure zone.	None; this is the best possible secure zone given the type of set-up (where the other switches in the network are either non-Cisco or do not support the feature)

Figure 11: Secure zone 1 - Inefficient and unscalable secure zone



Binding table for SW_A

VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
100	192.0.2.1	00:1A:C2:7B:00:47	P_1	REACHABLE	H1
100	192.0.2.2	00:1A:C2:7B:00:47	P_2	REACHABLE	H2
200	192.0.2.3	00:1A:C2:7B:00:48	P_AUP	REACHABLE	H3
200	192.0.2.4	00:1A:C2:7B:00:48	P_AUP	REACHABLE	H4
300	192.0.2.5	00:1A:C2:7B:00:49	P_AUP	REACHABLE	H5
300	192.0.2.6	00:1A:C2:7B:00:49	P_AUP	REACHABLE	H6

Binding table for SW_B

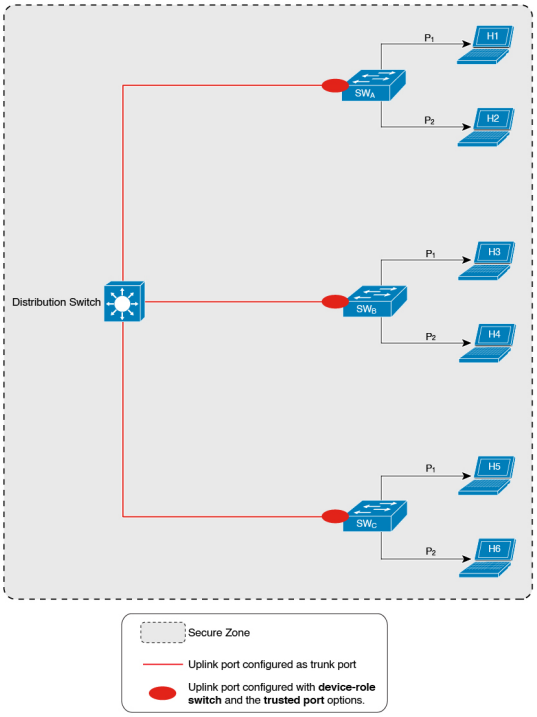
VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
200	192.0.2.3	00:1A:C2:7B:00:48	P_1	REACHABLE	H3
200	192.0.2.4	00:1A:C2:7B:00:48	P_2	REACHABLE	H4
100	192.0.2.1	00:1A:C2:7B:00:47	P_BUP	REACHABLE	H1
100	192.0.2.2	00:1A:C2:7B:00:47	P_BUP	REACHABLE	H2
300	192.0.2.5	00:1A:C2:7B:00:49	P_BUP	REACHABLE	H5
300	192.0.2.6	00:1A:C2:7B:00:49	P_BUP	REACHABLE	H6

Binding table for SW_C

VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
300	192.0.2.5	00:1A:C2:7B:00:49	P_1	REACHABLE	H5
300	192.0.2.6	00:1A:C2:7B:00:49	P_2	REACHABLE	H6
100	192.0.2.1	00:1A:C2:7B:00:47	P_CUP	REACHABLE	H1
100	192.0.2.2	00:1A:C2:7B:00:47	P_CUP	REACHABLE	H2
200	192.0.2.3	00:1A:C2:7B:00:48	P_CUP	REACHABLE	H3
200	192.0.2.4	00:1A:C2:7B:00:48	P_CUP	REACHABLE	H4

465700

Figure 12: Secure zone 2 - Efficient and scalable secure zone when binding tables are decentralized



Binding table for SW_A

VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
100	192.0.2.1	00:1A:C2:7B:00:47	P_1	REACHABLE	H1
100	192.0.2.2	00:1A:C2:7B:00:47	P_2	**REACHABLE**	H2

