

Security Configuration Guide, Cisco Catalyst IE9300 Rugged Series Switches

First Published: 2022-05-20

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

© 2022–2026 Cisco Systems, Inc. All rights reserved.

Full Cisco Trademarks with Software License

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)

Communications, services, and additional information

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

Cisco Bug Search Tool

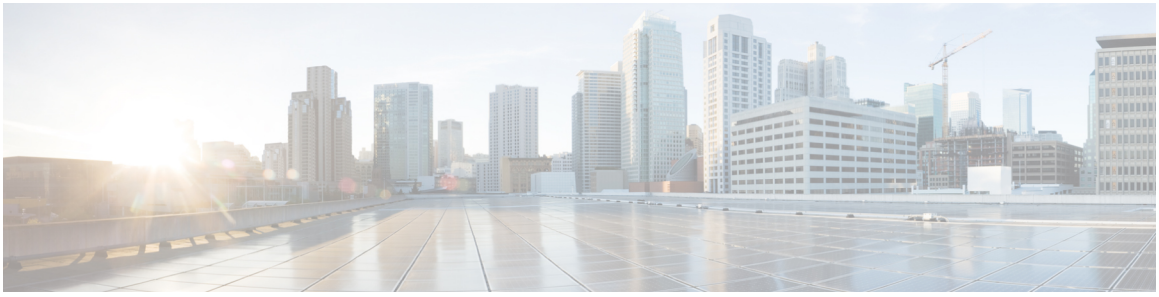
Cisco Bug Search Tool (BST) is a gateway to the Cisco bug-tracking system, which maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. The BST provides you with detailed defect information about your products and software.

Documentation feedback

To provide feedback about Cisco technical documentation, use the feedback form available in the right pane of every online document.

Bias Free Language

The documentation set for this product strives to use bias-free language. For purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on standards documentation, or language that is used by a referenced third-party product.



CONTENTS

Full Cisco Trademarks with Software License iii

Communications, services, and additional information iv

Cisco Bug Search Tool iv

Documentation feedback iv

Bias Free Language v

CHAPTER 1

RADIUS 1

RADIUS systems 1

How RADIUS operations work 2

RADIUS change of authorization 3

Change-of-Authorization requests 5

RFC 5176 compliance 5

CoA request response codes 6

Session identification 7

CoA ACK response codes 8

CoA NAK response codes 8

CoA request commands 8

Session reauthentication 8

Session terminations 9

Processing disconnect requests 10

CoA requests: disable host port 10

Coa requests: bounce port 11

RADIUS and switch access 12
RADIUS server hosts 12
RADIUS login authentication 13
AAA server groups 13
AAA Authorization 14
RADIUS accounting 14
Vendor-specific RADIUS attributes 14
Vendor-proprietary RADIUS server communications 25
DSCP markings for RADIUS packets 25
Restrictions for configuring RADIUS 26
Prerequisites for configuring RADIUS 27
RADIUS 28
Default RADIUS configurations 28
Identify the RADIUS server host 28
Configure RADIUS login authentication 30
Define AAA server groups 32
Configure RADIUS authorization for user privileged access and network services 33
Start RADIUS accounting 34
Configure settings for all RADIUS servers 35
Configure vendor-specific RADIUS attributes 36
Configure the device for vendor-proprietary RADIUS server communication 37
Configure DSCP marking on a RADIUS server 38
Configure the source interface and DSCP marking on a RADIUS server group 40
Configure CoA 41
Monitor CoA 43

CHAPTER 2 **MACsec Encryption 45**
MACsec Encryption 45
MACsec Key Agreement 46
MKA Policies 47
Definition of Policy-Map Actions 47
Virtual Ports 47
MKA Statistics 48
Key Lifetime and Hitless Key Rollover 48

Default timer-based SAK rekey 48
Replay Protection Window Size 49
MACsec, MKA and 802.1x Host Modes 49
Single-Host Mode 49
Multiple-Host Mode 49
Multiple-Domain Mode 50
MACsec MKA using Certificate-based MACsec 50
Prerequisites for MACsec MKA Using Certificate-based MACsec 50
Switch-to-Switch MKA MACsec Must Secure Policy 51
MKA/MACsec for Port Channel 51
MACsec Cipher Announcement 51
Limitations for MACsec Cipher Announcement 52
How to Configure MACsec Encryption 52
Prerequisites for MACsec Encryption 52
Restrictions for MACsec Encryption 52
Recommendations for MACsec Encryption 53
MKA and MACsec Configuration 53
Configure an MKA Policy 53
Configure Switch-to-host MACsec Encryption 55
Configure MACsec MKA using PSK 58
Configure MACsec MKA on an Interface using PSK 59
Configuring MACsec MKA Using Certificate-based MACsec 61
Generate Key Pairs 61
Configure Enrollment using SCEP 62
Configure Enrollment Manually 64
Enable 802.1x Authentication and Configure AAA 67
Apply the 802.1x MKA MACsec Configuration on the Interfaces 69
Configure MKA/MACsec for Port Channel using PSK 71
Configure Port Channel Logical Interfaces for Layer 2 EtherChannels 73
Configure Port Channel Logical Interfaces for Layer 3 EtherChannels 74
Configuring MACsec Cipher Announcement 74
Configure an MKA Policy for Secure Announcement 74
Configure Secure Announcement Globally 76
Configure EAPoL Announcements on an Interface 76

Configuration Examples for MACsec Encryption 77
Example: Configuring MKA and MACsec 77
Examples: Configuring MACsec MKA Using PSK 77
Examples: Configuring MACsec MKA using Certificate-based MACsec 79
Examples: Configuring MACsec MKA for Port Channel using PSK 79
Examples: Configuring MACsec Cipher Announcement 86
Examples: Displaying MKA Information 89
Additional References for MACsec Encryption 95
Feature History for MACsec Encryption 96
Overview of PQC for MACsec with EAP-TLS 96
Key elements of PQC 97
PQC implementation in EAP-TLS 98
Prerequisites 100
Restrictions 100
Configuration matrix for MACsec with EAP-TLS 100
Workflow to set up PQC for MACsec with EAP-TLS 101
Configure EAP-TLS 1.3 101
Configure PQC for EAP-TLS sessions 102
Verify PQC for EAP-TLS sessions 102

CHAPTER 3 **Post Quantum Cryptography for SSHv2 sessions 105**
Implementation of Post Quantum Cryptography 105
Overview of PQC for SSHv2 sessions 105
Factors to consider for deploying PQC 106
Protocol requirements 106
Restrictions for PQC 106
PQC negotiation scenarios 106
Configure PQC for SSHv2 sessions 107
Verify PQC for SSHv2 sessions 108
Key elements of PQC 108

CHAPTER 4 **Network Edge Access Topology 111**
802.1x supplicant and authenticator switches with network edge access topology 111
Guidelines and limitations for NEAT configuration 113

Configure an authenticator switch with NEAT 113
Configure a supplicant switch with NEAT 115
Configuration verification 117
Feature History 119

CHAPTER 5 FlexLink+ 121
FlexLink+ redundancy functions 121
Guidelines and restrictions 122
Configure active port for Flex Link+ 122
Configure standby port for FlexLink+ 123
Configure VLAN load balancing on FlexLink+ 124
Configure time for VLAN load balancing on FlexLink+ 125
Configure manual preemption for VLAN load balancing on FlexLink+ 125
Configure FlexLink+ topology change messages 126