Binding table for SW_B

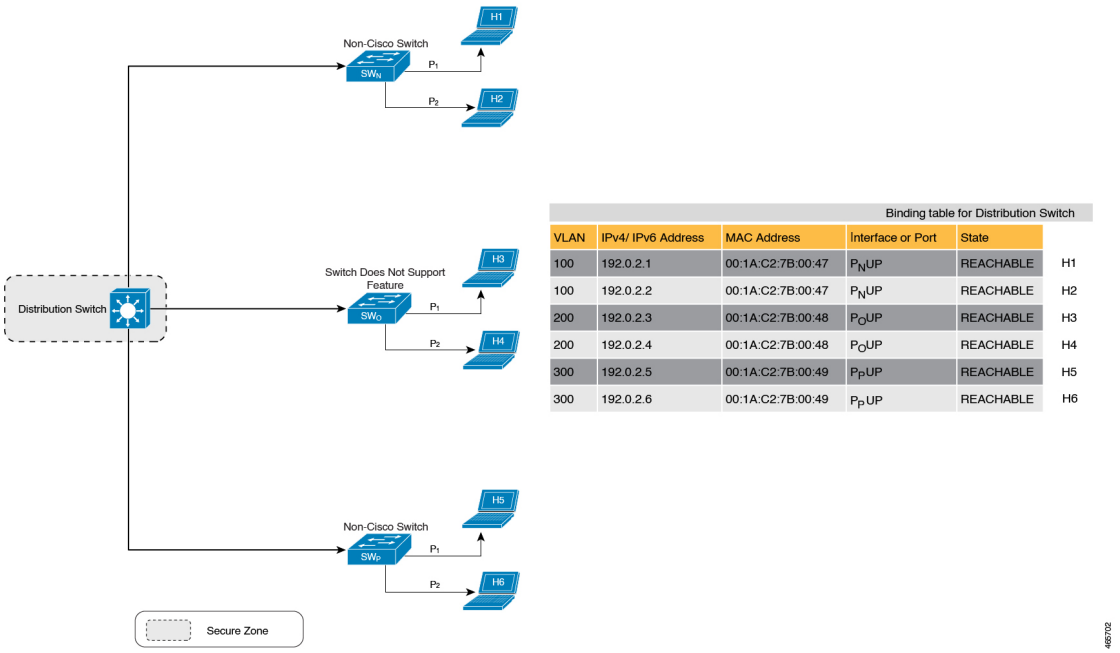
VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
200	192.0.2.3	00:1A:C2:7B:00:48	P_1	REACHABLE	H3
200	192.0.2.4	00:1A:C2:7B:00:48	P_2	REACHABLE	H4

Binding table for SW_C

VLAN	IPv4/ IPv6 Address	MAC Address	Interface or Port	State	
300	192.0.2.5	00:1A:C2:7B:00:49	P_1	REACHABLE	H5
300	192.0.2.6	00:1A:C2:7B:00:49	P_2	REACHABLE	H6

465701

Figure 13: Secure zone 3: Efficient secure zone when binding table is centralized



Address count limits

The address count limit parameter specifies how many IP and MAC addresses can be entered in a binding table. These limits are determined based on known and expected hosts. This enables preemptive action against rogue hosts or IPs.

At a policy level, there are separate limits for IP addresses per port, IPv4 addresses per MAC, and IPv6 addresses per MAC. Configure or change only the number of IP addresses per port.

IP per port

The IP per port option is the total number of IP addresses allowed for a port. The address can be IPv4 or IPv6. When the limit is reached, no further IP addresses (i.e., entries) are added to the binding table.

To configure this parameter in a policy, enter the **limit address-count** *ip-per-port* keyword in device-tracking configuration mode. If you configure a lower limit, it applies only to new entries. An existing entry remains in the binding table and goes through its binding entry lifecycle.

IPv4 per MAC and IPv6 per MAC

This refers to the number of IPv4 addresses that can be mapped to one MAC address and the number of IPv6 addresses that can be mapped to one MAC address. When the limit is reached, no further entries can be added to the binding table, and traffic from new hosts will be dropped

Note The IPv4 per MAC limit and the IPv6 per MAC limit that is effective on an interface or VLAN is as defined in the policy that is applied. If the policy does not specify a limit, this means that a limit does not exist. Limit configuration for IPv4 per MAC or IPv6 per MAC is not possible for programmatic, custom, or default policies.

Enter the **show device-tracking policy** *policy name* command to check if a limit exisits.

Here is an example output of a policy where an IPv4 per MAC and an IPv6 per MAC limit exists:

```
Device# show device-tracking policy LISP-DT-GUARD-VLAN

Policy LISP-DT-GUARD-VLAN configuration:
  security-level guard (*)
  <output truncated>

  limit address-count for IPv4 per mac 4 (*)
  limit address-count for IPv6 per mac 12 (*)
  tracking enable

<output truncated>
```

Address count limit and SISF action on the binding table

The Actions on a packet and the binding table , on page 65 section clarifies the difference between SISF actions on packet and those on the binding table.

The address count limit parameter affects actions that are performed on the binding table: If a limit is reached, binding entries are not added to the binding table. (It does not influence packet action).

Address count limit and interactions with other SISF settings

- The limits do not have a hierarchy, but the threshold that is set for each limit affects the others.

  For example, if the IP per port limit is 100, and the IPv4 per MAC limit is one, the limit is reached with a single host's IPv4-MAC binding entry. No further IP entries, which are bound to the same MAC are allowed in the table even though the port has a provision for 99 more IP addresses. Similarly, if the IP per port limit is one, and the IPv4 per MAC limit is 100. The limit is reached with a single host's IPv4-MAC binding entry. No further IP entries are allowed in the table even though the MAC has a provision for 99 more IP addresses for *that* MAC.

- Global and policy-level limits

  The limits configured with the **device-tracking binding max-entries** command are at the global level, the limits configured with the **limit address-count** command in the device-tracking configuration mode are for a policy, which is at the interface or VLAN level.

  If a policy-level value *and* a globally configured value exists, the creation of binding entries is stopped when *a* limit is reached - this limit can be any one of the global values or the policy-level value.

  If only globally configured values exist, the creation of binding entries is stopped when *a* limit is reached.

  If the only policy-level value exists, the creation of binding entries is stopped when the policy-level limit is reached.

Tracking

The tracking parameter involves monitoring hosts in the network. In section States and lifetime of a binding table entry, on page 61 above, this is referred to as "polling" which is described in detail.

To configure polling parameters at the global level, enter the **device-tracking tracking** command in global configuration mode. After configuring this command, you can choose to turn polling on or off for individual interfaces and VLANs. For this, you must enable or disable polling in the policy.

To enable polling in a policy, enter the **tracking enable** command in the device-tracking configuration mode. By default, polling is disabled in a policy.

Guidelines to create a policy

- If you have multiple policies available on a given target, the system uses an internal policy priority to determine which policy takes precedence.

  A manually created policy has the highest priority. Therefore, a custom policy can be created to override the settings and take precedence over a programmatically created policy.

- The parameters of a programmatically created policy cannot be changed. You can configure certain attributes of a custom policy.

Guidelines to apply a policy

- You can attach multiple policies to the same VLAN.
- If a programmatic policy is attached to a VLAN and you wish to change the policy settings, create a custom device-tracking policy and attach it to the VLAN.
- If you attach multiple policies with different priorities to the same VLAN, the policy with the highest priority takes effect. Exceptions include the limit address-count settings for IPv4 per MAC and IPv6 per MAC, where the settings of the policy with the lowest priority are effective.
- When you attach a device-tracking policy to an interface under a VLAN, the interface's policy settings prevail over those of the VLAN. However, limit address-count values for IPv4 per MAC and IPv6 per MAC are aggregated from both policies.
- You must remove the device tracking client feature configuration before you can remove a policy.

Configure SISF

By default, SISF or SISF-based device-tracking is disabled. You enable it by defining a device-tracking policy and attaching the policy to a specific target. The target could be an interface or VLAN. Use the method that best suits your needs.

Method to enable SISF	Applicable configuration tasks	Result
Option 1: Manually, by using interface configuration commands to create and apply the **default** policy to a target.	Apply the default device tracking policy to a target, on page 79	Automatically applies the **default** device tracking policy to the specified target. The **default** policy is a built-in policy with default settings; you cannot change any of the attributes of the default policy. See **Option 2** if you want to configure device tracking policy attributes.
Option 2: Manually, by using global configuration commands to create a custom policy and applying the custom policy to a target.	1. Create a custom device tracking policy with custom settings, on page 80 2. Attach the custom policy to an interface or VLAN: Attach a device tracking policy to an interface, on page 83 OR Attach a device tracking policy to a VLAN, on page 84	Creates a custom policy with the name and policy parameters you configure, and attaches the policy to the specified target.
Option 3: Programmatically, by configuring the snooping command.	Enter the **ip dhcp snooping vlan** *vlan* command in global configuration mode. Example: Programatically enable SISF by configuring DHCP Snooping, on page 87	When you configure the command, the system automatically creates policy `DT-PROGRAMMATIC`. Use this method if you want to enable SISF-based device tracking for these clients: IEEE 802.1X, Web authentication, Cisco TrustSec, IP Source Guard, and SANET.
Option 4: Programmatically, by configuring Locator ID Separation Protocol (LISP).	Example: Programatically enable SISF by configuring LISP, on page 88	When you configure LISP, the system automatically creates policy `LISP-DT-GUARD-VLAN`, `LISP-DT-GLEAN-VLAN`, or other variants, depending on the software release.
Option 5: Programmatically, by configuring EVPN VLAN.	Example: Programatically enable SISF by configuring EVPN on VLAN, on page 87	When you configure EVPN on VLAN, the system automatically creates policy `evpn-sisf-policy`, `evpn-device-track`, or other variants, depending on the software release.

Method to enable SISF	Applicable configuration tasks	Result
Option 6: By using an interface template	Enable SISF using an interface template, on page 85	By adding the policy to an interface template, you can apply the same policy to multiple targets, without having to create it separately for each target.
Option 7: Migrating from legacy IPDT and IPv6 Snooping.	Migrate from legacy IPDT and IPv6 Snooping to SISF-based device tracking, on page 78	Convert legacy IPDT and IPv6 Snooping configuration to the SISF-based device-tracking commands.

Migrate from legacy IPDT and IPv6 Snooping to SISF-based device tracking

The **device-tracking upgrade-cli** global configuration command upgrades the CLI differently based on the legacy configuration that exists on your device. Consider these configuration scenarios and migration results before updating your existing configuration:

Note Avoid configuring both the old IPDT and IPv6 Snooping commands with the SISF-based device-tracking commands.

Only IPDT configuration exists

If your device has only IPDT configuration, running the **device-tracking upgrade-cli** command creates and attaches a SISF policy to the interface.

Continuing with legacy commands restricts operations to legacy modes, allowing only legacy IPDT and IPv6 Snooping commands to work.