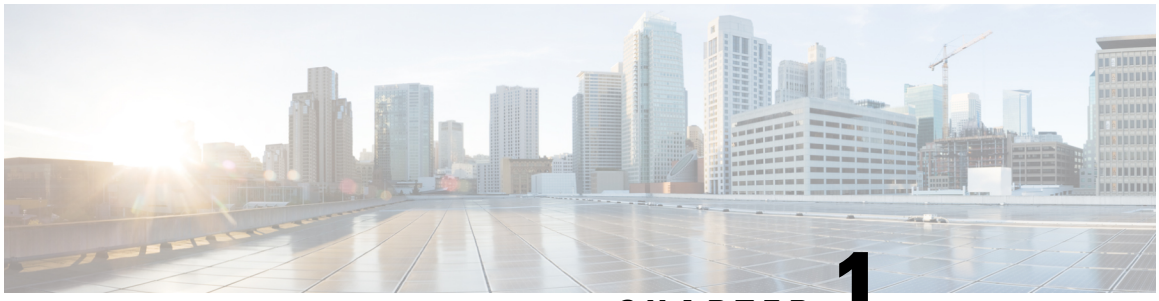
CHAPTER 6 Wired dynamic PVLAN 127
Wired dynamic PVLANs 127
Restrictions for wired dynamic PVLAN 129
Configure wired dynamic PVLAN 129

CHAPTER 7 IPv4 ACLs 133
Restrictions for IPv4 access control lists 133
Information About IPv4 Access Control Lists 135
Access control lists 135
Access control entries 136
ACL supported types 136
Supported ACLs 136
ACL precedences 136
Port ACLs 137
Router ACLs 139
VLAN maps 140
ACEs and fragmented and unfragmented traffic 140
Standard and extended IPv4 ACLs 141
Restrictions for IPv4 ACL 142

Access list numbers 142
Numbered standard IPv4 ACLs 143
Numbered extended IPv4 ACLs 143
Named IPv4 ACLs 144
ACL logging 144
Hardware and software treatment of IP ACLs 145
Guidelines for VLAN map configuration 146
VLAN Maps with Router ACLs 147
Guidelines for VLAN maps and router ACL configuration 147
Time ranges for ACLs 147
IPv4 ACL interface considerations 148
How to Configure IPv4 Access Control Lists 148
Configure IPv4 ACLs 148
Create a numbered standard ACL 149
Create a numbered extended ACL 150
Create a named standard ACL 153
Create an extended named ACL 154
Configure time ranges for ACLs 155
Apply an IPv4 ACL to a terminal line 156
Apply an IPv4 ACL to an interface 158
Configure a named MAC extended ACL 159
Apply a MAC ACL to a layer 2 interface 160
Configure an IPv4 ACL in template mode 162
Configure a VLAN map 164
Apply a VLAN map to a VLAN 166
IPv4 ACL monitoring commands 167
Configuration Examples for IPv4 Access Control Lists 168
ACLs in small networked offices 168
Configure ACLs in a small networked office 169
Configure numbered ACLs 170
Configure extended access control lists 171
Configure named ACLs 172
ACL logging examples 174
Configuration examples for ACEs and fragmented and unfragmented traffic 175

Configure time ranges with ACLs 176
Configure a time range for an IP access control list 178
Include comments in ACLs 179
Create an ACL and a VLAN map to deny a packet 179
Create an ACL and a VLAN map to permit a packet 180
Configure default action for VLAN map to drop IP packets and forward MAC packets 181
Configure default action of dropping MAC packets and forwarding IP packets 183
Configure a VLAN map with default action to drop all packets 184
Example: Using VLAN Maps in a Network 185
Configure a wiring closet VLAN map to drop HTTP traffic 185
Restricting access to servers on other VLANs 186
Deny access to a SERVER on another VLAN 187

CHAPTER 8 IPv6 ACLs 189
IPv6 access control lists 189
Guidelines and restrictions for IPv6 ACLs 189
Supported ACLs 190
Types of ACL 190
Per-user IPv6 ACLs 191
Filter ID IPv6 ACLs 191
ACL precedences 191
VLAN maps 191
Interactions with other features and switches 192
Configure an IPv6 ACL 193
Default configuration for IPv6 ACLs 193
Configure IPv6 ACLs 193
Attach an IPv6 ACL to an interface 196
Configure an IPv6 ACL in template mode 197
Configure a VLAN map 199
Apply a VLAN map to a VLAN 201
IPv6 ACL monitoring commands 201



CHAPTER 1

RADIUS

- RADIUS systems, on page 1
- How RADIUS operations work, on page 2
- RADIUS change of authorization, on page 3
- RADIUS and switch access, on page 12
- RADIUS server hosts, on page 12
- RADIUS login authentication, on page 13
- AAA server groups, on page 13
- AAA Authorization, on page 14
- RADIUS accounting, on page 14
- Vendor-specific RADIUS attributes, on page 14
- Vendor-proprietary RADIUS server communications, on page 25
- DSCP markings for RADIUS packets, on page 25
- Restrictions for configuring RADIUS, on page 26
- Prerequisites for configuring RADIUS, on page 27
- RADIUS, on page 28
- Monitor CoA, on page 43

RADIUS systems

A RADIUS system is a distributed client/server authentication protocol that

- secures networks against unauthorized access
- enables clients on supported Cisco devices to send authentication requests to a central RADIUS server, and
- stores all user authentication and network service access information on the server.

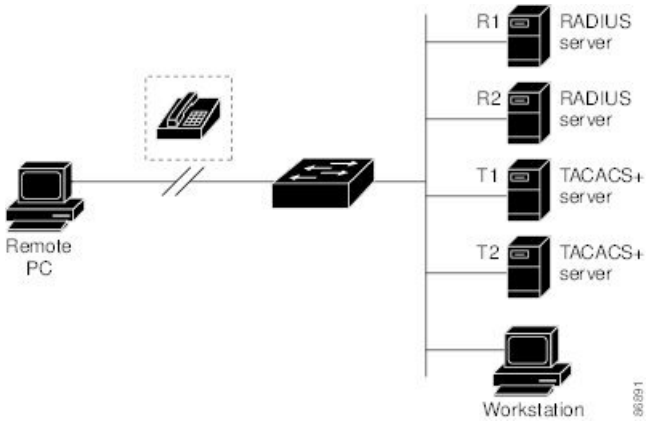
Suggested network environments for RADIUS

RADIUS is used in network environments that require access security, including:

- Networks with multiple-vendor access servers, each supporting RADIUS. For example, access servers from several vendors use a single RADIUS server-based security database. In an IP-based network with multiple vendors' access servers, dial-in users are authenticated through a RADIUS server that has been customized to work with the Kerberos security system.

- Turnkey network security environments in which applications support the RADIUS protocol, such as in an access environment that uses a *smart card* access control system.
- Networks already using RADIUS. You can add a Cisco device containing a RADIUS client to the network. This might be the first step when you make a transition to a TACACS+ server.

Figure 1: Transitioning from RADIUS to TACACS+ services



- Networks in which the user must only access a single service. Using RADIUS, you can control user access to a single host, to a single utility such as Telnet, or to the network through a protocol such as IEEE 802.1x. For more information about this protocol, see the chapter *Configuring IEEE 802.1x Port-Based Authentication*.
- Networks that require resource accounting. You can use RADIUS accounting independently of RADIUS authentication or authorization. The RADIUS accounting functions allow data to be sent at the start and end of services, showing the amount of resources (such as time, packets, bytes, and so forth) used during the session. An Internet service provider might use a freeware-based version of RADIUS access control and accounting software to meet special security and billing needs.

How RADIUS operations work

RADIUS (Remote Authentication Dial-In User Service) is used to control access to devices by authenticating users through a centralized server. When a user attempts to log in to a device protected by a RADIUS server, a series of authentication steps are performed.

Summary

The key components involved in the RADIUS authentication process are:

- User: Attempts to access a RADIUS-protected device and provides login credentials.
- Device: Prompts the user for credentials and communicates with the RADIUS server.
- RADIUS server: Receives authentication requests and responds with access decisions and additional authorization data.

This process ensures that only authenticated users can access network devices and that appropriate authorization parameters are applied.