Only IPv6 snooping configuration exists

On a device with existing IPv6 Snooping configuration, you can configure further using the old IPv6 Snooping commands. Here are the options you can choose from:

- (Recommended) Use the **device-tracking upgrade-cli** command to convert all your legacy configuration to the SISF-based device-tracking commands. After conversion, your device will support only SISF-based device-tracking commands.
- Use the legacy IPv6 Snooping commands for your future configuration and do not run the **device-tracking upgrade-cli** command. With this option, only the legacy IPv6 Snooping commands are available on your device, and you cannot use the SISF-based device-tracking commands.

Both IPDT and IPv6 snooping configuration exist

On a device that has both legacy IPDT configuration and IPv6 Snooping configuration, you can convert legacy commands to the SISF-based device-tracking commands. However, note that only one snooping policy can be attached to an interface, and the IPv6 Snooping policy parameters override the IPDT settings.

Note If you do not migrate to the SISF-based device-tracking commands and continue to use the legacy IPv6 Snooping or IPDT commands, your IPv4 device-tracking configuration information may be displayed in the IPv6 Snooping commands, as the SISF-based device-tracking feature handles both IPv4 and IPv6 configuration. To avoid this, we recommend that you convert your legacy configuration to SISF-based device-tracking commands.

No IPDT or IPv6 snooping configuration exists

If your device has no legacy IP Device Tracking or IPv6 Snooping configurations, you can use only the SISF-based device-tracking commands for all your future configuration. The legacy IPDT commands and IPv6 Snooping commands are not available.

Apply the default device tracking policy to a target

Perform these steps to apply the default device tracking policy to an interface or 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	Specify an interface or a VLAN
	• **interface** *type number*
	• **vlan configuration** *vlan_list*
	Example:
	`Device(config)# interface gigabitethernet 1/1/4`
	`OR`
	`Device(config)# vlan configuration 333`
		interface *type number*: Specifies the interface and enters interface configuration mode. Attach the device tracking policy to the specified interface.
		vlan configuration *vlan_list*: Specifies the VLANs and enters VLAN feature configuration mode. The device tracking policy will be attached to the specified VLAN.

Step 4	**device-tracking**
	Example:
	`Device(config-if)# device-tracking`
	`OR`
	`Device(config-vlan-config)# device-tracking`
		Enables SISF-based device tracking and attaches the default policy to the interface or VLAN.
		The default policy is built-in with settings that cannot be changed.

Step 5	**end** **Example:**	Exits interface or VLAN feature configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose

	`Device(config-if)# end`
	OR
	`Device(config-vlan-config)# end`	

Step 6	**show device-tracking policy** *policy-name* **Example:** `Device# show device-tracking policy default`	Displays device-tracking policy configuration, and all the targets it is applied to.

Create a custom device tracking policy with custom settings

In privileged EXEC mode, perform these steps to create and configure a device tracking policy:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	[**no**] **device-tracking policy** *policy-name* **Example:** `Device(config)# device-tracking policy example_policy`	Creates the policy and enters device-tracking configuration mode.

Step 3	[**data-glean** | **default** | **destination-glean** | **device-role** | **distribution-switch** | **exit** | **limit** | **no** | **prefix-glean** | **protocol** | **security-level** | **tracking** | **trusted-port** | **vpc**]
	Example:
	`Device(config-device-tracking)# destination-glean log-only`
		Enter the question mark (?) at the system prompt to obtain a list of available options in this mode. You can configure the following for both IPv4 and IPv6:
		• (Optional) **data-glean**: Enables learning of addresses from a data packet snooped from a source inside the network and populates the binding table with the data traffic source address. Enter one of these options:
		  • **log-only**: Generates a syslog message upon data packet notification
		  • **recovery**: Uses a protocol to enable binding table recovery. Enter **NDP** or **DHCP**.
		• (Optional) **default**: Sets the policy attribute to its default value. You can set these policy attributes to their default values: **data-glean**, **destination-glean**,

	Command or Action	Purpose
		device-role, **limit**, **prefix-glean**, **protocol**, **security-level**, **tracking**, **trusted-port**. • (Optional) **destination-glean**: Populates the binding table by gleaning data traffic destination address. Enter one of these options: • **log-only**: Generates a syslog message upon data packet notification • **recovery**: Uses a protocol to enable binding table recovery. Enter **DHCP**. • (Optional) **device-role**: Sets the role of the device attached to the port. It can be a node or a switch. Enter one of these options: • **node**: Configures the attached device as a node. This is the default option. • **switch**: Configures the attached device as a switch. • (Optional) **distribution-switch**: Although visible on the CLI, this option is not supported. Any configuration settings you make will not take effect. • **exit**: Exits the device-tracking policy configuration mode. • **limit address-count**: Specifies an address count limit per port. The range is 1 to 32000. • **no**: Negates the command or sets it to defaults. • (Optional) **prefix-glean**: Enables learning of prefixes from either IPv6 Router Advertisements or from DHCP-PD. You have the following option: • (Optional) **only**: Gleans only prefixes and not host addresses. • (Optional) **protocol**: Sets the protocol to glean; by default, all are gleaned. Enter one of these options: • **arp** [**prefix-list** *name*]: Gleans addresses in ARP packets. Optionally,

	Command or Action	Purpose
		enter the name of prefix-list that is to be matched. • **dhcp4** [**prefix-list** *name*]: Glean addresses in DHCPv4 packets. Optionally, enter the name of prefix-list that is to be matched. • **dhcp6** [**prefix-list** *name*]: Glean addresses in DHCPv6 packets. Optionally, enter the name of prefix-list that is to be matched. • **ndp** [**prefix-list** *name*]: Glean addresses in NDP packets. Optionally, enter the name of prefix-list that is to be matched. • (Optional) **security-level**: Specifies the level of security enforced by the feature. Enter one of these options: • **glean**: Gleans addresses passively. • **guard**: Inspects and drops un-authorized messages. This is the default. • **inspect**: Gleans and validates messages. • (Optional) **tracking**: Specfies a tracking option. Enter one of these options: • **disable** [**stale-lifetime** [*1-86400-seconds* | **infinite**]]: Turns of device-tracking. Optionally, you can enter the duration for which the entry is kept inactive before deletion, or keep it permanently inactive. • **enable** [**reachable-lifetime** [*1-86400-seconds* | **infinite**]]: Turns on device-tracking. Optionally, you can enter the duration for which the entry is kept reachable, or keep it permanently reachable. • (Optional) **trusted-port**: Sets up a trusted port. Disables the guard on applicable targets. Bindings learned through a trusted

	Command or Action	Purpose
		port have preference over bindings learned through any other port. A trusted port is given preference in case of a collision while making an entry in the table. • (Optional) **vpc**: Although visible on the CLI, this option is not supported. Any configuration settings you make will not take effect.
Step 4	**end** **Example:** `Device(config-device-tracking)# end`	Exits device-tracking configuration mode and returns to privileged EXEC mode.
Step 5	**show device-tracking policy** *policy-name* **Example:** `Device# show device-tracking policy example_policy`	Displays the device-tracking policy configuration.

What to do next

Attach the policy to an interface or VLAN.

Attach a device tracking policy to an interface

Perform these steps to attach a device tracking policy 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# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface* **Example:** `Device(config-if)# interface gigabitethernet 1/1/4`	Specifies an interface and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**device-tracking attach-policy** *policy name* **Example:** `Device(config-if)# device-tracking attach-policy example_policy`	Attaches the device tracking policy to the interface. Device tracking is also supported on EtherChannels. **Note** SISF based device-tracking policies can be disabled only if they are custom policies. Programmatically created policies can be removed only if the corresponding device-tracking client feature configuration is removed.
Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 6	**show device-tracking policies** [**interface** *interface*] **Example:** `Device# show device-tracking policies interface gigabitethernet 1/1/4`	Displays policies that match the specified interface type and number.

Attach a device tracking policy to a VLAN

Perform these steps to attach a device-tracking policy to VLANs across multiple 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# configure terminal`	Enters global configuration mode.
Step 3	**vlan configuration** *vlan_list* **Example:** `Device(config)# vlan configuration 333`	Specifies the VLANs to which the device tracking policy will be attached, and enters VLAN interface configuration mode.
Step 4	**device-tracking attach-policy** *policy_name* **Example:** `Device(config-vlan-config)# device-tracking attach-policy`	Attaches the device tracking policy to the specified VLANs across all switch interfaces. **Note**

	Command or Action	Purpose
	`example_policy`	Disable SISF-based device-tracking policies only if they are custom. Remove programmatically created policies only if the corresponding device-tracking client feature configuration is removed.
Step 5	**do show device-tracking policies vlan** *vlan-ID* **Example:** `Device(config-vlan-config)# do show device-tracking policies vlan 333`	Verifies that the policy is attached to the specified VLAN, without exiting the VLAN interface configuration mode.
Step 6	**end** **Example:** `Device(config-vlan-config)# end`	Exits VLAN feature configuration mode and returns to privileged EXEC mode.

Enable SISF using an interface template

An interface template is a container that holds configurations or policies. When you apply the interface template to a target, all the configurations are applied to the target. This enables you to configure multiple commands or features on one or more targets.

You can add the **device-tracking policy** *policy_name* global configuration command to an interface template. SISF-based device-tracking is enabled and the policy is applied, wherever the template is applied.

You can also apply the template through 802.1x authentication. Different templates (and therefore different policies) can be dynamically assigned to different interfaces during the 802.1x authentication process.

Note You can apply only one interface template to one port.

Before you begin

You have already created a custom policy. See Create a custom device tracking policy with custom settings, on page 80.