Workflow

The RADIUS authentication process involves the following stages:

1. Stage 1: The user attempts to log in to a device that is access controlled by a RADIUS server.
   - The device prompts the user to enter a username and PASSWORD.
2. Stage 2: The device sends the user's credentials to the RADIUS server.
   - The username and encrypted PASSWORD are transmitted over the network to the RADIUS server.
3. Stage 3: The RADIUS server processes the authentication request and responds to the user.
   - The user receives one of the following responses from the RADIUS server:
     - **ACCEPT**—The user is authenticated.
     - **REJECT**—The user is either not authenticated and is prompted to re-enter the username and PASSWORD, or access is denied.
     - **CHALLENGE**—A CHALLENGE requires additional data from the user.
     - **CHALLENGE PASSWORD**—A response requests the user to select a new PASSWORD.
   - The **ACCEPT** or **REJECT** response is bundled with additional data used for privileged EXEC or network authorization. The additional data included with the ACCEPT or REJECT packets includes:
     - Telnet, SSH, rlogin, or privileged EXEC services
     - Connection parameters, including the host or client IP address, access list, and user timeouts

RADIUS change of authorization

A RADIUS change of authorization is a AAA session management mechanism that

- provides a way to change the attributes of an authentication, authorization, and accounting (AAA) session after authentication
- allows administrators to send CoA packets from the AAA server to reinitialize authentication and apply new policies, and
- supports dynamic session reconfiguration using RADIUS CoA extensions as defined in RFC 5176.

RADIUS CoA interface and supported features

The RADIUS interface is typically used in a pulled model where the request originates from a network-attached device and the response comes from the queried servers. Cisco devices support the RADIUS CoA extensions defined in RFC 5176, which are used in a pushed model and allow for dynamic reconfiguring of sessions from external AAA or policy servers.

Cisco devices support these per-session CoA requests:

- Session reauthentication

- Session termination
- Session termination with port shutdown
- Session termination with port bounce

This feature is integrated with Cisco Secure Access Control Server (ACS) 5.1.

The RADIUS interface is enabled by default on Cisco devices. However, some basic configuration is required for the following attributes:

- Security and Password—refer to the "Preventing Unauthorized Access to Your Switch" section in this guide.
- Accounting—refer to the "Starting RADIUS Accounting" section in the Configuring Switch-Based Authentication chapter in this guide.

Cisco IOS XE software supports the RADIUS CoA extensions defined in RFC 5176, which are typically used in a push model to allow the dynamic reconfiguring of sessions from external AAA or policy servers. Per-session CoA requests are supported for session identification, session termination, host reauthentication, port shutdown, and port bounce. This model comprises one request (CoA-Request) and two possible response codes:

- CoA acknowledgement (ACK) [CoA-ACK]
- CoA nonacknowledgement (NAK) [CoA-NAK]

The request is initiated from a CoA client (typically a AAA or policy server) and directed to the device that acts as a listener.

The following table shows the RADIUS CoA commands and vendor-specific attributes (VSAs) supported by Identity-Based Networking Services. All CoA commands must include the session identifier between the device and the CoA client.

Table 1: RADIUS CoA commands supported by Identity-Based Networking Services

CoA command	Cisco VSA
Activate service	Cisco:Avpair="subscriber:command=activate-service" Cisco:Avpair="subscriber:service-name=<*service-name*>" Cisco:Avpair="subscriber:precedence=<*precedence-number*>" Cisco:Avpair="subscriber:activation-mode=replace-all"
Deactivate service	Cisco:Avpair="subscriber:command=deactivate-service" Cisco:Avpair="subscriber:service-name=<*service-name*>"
Bounce host port	Cisco:Avpair="subscriber:command=bounce-host-port"
Disable host port	Cisco:Avpair="subscriber:command=disable-host-port"
Session query	Cisco:Avpair="subscriber:command=session-query"

CoA command	Cisco VSA
Session reauthenticate	Cisco:Avpair="subscriber:command=reauthenticate" Cisco:Avpair="subscriber:reauthenticate-type=last" or Cisco:Avpair="subscriber:reauthenticate-type=rerun"
Session terminate	This is a standard disconnect request and does not require a VSA.
Interface template	Cisco:AVpair="interface-template-name=<*interfacetemplate*>"

Change-of-Authorization requests

A Change-of-Authorization request is a RADIUS protocol message that

- enables session identification, host reauthentication, and session termination in a push model
- is initiated from a CoA client (typically a RADIUS or policy server) and directed to the switch acting as a listener.

CoA request response codes

Change-of-Authorization requests use the following response codes:

- CoA acknowledgment (ACK) [CoA-ACK]
- CoA non-acknowledgment (NAK) [CoA-NAK]

RFC 5176 compliance

The switch supports the Disconnect Request message, also referred to as Packet of Disconnect (POD), for session termination.

This table shows the IETF attributes are supported for this feature.

Table 2: Supported IETF attributes

Attribute Number	Attribute Name
24	State
31	Calling-Station-ID
44	Acct-Session-ID
80	Message-Authenticator
101	Error-Cause

This table shows the possible values for the Error-Cause attribute.

Table 3: Error-Cause values

Value	Explanation
201	Residual Session Context Removed
202	Invalid EAP Packet (Ignored)
401	Unsupported Attribute
402	Missing Attribute
403	NAS Identification Mismatch
404	Invalid Request
405	Unsupported Service
406	Unsupported Extension
407	Invalid Attribute Value
501	Administratively Prohibited
502	Request Not Routable (Proxy)
503	Session Context Not Found
504	Session Context Not Removable
505	Other Proxy Processing Error
506	Resources Unavailable
507	Request Initiated
508	Multiple Session Selection Unsupported

CoA request response codes

A CoA request response code is a RADIUS protocol message that

- conveys a command to the switch
- follows the packet format defined in RFC 5176, and
- includes fields such as Code, Identifier, Length, Authenticator, and Attributes in TLV format.

Packet format details for CoA request response codes

The packet format for a CoA request response code as defined in RFC 5176 consists of the following fields: Code, Identifier, Length, Authenticator, and Attributes in the Type:Length:Value (TLV) format. The Attributes field is used to carry Cisco vendor-specific attributes (VSAs).

Session identification

A session identification is a set of attributes that

- allows the switch to locate a session for disconnect and CoA requests
- uses one or more specific identification attributes, and
- requires all included attributes to match the session for successful processing.

Session identification attributes

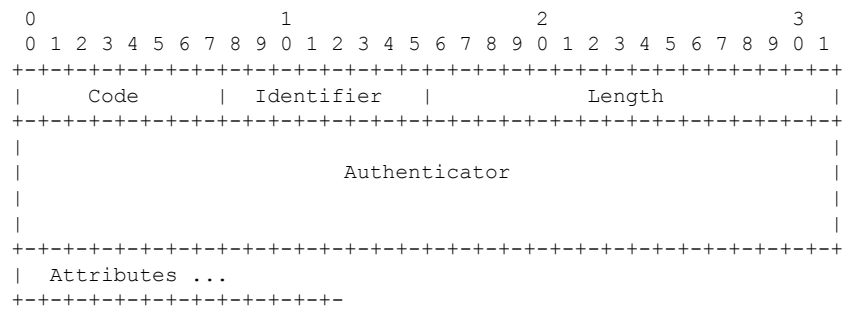
The switch locates the session based on one or more of the following attributes:

- Acct-Session-Id (IETF attribute #44)
- Audit-Session-Id (Cisco VSA)
- Calling-Station-Id (IETF attribute #31 which contains the host MAC address)
- IPv6 attributes, which can be one of the following:
  - Framed-IPv6-Prefix (IETF attribute #97) and Framed-Interface-Id (IETF attribute #96), which together create a full IPv6 address per RFC 3162
  - Framed-IPv6-Address
- Plain IP Address (IETF attribute #8)

Unless all session identification attributes included in the CoA message match the session, the switch returns a Disconnect-NAK or CoA-NAK with the “Invalid Attribute Value” error-code attribute.