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:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**template interface** *template_name* **Example:** `Device(config)# template interface template_w_sisf`	Creates a template with the name you specify and enters template configuration mode. In the accompanying example, a template called *template_w_sisf* is created.
Step 4	**device-tracking attach-policy** *policy_name* **Example:** `Device (config-template)# device-tracking attach-policy sisf_policy_for_template`	Attaches a policy to the template. SISF-based device-tracking is enabled and the policy is applied wherever the template is applied.
Step 5	**exit** **Example:** `Device (config-template)# exit`	Exits the template configuration mode and enters the global configuration mode.
Step 6	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 1/1/4`	Specifies an interface and enters interface configuration mode.
Step 7	**source template** *template_name* **Example:** `Device(config-if)# source template template_w_sisf`	Configures a static binding for an interface template. In the accompaying example, *template_w_sisf* is statically applied to an interface.
Step 8	**end** **Example:** `Device(config-if)# end`	Exits the interface configuration mode and enters the privileged EXEC mode.

Step 9	**show running-config interface** *type number*
	Example:
	`Device# show running-config interface gigabitethernet 1/1/4`
	
	`Building configuration...`
	`<output truncated>`
	
	`Current configuration : 71 bytes`
	`!`
	`interface GigabitEthernet1/1/14`
	`source template template_w_sisf`
	`end`
	
	`<output truncated>`
		Displays the contents of the running configuration.

Configuration examples

Refer this section for configuration examples of SISF.

Example: Programatically enable SISF by configuring DHCP Snooping

The following example shows how to configure the **ip dhcp snooping vlan** *vlan* command in global configuration mode to enable SISF-based device-tracking. Enabling SISF this way creates the `DT-PROGRMMATIC` policy on the system.

Enter the **show device-tracking policy** *policy_name* command in privileged EXEC mode, to display the settings for a `DT-PROGRMMATIC` policy.

Tip This is only sample output displaying the settings of a programmatic policy and may change from one release to another. Always use the **show** command to view the settings of a policy relevant to your software version.

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp snooping vlan 10
Device(config)# end

Device# show device-tracking policy DT-PROGRAMMATIC

Policy DT-PROGRAMMATIC configuration:
  security-level glean (*)
  device-role node
  gleaning from Neighbor Discovery
  gleaning from DHCP
  gleaning from ARP
  gleaning from DHCP4
  NOT gleaning from protocol unkn
  limit address-count for IPv4 per mac 1 (*)
  tracking enable
Policy DT-PROGRAMMATIC is applied on the following targets:
Target      Type    Policy               Feature            Target range
vlan 10     VLAN    DT-PROGRAMMATIC      Device-tracking    vlan all

  note:
  Binding entry Down timer: 24 hours (*)
  Binding entry Stale timer:   24 hours (*)
```

Example: Programatically enable SISF by configuring EVPN on VLAN

When you configure EVPN, the system automatically creates programmatic policy `evpn-device-track`. Enter the **show device-tracking policy** *policy_name* command in privileged EXEC mode, to display policy settings.

Tip This is only sample output displaying the settings of a programmatic policy and may change from one release to another. Always use the show command, to see the settings of a policy as applicable to the software version running on your device.

```
Device# show device-tracking policy evpn-device-track

Device-tracking policy evpn-device-track configuration:
  security-level glean
  device-role node
  gleaning from Neighbor Discovery
  gleaning from DHCP6
  gleaning from ARP
  gleaning from DHCP4
  NOT gleaning from protocol unkn
  limit address-count for IPv4 per mac 4 preference high
  limit address-count for IPv6 per mac 12 preference  high
```

Example: Programatically enable SISF by configuring LISP

Note The system creates `LISP-DT-GUARD-VLAN`, or `LISP-DT-GLEAN-VLAN`, or `LISP-DT-GUARD-VLAN-MULTI-IP` depending on *how* LISP is configured. You cannot change this, but if required you can create a custom policy with custom settings and attach it to the required target.

Tip This is only sample output displaying the settings of a programmatic policy and may change from one release to another. Always use the **show** command to see the settings of a policy applicable to the software version running on your device.

This is sample output of programmatic policy `LISP-DT-GLEAN-VLAN`. To display policy settings, enter the **show device-tracking policy** *policy_name* command in privileged EXEC mode.

```
Device# show device-tracking policy LISP-DT-GLEAN-VLAN

Policy LISP-DT-GLEAN-VLAN configuration:
  security-level glean (*)
  device-role node
  gleaning from Neighbor Discovery
  gleaning from DHCP
  gleaning from ARP
  gleaning from DHCP4
  NOT gleaning from protocol unkn
  limit address-count for IPv4 per mac 4 (*)
  limit address-count for IPv6 per mac 12 (*)
  tracking enable
Policy LISP-DT-GUARD-VLAN is applied on the following targets:
Target     Type   Policy                 Feature           Target range
vlan 10    VLAN   LISP-DT-GLEAN-VLAN     Device-tracking   vlan all

  note:
  Binding entry Down timer: 10 minutes (*)
  Binding entry Stale timer:   30 minutes (*)
```

This is a sample output of programmatic policy `LISP-DT-GUARD-VLAN`. To display policy settings, enter the **show device-tracking policy** *policy_name* command in privileged EXEC mode.