If more than one session identification attribute is included in the message, all the attributes must match the session or the switch returns a Disconnect-negative acknowledgment (NAK) or CoA-NAK with the error code “Invalid Attribute Value.”

The packet format for a CoA Request code as defined in RFC 5176 consists of the fields: Code, Identifier, Length, Authenticator, and Attributes in Type:Length:Value (TLV) format.



The attributes field is used to carry Cisco vendor-specific attributes (VSAs).

For CoA requests targeted at a particular enforcement policy, the device returns a CoA-NAK with the error code “Invalid Attribute Value” if any of the above session identification attributes are included in the message.

CoA ACK response codes

A CoA ACK response code is a RADIUS protocol response that

- is sent when the authorization state is changed successfully
- provides a positive acknowledgment (ACK), and
- returns attributes that vary based on the CoA request and individual CoA commands.

CoA NAK response codes

A CoA NAK response code is a RADIUS protocol message that

- indicates a failure to change the authorization state
- can include attributes that specify the reason for the failure, and
- is used to verify unsuccessful CoA operations using **show** commands.

CoA request commands

This reference provides information about the supported Change of Authorization (CoA) request commands and their associated Cisco Vendor-Specific Attributes (VSAs).

Table 4: Supported CoA commands

Command [1]	Cisco VSA
Reauthenticate host	Cisco:Avpair="subscriber:command=reauthenticate"
Terminate session	This is a standard disconnect request that does not require a VSA.
Bounce host port	Cisco:Avpair="subscriber:command=bounce-host-port"
Disable host port	Cisco:Avpair="subscriber:command=disable-host-port"

[1] All CoA commands must include the session identifier between the device and the CoA client.

Session reauthentication

A session reauthentication is a network access control mechanism that

- enables the AAA server to generate a reauthentication request for hosts with unknown identity or posture
- allows the host to be placed in the appropriate authorization group when credentials are known, and
- maintains current authorization until reauthentication leads to a different authorization result.

Session reauthentication process details

The AAA server initiates session authentication by sending a standard CoA-Request message containing a Cisco VSA in this form: *Cisco:Avpair="subscriber:command=reauthenticate"* and one or more session identification attributes.

The current session state determines the switch response to the message.

- If the session is currently authenticated by IEEE 802.1x, the switch responds by sending an EAPoL (Extensible Authentication Protocol over Lan) -RequestId message to the server.
- If the session is currently authenticated by MAC authentication bypass (MAB), the switch sends an access-request to the server, passing the same identity attributes used for the initial successful authentication.
- If session authentication is in progress when the switch receives the command, the switch terminates the process and restarts the authentication sequence, starting with the method configured to be attempted first.
- If the session is not yet authorized, or is authorized via guest VLAN, or critical VLAN, or similar policies, the reauthentication message restarts the access control methods, beginning with the method configured to be attempted first.

Session terminations

A session termination is a network management action that

- can be triggered by three types of CoA requests
- ends the session without disabling the host port, and
- re-initializes the authenticator state machine for the specified host without restricting network access.

Session termination reference information

Session terminations can be managed using CoA Disconnect-Request or CoA Request with specific VSAs.

To restrict a host's access to the network, use a CoA Request with the Cisco:Avpair="subscriber:command=disable-host-port" VSA.

- This command is useful when a host is causing problems on the network and immediate blocking is required.
- Restoring network access requires re-enabling the port using a non-RADIUS mechanism.

When a device with no supplicant, such as a printer, needs to acquire a new IP address after a VLAN change, terminate the session on the host port with port-bounce (temporarily disable and then re-enable the port).

Session termination examples

For example, a CoA Disconnect-Request terminates a session for a host without disabling the port, allowing continued network access. Using the disable-host-port command immediately blocks network access for problematic hosts. Port-bounce is used for devices like printers to refresh their IP address after a VLAN change.

Processing disconnect requests

The Disconnect-Request command is used to terminate sessions on a device. The device responds based on whether the session can be located and may repeat the process in failover situations.

Summary

The key components involved in processing disconnect requests are:

- Device: Locates and terminates sessions, and returns appropriate responses.
- Client: Sends Disconnect-Request commands and may resend requests after failover.
- Session: Represents the connection to be terminated.

This process ensures sessions are properly terminated and clients receive accurate status responses.

Workflow

The processing of disconnect requests involves the following stages:

1. Device receives a Disconnect-Request command from the client.
    - Device attempts to locate the session referenced in the request.

When...	And...	Then...	And...
Session cannot be located		Device returns Disconnect-NAK with “Session Context Not Found” error-code attribute	
Session is located		Device terminates the session	After session removal, device returns Disconnect-ACK
Device fails-over to standby before returning Disconnect-ACK	Client resends Disconnect-Request to new active device	Process is repeated on new device	If session not found, Disconnect-ACK is sent with “Session Context Not Found” error-code attribute

Result

The process ensures that sessions are terminated and clients are notified of the session status, even in failover scenarios.

CoA requests: disable host port

A CoA request to disable host port is a RADIUS server command that

- administratively shuts down the authentication port hosting a session
- results in session termination and immediate network access blocking for the host, and

- is carried in a standard CoA-Request message with a vendor-specific attribute: Cisco:Avpair="subscriber:command=disable-host-port".

Session identification and response behavior

Because this command is session-oriented, it must be accompanied by one or more of the session identification attributes described in the “Session Identification” section.

If the session cannot be located, the device returns a CoA-NAK message with the “Session Context Not Found” error-code attribute. If the session is located, the device disables the hosting port and returns a CoA-ACK message.

If the device fails before returning a CoA-ACK to the client, the process is repeated on the new active device when the request is re-sent from the client. If the device fails after returning a CoA-ACK message to the client but before the operation has completed, the operation is restarted on the new active device.



Note A Disconnect-Request failure following command re-sending could be the result of either a successful session termination before change-over (if the Disconnect-ACK was not sent) or a session termination by other means (for example, a link failure) that occurred after the original command was issued and before the standby device became active.

Coa requests: bounce port

A CoA bounce port request is a RADIUS server command that

- can cause a link flap on an authentication port, triggering DHCP renegotiation from one or more hosts connected to the port
- is carried in a standard CoA-Request message containing the VSA: Cisco:Avpair="subscriber:command=bounce-host-port", and
- is session-oriented and must be accompanied by one or more session identification attributes.

Session handling and device behavior for CoA bounce port requests

When a CoA bounce port request is received, the device processes it as follows:

- If the session cannot be located, the device returns a CoA-NAK message with the “Session Context Not Found” error-code attribute.
- If the session is located, the device disables the hosting port for a period of 10 seconds, re-enables it (port-bounce), and returns a CoA-ACK.

If the device fails before returning a CoA-ACK to the client, the process is repeated on the new active device when the request is re-sent from the client. If the device fails after returning a CoA-ACK message to the client but before the operation has completed, the operation is re-started on the new active device.

This incident can occur when there is a VLAN change and the endpoint is a device (such as a printer) that does not have a mechanism to detect a change on this authentication port.

RADIUS and switch access

A RADIUS and switch access is a network authentication and authorization mechanism that

- enables and configures RADIUS for network devices
- provides detailed accounting information, and
- offers flexible administrative control over authentication and authorization processes.

RADIUS server hosts

A RADIUS server host is a network authentication server that

- is identified by its hostname or IP address and specific UDP port numbers
- uses a shared secret text string to encrypt passwords and exchange responses, and
- supports configuration of timeout, retransmission, and encryption key values globally or per server.

Device-to-RADIUS-server communication components

Device-to-RADIUS-server communication involves several components:

- Hostname or IP address
- Authentication destination port
- Accounting destination port
- Key string
- Timeout period
- Retransmission value

You identify RADIUS security servers by their hostname or IP address, hostname and specific UDP port numbers, or their IP address and specific UDP port numbers. The combination of the IP address and the UDP port number creates a unique identifier, allowing different ports to be individually defined as RADIUS hosts providing a specific AAA service. This unique identifier enables RADIUS requests to be sent to multiple UDP ports on a server at the same IP address.

If two different host entries on the same RADIUS server are configured for the same service—for example, accounting—the second host entry configured acts as a fail-over backup to the first one. Using this example, if the first host entry fails to provide accounting services, the %RADIUS-4-RADIUS_DEAD message appears, and then the device tries the second host entry configured on the same device for accounting services. The RADIUS host entries are tried in the order that they are configured.

A RADIUS server and the device use a shared secret text string to encrypt passwords and exchange responses. To configure RADIUS to use the AAA security commands, you must specify the host running the RADIUS server daemon and a secret text (key) string that it shares with the device.

The timeout, retransmission, and encryption key values can be configured globally for all RADIUS servers, on a per-server basis, or in some combination of global and per-server settings.

RADIUS login authentication

A RADIUS login authentication is an AAA authentication mechanism that

- uses named lists of authentication methods applied to ports
- defines the sequence and types of authentication to be performed, and
- ensures backup authentication by querying multiple methods if the initial method fails.

Authentication method lists

Method lists describe the sequence and authentication methods to be queried to authenticate a user.

- You can designate one or more security protocols for authentication.
- The software uses the first method listed to authenticate users.
- If the first method fails to respond, the software selects the next authentication method in the list.
- This process continues until successful communication with a listed authentication method or until all methods are exhausted.
- If authentication fails at any point, the process stops and no other methods are attempted.

The default method list is automatically applied to all ports except those with a named method list explicitly defined.

AAA server groups

An AAA server group is a server grouping mechanism that

- enables you to group existing server hosts for authentication
- allows selection of a subset of configured server hosts for a particular service, and
- uses a global server-host list to specify the IP addresses of selected server hosts.

Server group configuration details

Server groups can include multiple host entries for the same server if each entry has a unique identifier, which is the combination of the IP address and UDP port number. This allows different ports to be individually defined as RADIUS hosts providing a specific AAA service.

This unique identifier enables RADIUS requests to be sent to different UDP ports on a server at the same IP address.

If you configure two different host entries on the same RADIUS server for the same service, such as accounting, the second configured host entry acts as a fail-over backup to the first one. If the first host entry fails to provide accounting services, the network access server tries the second host entry configured on the same device for accounting services. The RADIUS host entries are tried in the order in which they are configured.

AAA Authorization

AAA authorization limits the services available to a user. When AAA authorization is enabled, the device uses information retrieved from the user's profile, which is in the local user database or on the security server, to configure the user's session. The user is granted access to a requested service only if the information in the user profile allows it.

RADIUS accounting

A RADIUS accounting feature is an AAA accounting mechanism that

- tracks the services that users are using and the amount of network resources they are consuming
- reports user activity to the RADIUS security server in the form of accounting records, and
- stores each accounting record as attribute-value (AV) pairs on the security server for analysis.

RADIUS accounting reference information

RADIUS accounting enables analysis of user activity data for network management, client billing, or auditing.

Vendor-specific RADIUS attributes

The Internet Engineering Task Force (IETF) draft standard specifies a method for communicating vendor-specific information between the device and the RADIUS server by using the vendor-specific attribute (attribute 26). Vendor-specific attributes (VSAs) allow vendors to support their own extended attributes not suitable for general use. The Cisco RADIUS implementation supports one vendor-specific option by using the format recommended in the specification. Cisco's vendor-ID is 9, and the supported option has vendor-type 1, which is named *cisco-avpair*. The value is a string with this format:

```
protocol : attribute sep value *
```

Protocol is a value of the Cisco protocol attribute for a particular type of authorization. *Attribute* and *value* are an appropriate attributevalue (AV) pair defined in the Cisco TACACS+ specification, and *sep* is = for mandatory attributes and is * for optional attributes. The full set of features available for TACACS+ authorization can then be used for RADIUS.

For example, this AV pair causes Cisco's "multiple named IP address pools" feature to be activated during IP authorization (during PPP's Internet Protocol Control Protocol (IPCP) address assignment):

```
cisco-avpair= ″ip:addr-pool=first″
```

If you insert an "*", the AV pair "ip:addr-pool=first" becomes optional. Note that any AV pair can be made optional:

```
cisco-avpair= ″ip:addr-pool*first″
```

This example shows how to cause a user logging in from a network access server to have immediate access to EXEC commands:

```
cisco-avpair= ″shell:priv-lvl=15″
```

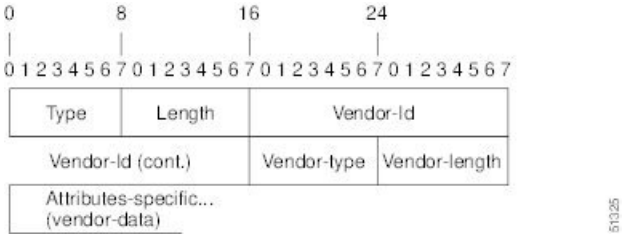
Other vendors have their own unique vendor-IDs, options, and associated VSAs. For more information about vendor-IDs and VSAs, see RFC 2138, "Remote Authentication Dial-In User Service (RADIUS)."

Attribute 26 contains the following three elements:

- Type
- Length
- String (also known as data)
  - Vendor-ID
  - Vendor-Type
  - Vendor-Length
  - Vendor-Data

The figure below shows the packet format for a VSA encapsulated "behind" attribute 26.

Figure 2: VSA Encapsulated Behind Attribute 26



Note It is up to the vendor to specify the format of their VSA. The Attribute-Specific field (also known as Vendor-Data) is dependent on the vendor's definition of that attribute.

The table below describes significant fields listed in the Vendor-Specific RADIUS IETF Attributes table (second table below), which lists supported vendor-specific RADIUS attributes (IETF attribute 26).

Table 5: Vendor-Specific Attributes Table Field Descriptions

Field	Description
Number	All attributes listed in the following table are extensions of IETF attribute 26.
Vendor-Specific Command Codes	A defined code used to identify a particular vendor. Code 9 defines Cisco VSAs, 311 defines Microsoft VSAs, and 529 defines Ascend VSAs.
Sub-Type Number	The attribute ID number. This number is much like the ID numbers of IETF attributes, except it is a "second layer" ID number encapsulated behind attribute 26.
Attribute	The ASCII string name of the attribute.
Description	Description of the attribute.

Table 6: Vendor-Specific RADIUS IETF Attributes

Number	Vendor-Specific Company Code	Sub-Type Number	Attribute	Description
MS-CHAP Attributes				
26	311	1	MSCHAP-Response	Contains the response value provided by a PPP MS-CHAP user in response to the challenge. It is only used in Access-Request packets. This attribute is identical to the PPP CHAP Identifier. (RFC 2548
26	311	11	MSCHAP-Challenge	Contains the challenge sent by a network access server to an MS-CHAP user. It can be used in both Access-Request and Access-Challenge packets. (RFC 2548)
VPDN Attributes				
26	9	1	l2tp-cm-local-window-size	Specifies the maximum receive window size for L2TP control messages. This value is advertised to the peer during tunnel establishment.
26	9	1	l2tp-drop-out-of-order	Respects sequence numbers on data packets by dropping those that are received out of order. This does not ensure that sequence numbers will be sent on data packets, just how to handle them if they are received.
26	9	1	l2tp-hello-interval	Specifies the number of seconds for the hello keepalive interval. Hello packets are sent when no data has been sent on a tunnel for the number of seconds configured here.