```
Device# show device-tracking policy LISP-DT-GUARD-VLAN

Policy LISP-DT-GUARD-VLAN configuration:
  security-level guard (*)
```

```
  device-role node
  gleaning from Neighbor Discovery
  gleaning from DHCP
  gleaning from ARP
  gleaning from DHCP4
  NOT gleaning from protocol unkn
  limit address-count for IPv4 per mac 4 (*)
  limit address-count for IPv6 per mac 12 (*)
  tracking enable
Policy LISP-DT-GUARD-VLAN is applied on the following targets:
Target      Type    Policy                    Feature            Target range
vlan 10     VLAN    LISP-DT-GUARD-VLAN        Device-tracking    vlan all
  note:
  Binding entry Down timer: 10 minutes (*)
  Binding entry Stale timer:   30 minutes (*)
```

This is a sample output of programmatic policy `LISP-DT-GUARD-VLAN-MULTI-IP`. To display policy settings, enter the **show device-tracking policy** *policy_name* command in privileged EXEC mode.

```
Device# show device-tracking policy LISP-DT-GUARD-VLAN-MULTI-IP

Device-tracking policy LISP-DT-GUARD-VLAN-MULTI-IP configuration:
  security-level guard
  device-role node
  gleaning from Neighbor Discovery
  gleaning from DHCP6
  gleaning from ARP
  gleaning from DHCP4
  NOT gleaning from protocol unkn
  limit address-count for IPv4 per mac 21 preference high
  limit address-count for IPv6 per mac 84
  origin fabric
  tracking enable reachable-lifetime 240
```

Example: Mitigate the IPv4 duplicate address problem

This example shows how you can tackle the `Duplicate IP Address 0.0.0.0 error message` problem encountered by clients that run Microsoft Windows:

Configure the **device-tracking tracking auto-source** command in global configuration mode. This command determines the source IP and MAC address used in the ARP probe sent by the switch to probe a client, in order to maintain its entry in the device-tracking table. The purpose, is to avoid using 0.0.0.0 as source IP address.

Note Configure the **device-tracking tracking auto-source** command when a switch virtual interface (SVI) is not configured. You do not have to configure it when a SVI is configured with an IPv4 address on the VLAN.

Command	Action (In order to select source IP and MAC address for device tracking ARP probe)	Notes
device-tracking tracking auto-source global configuration command.	• Set source to VLAN SVI if present. • Look for IP and MAC binding in device-tracking table from same subnet. • Use 0.0.0.0	We recommend that you disable device-tracking on all trunk ports to avoid MAC flapping.
device-tracking tracking auto-source override global configuration command.	• Set source to VLAN SVI if present. • Use 0.0.0.0	Not recommended when there is no SVI.
device-tracking tracking auto-source fallback 0.0.0.X 255.255.255.0 global configuration command.	• Set source to VLAN SVI if present. • Look for IP and MAC binding in device-tracking table from same subnet. • Compute source IP from client IP using host bit and mask provided. Source MAC is taken from the MAC address of the switchport facing the client*.	We recommend that you disable device-tracking on all trunk ports to avoid MAC flapping. The computed IPv4 address must not be assigned to any client or network device.
device-tracking tracking auto-source fallback 0.0.0.X 255.255.255.0 override global configuration command.	• Set source to VLAN SVI if present. Compute source IP from client IP using host bit and mask provided*. Source MAC is taken from the MAC address of the switchport facing the client*.	-

* Depending on the client IP address, an IPv4 address has to be reserved for the source IP.

A reserved source IPv4 address = (host-ip and mask) | client-ip

- Client IP = 192.0.2.25
- Source IP = (192.0.2.25 and 255.255.255.0) | (0.0.0.1) = 192.0.2.1

IP address 192.0.2.1 should not be assigned to any client or network device.

Example: Disable IPv6 device tracking on a target

By default, SISF-based device-tracking supports both IPv4 and IPv6. These configuration examples show how you can disable IPv6 device-tracking where supported.

To disable device-tracking for IPv6, when a *custom* policy is attached to a target (all releases):

```
Device(config)# device-tracking policy example-policy
Device(config-device-tracking)# no protocol ndp
Device(config-device-tracking)# no protocol dhcp6
Device(config-device-tracking)# end
```

Example: Enable IPv6 for SVI on VLAN

When IPv6 is enabled in the network and a switched virtual interface (SVI) is configured on a VLAN, we recommend that you add the following to the SVI configuration. This configuration enables the SVI to automatically acquire a link-local address; this address is used as the source IP address of the SISF probe, thus preventing the duplicate IP address issue.