Number	Vendor-Specific Company Code	Sub-Type Number	Attribute	Description
26	9	1	l2tp-hidden-AVP	When enabled, sensitive AVPs in L2TP control messages are scrambled or hidden.
26	9	1	l2tp-nosession-timeout	Specifies the number of seconds that a tunnel will stay active with no sessions before timing out and shutting down.
26	9	1	tunnel-tos-reflect	Copies the IP ToS field from the IP header of each payload packet to the IP header of the tunnel packet for packets entering the tunnel at the LNS.
26	9	1	l2tp-tunnel-authen	If this attribute is set, it performs L2TP tunnel authentication.
26	9	1	l2tp-tunnel-password	Shared secret used for L2TP tunnel authentication and AVP hiding.
26	9	1	l2tp-UDP-checksum	This is an authorization attribute and defines whether L2TP should perform UDP checksums for data packets. Valid values are “yes” and “no.” The default is no.
Store and Forward Fax Attributes				
26	9	3	Fax-Account-ID-Origin	Indicates the account ID origin as defined by system administrator for the **mmoipaaareceive-ID** or the **mmoipaaasend-ID** commands.
26	9	4	Fax-Msg-ID=	Indicates a unique fax message identification number assigned by Store and Forward Fax.

Number	Vendor-Specific Company Code	Sub-Type Number	Attribute	Description
26	9	5	Fax-Pages	Indicates the number of pages transmitted or received during this fax session. This page count includes cover pages.
26	9	6	Fax-Coverpage-Flag	Indicates whether or not a cover page was generated by the off-ramp gateway for this fax session. True indicates that a cover page was generated; false means that a cover page was not generated.
26	9	7	Fax-Modem-Time	Indicates the amount of time in seconds the modem sent fax data (x) and the amount of time in seconds of the total fax session (y), which includes both fax-mail and PSTN time, in the form x/y. For example, 10/15 means that the transfer time took 10 seconds, and the total fax session took 15 seconds.
26	9	8	Fax-Connect-Speed	Indicates the modem speed at which this fax-mail was initially transmitted or received. Possible values are 1200, 4800, 9600, and 14400.
26	9	9	Fax-Recipient-Count	Indicates the number of recipients for this fax transmission. Until e-mail servers support Session mode, the number should be 1.
26	9	10	Fax-Process-Abort-Flag	Indicates that the fax session was terminated or successful. True means that the session was terminated; false means that the session was successful.

Number	Vendor-Specific Company Code	Sub-Type Number	Attribute	Description
26	9	11	Fax-DSN-Address	Indicates the address to which DSNs will be sent.
26	9	12	Fax-DSN-Flag	Indicates whether or not DSN has been enabled. True indicates that DSN has been enabled; false means that DSN has not been enabled.
26	9	13	Fax-MDN-Address	Indicates the address to which MDNs will be sent.
26	9	14	Fax-MDN-Flag	Indicates whether or not message delivery notification (MDN) has been enabled. True indicates that MDN had been enabled; false means that MDN had not been enabled.
26	9	15	Fax-Auth-Status	Indicates whether or not authentication for this fax session was successful. Possible values for this field are success, failed, bypassed, or unknown.
26	9	16	Email-Server-Address	Indicates the IP address of the e-mail server handling the on-ramp fax-mail message.
26	9	17	Email-Server-Ack-Flag	Indicates that the on-ramp gateway has received a positive acknowledgment from the e-mail server accepting the fax-mail message.
26	9	18	Gateway-ID	Indicates the name of the gateway that processed the fax session. The name appears in the following format: hostname.domain-name.

Number	Vendor-Specific Company Code	Sub-Type Number	Attribute	Description
26	9	19	Call-Type	Describes the type of fax activity: fax receive or fax send.
26	9	20	Port-Used	Indicates the slot/port number of the Cisco AS5300 used to either transmit or receive this fax-mail.
26	9	21	Abort-Cause	If the fax session terminates, indicates the system component that signaled the termination. Examples of system components that could trigger an termination are FAP (Fax Application Process), TIFF (the TIFF reader or the TIFF writer), fax-mail client, fax-mail server, ESMTP client, or ESMTP server.
H323 Attributes				
26	9	23	Remote-Gateway-ID (h323-remote-address)	Indicates the IP address of the remote gateway.
26	9	24	Connection-ID (h323-conf-ID)	Identifies the conference ID.
26	9	25	Setup-Time (h323-setup-time)	Indicates the setup time for this connection in Coordinated Universal Time (UTC) formerly known as Greenwich Mean Time (GMT) and Zulu time.
26	9	26	Call-Origin (h323-call-origin)	Indicates the origin of the call relative to the gateway. Possible values are originating and terminating (answer).
26	9	27	Call-Type (h323-call-type)	Indicates call leg type. Possible values are **telephony** and **VoIP**.

Number	Vendor-Specific Company Code	Sub-Type Number	Attribute	Description
26	9	28	Connect-Time (h323-connect-time)	Indicates the connection time for this call leg in UTC.
26	9	29	Disconnect-Time (h323-disconnect-time)	Indicates the time this call leg was disconnected in UTC.
26	9	30	Disconnect-Cause (h323-disconnect-cause)	Specifies the reason a connection was taken offline per Q.931 specification.
26	9	31	Voice-Quality (h323-voice-quality)	Specifies the impairment factor (ICPIF) affecting voice quality for a call.
26	9	33	Gateway-ID (h323-gw-ID)	Indicates the name of the underlying gateway.
Large Scale Dialout Attributes				
26	9	1	callback-dialstring	Defines a dialing string to be used for callback.
26	9	1	data-service	No description available.
26	9	1	dial-number	Defines the number to dial.
26	9	1	force-56	Determines whether the network access server uses only the 56 K portion of a channel, even when all 64 K appear to be available.
26	9	1	map-class	Allows the user profile to reference information configured in a map class of the same name on the network access server that dials out.
26	9	1	send-auth	Defines the protocol to use (PAP or CHAP) for username-password authentication following CLID authentication.

Number	Vendor-Specific Company Code	Sub-Type Number	Attribute	Description
26	9	1	send-name	PPP name authentication. To apply for PAP, do not configure the **PPPPAPsent-namepassword** command on the interface. For PAP, “preauth:send-name” and “preauth:send-secret” will be used as the PAP username and PAP password for outbound authentication. For CHAP, “preauth:send-name” will be used not only for outbound authentication, but also for inbound authentication. For a CHAP inbound case, the NAS will use the name defined in “preauth:send-name” in the challenge packet to the caller box. **Note** The send-name attribute has changed over time: Initially, it performed the functions now provided by both the send-name and remote-name attributes. Because the remote-name attribute has been added, the send-name attribute is restricted to its current behavior.

Number	Vendor-Specific Company Code	Sub-Type Number	Attribute	Description
26	9	1	send-secret	PPP password authentication. The vendor-specific attributes (VSAs) “preauth:send-name” and “preauth:send-secret” will be used as the PAP username and PAP password for outbound authentication. For a CHAP outbound case, both “preauth:send-name” and “preauth:send-secret” will be used in the response packet.
26	9	1	remote-name	Provides the name of the remote host for use in large-scale dial-out. Dialer checks that the large-scale dial-out remote name matches the authenticated name, to protect against accidental user RADIUS misconfiguration. (For example, dialing a valid phone number but connecting to the wrong device.)
Miscellaneous Attributes				

Number	Vendor-Specific Company Code	Sub-Type Number	Attribute	Description
26	9	2	Cisco-NAS-Port	Specifies additional vendor specific attribute (VSA) information for NAS-Port accounting. To specify additional NAS-Port information in the form an Attribute-Value Pair (AVPair) string, use the **RADIUS-serverVSAsend** global configuration command. **Note** This VSA is typically used in Accounting, but may also be used in Authentication (Access-Request) packets.
26	9	1	min-links	Sets the minimum number of links for MLP.
26	9	1	proxyacl#<n>	Allows users to configure the downloadable user profiles (dynamic ACLs) by using the authentication proxy feature so that users can have the configured authorization to permit traffic going through the configured interfaces.

Number	Vendor-Specific Company Code	Sub-Type Number	Attribute	Description
26	9	1	SPI	Carries the authentication information needed by the home agent to authenticate a mobile node during registration. The information is in the same syntax as the **IPmobilesecurehost<addr>** configuration command. Basically it contains the rest of the configuration command that follows that string, verbatim. It provides the Security Parameter Index (SPI), key, authentication algorithm, authentication mode, and replay protection timestamp range.

Vendor-proprietary RADIUS server communications

A vendor-proprietary RADIUS server communication is a RADIUS server interaction method that

- enables devices and RADIUS servers to exchange vendor-specific information
- allows vendors to extend the RADIUS attribute set in unique ways, and
- is supported in Cisco IOS XE software for a subset of vendor-proprietary RADIUS attributes.

Configuration of vendor-proprietary RADIUS server communications

To configure RADIUS, whether vendor-proprietary or IETF draft-compliant, you must specify the host running the RADIUS server daemon and the secret text string it shares with the device.

You specify the RADIUS host and secret text string by using the **RADIUS server** global configuration commands.

DSCP markings for RADIUS packets

A DSCP marking for RADIUS packets is a Quality of Service (QoS) feature that

- classifies and manages traffic for preferential handling over other traffic classes
- uses the 6-bit differentiated services code point (DSCP) setting in IP packets to mark traffic classes with relative priorities, and

- allows faster authentication and accounting of RADIUS packets in Cisco IOS XE Software.

Configuration precedence for DSCP markings

DSCP marking can be configured on the RADIUS server, RADIUS server group, and in global configuration mode. The DSCP marking values entered on the RADIUS server take precedence.

- If there is no DSCP marking configuration on the RADIUS server, the DSCP marking values configured for the server group are applied to the RADIUS packets.
- If there is no DSCP marking configuration for the RADIUS server or RADIUS server group, the DSCP marking values configured in global configuration mode are applied to the RADIUS packets.

Restrictions for configuring RADIUS

General restrictions for RADIUS configuration

To prevent a lapse in security, you cannot configure RADIUS through a network management application.

- RADIUS and AAA servers can be configured to run only on the standard default ports:
  - 1812 and 1813
  - 1645 and 1646

Situations where RADIUS is not suitable

RADIUS is not suitable in the following network security situations:

- Multiprotocol access environments. RADIUS does not support AppleTalk Remote Access (ARA), NetBIOS Frame Control Protocol (NBFCP), NetWare Asynchronous Services Interface (NASI), or X.25 PAD connections.
- Switch-to-switch or router-to-router situations. RADIUS does not provide two-way authentication. RADIUS can be used to authenticate from one device to a non-Cisco device if the non-Cisco device requires authentication.
- Networks using a variety of services. RADIUS generally binds a user to one service model.

DSCP marking support for RADIUS packets

DSCP marking for authentication and accounting is not supported for private servers, fully qualified domain name (FQDN) servers and radsec servers.

- In the case of wired IEEE 802.1x authentication, when source port extension is not enabled, the default ports are in use. The DSCP marking is set to the default ports and all the requests will be marked with the same DSCP value.
- DSCP marking is not supported in the case of wireless IEEE 802.1x authentication, where the source port extension is enabled by default.

Prerequisites for configuring RADIUS

Prerequisites for RADIUS configuration

Recommendation: Enable RADIUS and Authentication, Authorization, and Accounting (AAA) to use any configuration commands. RADIUS is facilitated through AAA and can be enabled only through AAA commands.

- Use the **AAA new-model** global configuration command to enable AAA.
- Use the **AAA authentication** global configuration command to define method lists for RADIUS authentication.
- Use **line** and **interface** commands to enable the defined method lists to be used.
- At a minimum, identify the host or hosts that run the RADIUS server software and define the method lists for RADIUS authentication. Optionally, define method lists for RADIUS authorization and accounting.
- You should have access to and configure a RADIUS server before configuring RADIUS features on your device.
- The RADIUS host is normally a multiuser system running RADIUS server software from Cisco (Cisco ISE), Livingston, Merit, Microsoft, or another software provider. For more information, see the RADIUS server documentation.
- To use the Change-of-Authorization (CoA) interface, a session must already exist on the switch. CoA can be used to identify a session and enforce a disconnect request. The update affects only the specified session.

These prerequisites apply to all devices requiring RADIUS-based access control and AAA configuration.

Ensuring proper AAA and RADIUS configuration is necessary for secure device access and accurate authentication, authorization, and accounting.

Devices configured with these prerequisites will support secure access and allow for flexible authentication and authorization management.

Follow the listed prerequisites and configuration steps to resolve issues related to RADIUS access and AAA integration.

RADIUS operation requirements

Requirement: Users must successfully complete RADIUS authentication before proceeding to RADIUS authorization, if enabled.

- For RADIUS over IPv6 configurations, enable IPv6 unicast routing by using the **ipv6 unicast-routing** command.

These requirements apply to environments using RADIUS for device access and those implementing IPv6 configurations.

Proper authentication flow and IPv6 routing are essential for secure and reliable RADIUS operation.

Following these requirements ensures users are authenticated and authorized correctly, and that RADIUS operates as intended in IPv6 environments.

Enable the necessary commands and verify authentication steps to resolve operational issues with RADIUS.

RADIUS

Default RADIUS configurations

A default RADIUS configuration is a security setting that

- disables RADIUS and AAA by default
- prevents configuration of RADIUS through a network management application, and
- enables RADIUS to authenticate users accessing the device through the CLI when activated.

Default RADIUS configuration reference information

RADIUS and AAA are disabled by default.

To prevent a lapse in security, you cannot configure RADIUS through a network management application. When enabled, RADIUS can authenticate users accessing the device through the CLI.

Identify the RADIUS server host

This task enables you to configure a RADIUS server host on your device, allowing the device to communicate with the RADIUS server for authentication and accounting purposes.

Use this task when you need to set up or modify RADIUS server communication on your device. You can apply global settings to all RADIUS servers or configure per-server parameters to override global values.

To apply these settings globally to all RADIUS servers communicating with the device, use the three unique global configuration commands: **RADIUS-server timeout**, **RADIUS-server retransmit**, and **key***string*.

You can configure the device to use AAA server groups to group existing server hosts for authentication.

You also need to configure some settings on the RADIUS server. These settings include the IP address of the device and the key string to be shared by both the server and the device.

Before you begin

If you configure both global and per-server functions (timeout, retransmission, and key commands) on the device, the per-server timer, retransmission, and key value commands override global timer, retransmission, and key value commands.



Note RADIUS and AAA servers can be configured to run only on the standard default ports:

- 1812 and 1813
- 1645 and 1646

Follow these steps to identify and configure the RADIUS server host:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **RADIUS server** *server name* command to specify the name for the RADIUS server configuration for Protected Access Credential (PAC) provisioning, and enters RADIUS server configuration mode.

Example:

```
(config)# radius server rsim
```

Step 4 (Optional) Use the **address** {**ipv4 | ipv6**} *IP address* {**auth-port** *port number* | **acct-port***port number* } command to specify the RADIUS server parameters.

Example:

```
Switch(config-radius-server)# address ipv4 124.2.2.12 auth-port 1612
```

For **auth-port***port-number*, specify the UDP destination port for authentication requests. The default is 1645. The range is 0 to 65536.

For **acct-port***port-number*, specify the UDP destination port for accounting requests. The default is 1646.

Step 5 (Optional) Use the **key** *string* command to specify the authentication and encryption key used between the device and the RADIUS daemon running on the RADIUS server.

Example:

```
Switch(config-radius-server)# key rad123
```

Note
The key is a text string that must match the encryption key used on the RADIUS server. Always configure the key as the last item in the **RADIUS server** command. Leading spaces are ignored, but spaces within and at the end of the key are used. If you use spaces in your key, do not enclose the key in quotation marks unless the quotation marks are part of the key.