```
Device> enable
Device# configure terminal
Device(config)# interface vlan 10
Device(config-if)# ipv6 enable
Device(config-if)# end
```

Example: Configure a multi-switch network to stop creating binding entries from a trunk port

In a multi-switch network, SISF-based device tracking provides the capability to distribute binding table entries between switches running the feature. Both the trusted-port and device-role switch options must be configured in the policy. No entry is created for a host that appears over a trunk port. This is achieved by configuring a policy with the **trusted-port** and **device-role switch** options, and attaching it to the trunk port.



Note Both the **trusted-port** and **device-role switch** options must be configured in the policy.

Furthermore, we recommend applying such a policy to a port facing a device that also has SISF-based device tracking enabled.

```
Device> enable
Device# configure terminal
Device(config)# device-tracking policy example_trusted_policy
Device(config-device-tracking)# device-role switch
Device(config-device-tracking)# trusted-port
Device(config-device-tracking)# exit
Device(config)# interface gigabitethernet 1/0/25
Device(config-if)# device-tracking attach-policy  example_trusted_policy
Device(config-if)# end
```

Example: Avoid a short device-tracking binding reachable time

When migrating from an older release, this configuration may be present:

```
device-tracking binding reachable-lifetime 10
```

Remove this by entering the **no** version of the command.

```
Device> enable
Device# configure terminal
Device(config)# no device-tracking binding reachable-lifetime 10
Device(config)# end
```

Example: Detect and prevent spoofing

Address spoofing is a man-in-the-middle attack that allows an attacker to intercept communication between network devices. These attacks attempt to divert traffic from its originally intended host to the attacker instead.

For example, attacks are carried out by sending unsolicited Address Resolution Protocol (ARP) replies or with IPv6 Neighbor Advertisements carrying a mapping that is different from the legitimate one, such as <IPTARGET, MACTHIEF>. When the IPTARGET is of the default gateway, all traffic that is meant to leave the subnet is routed to the attacker.

This example shows shows the required SISF configuration to enable the system to detect and prevent spoofing. It also shows the system messages that are logged when spoofing is detected, and the action that the system takes. It includes an excerpt of LISP configuration in an SDA setup for example purposes only. Actual LISP configuration may involve additional configuration.

Sample LISP configuration:

```
instance-id 100
  service ethernet
   eid-table vlan 100               <<< triggers creation of programmatic policy
"LISP-DT-GUARD-VLAN"
   database-mapping mac locator-set XTR11
   exit-service-ethernet
  !
  exit-instance-id
```

Settings of the programmatic policy:

```
Device# show device-tracking policy LISP-DT-GUARD-VLAN

Device-tracking policy LISP-DT-GUARD-VLAN configuration:
  security-level guard              <<< enables the detection and prevention of IPv4 and
IPv6 spoofing
  device-role node
  gleaning from Neighbor Discovery
  gleaning from DHCP
  gleaning from ARP
  gleaning from DHCP4
  NOT gleaning from protocol unkn
  limit address-count for IPv4 per mac 21
  limit address-count for IPv6 per mac 58
  origin fabric
  tracking enable reachable-lifetime 240
```

This device-tracking counters show you that packet drops have occurred. However, the drops may be caused by reasons other than address spoofing as well. Use the information in the counters along with system messages to ascertain if spoofing has occurred.

```
Device# show device-tracking counters vlan 11

Received messages on vlan 11   :
Protocol        Protocol message
NDP             RS[4] RA[4] NS[1777] NA[2685]
```

```
DHCPv6
ARP             REQ[12] REP[1012]
DHCPv4
ACD&DAD         --[8]
:
Dropped messages on vlan 10   :
Feature             Protocol Msg [Total dropped]
Device-tracking:    ARP      REQ [23]
                    reason:  Packet accepted but not forwarded [23]
                             REP [450]
                    reason:  Silent drop [445]
                    reason:  Packet accepted but not forwarded [5] :
```

Required configuration to display system messages:

```
Device# device-tracking logging theft
Device# device-tracking logging packet drop
```

While the packet drops in the device-tracking counters do not conclusively prove that spoofing has occurred, the system messages help you ascertain this.

```
%SISF-4-IP_THEFT: IP Theft IP=3001::5 VLAN=10 Cand-MAC=aabb.cc00.6600 Cand-I/F=Et0/0 Known
 MAC aabb.cc00.6900 Known I/F Et0/1
%SISF-4-IP_THEFT: IP Theft IP=FE80::A8BB:CCFF:FE00:6900 VLAN=10 Cand-MAC=aabb.cc00.6600
Cand-I/F=Et0/0 Known MAC aabb.cc00.6900 Known I/F Et0/1
```

In the log, verified binding information, including IP, MAC address, interface, or VLAN, is preceded by the term 'Known'. A suspicious IP address and MAC address is preceded by the term "New" or "Cand". Interface and VLAN information is also provided along with the suspiscious IP or MAC address - this helps you identify where the suspiscious traffic was seen.

For more information about interpreting these system messages, refer to the usage guidelines of the **device-tracking logging** command in the command reference of the corresponding release.