Step 6 (Optional) Use the **retransmit** *value* command to specify the number of times a RADIUS request is resent when the server is not responding or responding slowly

Example:

```
Switch(config-radius-server)# retransmit 10
```

The range is 1 to 100. This setting overrides the **RADIUS-server retransmit** global configuration command setting.

Step 7 Optional Use the **timeout** *seconds* command to specify the time interval that the device waits for the RADIUS server to reply before sending a request again.

Example:

```
Switch(config-radius-server)# timeout 60
```

The range is 1 to 1000. This setting overrides the **RADIUS-server timeout** global configuration command setting.

Step 8 Use the **end** command to exit global configuration mode and returns to privileged EXEC mode.

Example:

```
Switch(config)# end
```

After completing this task, the device is configured to communicate with the specified RADIUS server host using the defined parameters. Authentication and accounting requests will be sent to the configured server, and per-server settings will override global values as specified.

Configure RADIUS login authentication

This task enables you to secure device login by configuring RADIUS authentication using AAA methods. It ensures that only authorized users can access the device through HTTP or console by applying a RADIUS-based authentication list.

Use this procedure when you need to enforce centralized authentication for device logins, such as in environments requiring enhanced security or compliance with organizational policies. RADIUS authentication is commonly used in enterprise networks to manage user access.

Before you begin

To secure the device for HTTP access by using AAA methods, you must configure the **ip HTTP authentication AAA** global configuration command. By default, configuring AAA authentication does not secure the device for HTTP access by using AAA methods.

Follow these steps to configure RADIUS login authentication:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 USe the **AAA new-model** command to enable AAA.

Example:

```
Switch(config)# aaa new-model
```

Step 4 Use the **AAA authentication login** {**default** | *list-name*} *method1* [*method2...*] command to create a login authentication method list.

Example:

```
Switch(config)# aaa authentication login default local
```

- To create a default list that is used when a named list is *not* specified in the **login authentication** command, use the **default** keyword followed by the methods that are to be used in default situations. The default method list is automatically applied to all ports.
- For *list-name*, specify a character string to name the list you are creating.
- For *method1...*, specify the actual method the authentication algorithm tries. The additional methods of authentication are used only if the previous method returns an error, not if it fails.

  Select one of these methods:

  - *enable*—Use the enable password for authentication. Before you can use this authentication method, you must define an enable password by using the **enable** *password* global configuration command.
  - *group RADIUS*—Use RADIUS authentication. Before you can use this authentication method, you must configure the RADIUS server.
  - *line*—Use the line password for authentication. Before you can use this authentication method, you must define a line password. Use the **password** *password* line configuration command.
  - *local*—Use the local username database for authentication. You must enter username information in the database. Use the **username** *name* **password** global configuration command.
  - *local-case*—Use a case-sensitive local username database for authentication. You must enter username information in the database by using the **username** *password* global configuration command.
  - *none*—Do not use any authentication for login.

Step 5 Use the **line** [**console** | **tty** | **vty**] *line-number* [*ending-line-number*] command to enter line configuration mode, and configure the lines to which you want to apply the authentication list.

Example:

```
Switch(config)# line 1 4
```

Step 6 Use the **login authentication** {**default** | *list-name*} command to apply the authentication list to a line or set of lines.

Example:

```
Switch(config-line)# login authentication default
```

- If you specify **default**, use the default list created with the **AAA authentication login** command.
- For *list-name*, specify the list created with the **AAA authentication login** command.

Step 7 Use the **end** command to exit global configuration mode and returns to privileged EXEC mode.

Example:

```
Switch(config)# end
```

Define AAA server groups

This task enables you to configure AAA server groups by associating RADIUS servers with defined groups for authentication and accounting purposes.

You use the **server** group server configuration command to associate a particular server with a defined group server. You can either identify the server by its IP address or identify multiple host instances or entries by using the optional **auth-port** and **acct-port** keywords.

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **RADIUS server** *name* command to specify the name of the RADIUS server configuration for Protected Access Credential (PAC) provisioning and enters RADIUS server configuration mode.

Example:

```
Switch(config)# radius server ISESwitch
```

The device also supports RADIUS for IPv6.

Step 4 Use the **address** {**ipv4**|**ipv6**} {*IP-address*|*hostname*} **auth-port***port-number* **acct-port***port-number* command to configure the IPv4 address for the RADIUS server accounting and authentication parameters.

Example:

```
Switch(config-radius-server)# address ipv4 10.1.1.1 auth-port 1645 acct-port 1646
```

Step 5 Use the **key** *string* command to specify the authentication and encryption key for all RADIUS communications between the device and the RADIUS server.

Example:

```
Switch(config-radius-server)# key cisco123
```

Step 6 Use the **end** command to exit global configuration mode and returns to privileged EXEC mode.

Example:

```
Switch(config)# end
```

After completing these steps, the device is configured with the specified AAA server group and RADIUS server parameters for authentication and accounting.

Configure RADIUS authorization for user privileged access and network services

This task enables you to configure RADIUS authorization, which controls user access to privileged EXEC mode and network services on the device.

- Helps ensure that only authorized users can access sensitive device functions and network services.

Use this task when you need to restrict or manage user access to privileged EXEC mode and network services using RADIUS authorization.

Authorization is bypassed for authenticated users who log in through the CLI even if authorization has been configured.

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **aaa authorization network** *authorization-list* **RADIUS** command to configure the device for user RADIUS authorization for all network-related service requests.

Example:

```
Switch(config)# aaa authorization network list1 radius
```

Step 4 Use the **aaa authorization EXEC** *authorization-list* **RADIUS** command to configure the device for user RADIUS authorization if the user has privileged EXEC access.

Example:

```
Switch(config)# aaa authorization exec list1 radius
```

The **EXEC** keyword might return user profile information (such as **autocommand** information).

Step 5 Use the **end** command to exit global configuration mode and returns to privileged EXEC mode.

Example:

```
Switch(config)# end
```

After completing this task, RADIUS authorization is enabled for user privileged access and network services. Only users authorized by the RADIUS server will be able to access the specified services and privileged EXEC mode.

What to do next

You can use the **aaa authorization** global configuration command with the **RADIUS** keyword to set parameters that restrict a user's network access to privileged EXEC mode.

The **aaa authorization EXEC RADIUS local** command sets these authorization parameters:

- Use RADIUS for privileged EXEC access authorization if authentication was performed by using RADIUS.
- Use the local database if authentication was not performed by using RADIUS.

Start RADIUS accounting

This task enables RADIUS accounting on the device to track services accessed by users and to record network and EXEC process activities for auditing and monitoring purposes.

Use this procedure when you need to monitor and record user and network service access on your device using RADIUS accounting. This is typically required for compliance, auditing, or troubleshooting user activity.

Before you begin

No explicit prerequisites are stated. Ensure you have access to the device and appropriate privileges to configure RADIUS accounting.

Follow these steps to start RADIUS accounting:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **aaa accounting network** *accounting-list* **start-stop RADIUS** command to enable RADIUS accounting for all network-related service requests.

Example:

```
Switch(config)# aaa accounting network accounting-list start-stop radius
```

Step 4 Use the **aaa accounting EXEC** *accounting-list* **start-stop RADIUS** command to enable RADIUS accounting to send a start-record accounting notice at the beginning of a privileged EXEC process and a stop-record at the end.

Example:

```
Switch(config)# aaa accounting exec acc-list start-stop radius
```

Step 5 Use the **end** command to exit global configuration mode and returns to privileged EXEC mode.

Example:

```
Switch(config)# end
```

After completing these steps, RADIUS accounting is enabled on the device. The device will send accounting records for network and EXEC processes to the configured RADIUS server.

Configure settings for all RADIUS servers

This task enables you to configure global settings for all RADIUS servers, allowing the switch to communicate securely and efficiently with RADIUS servers for authentication purposes.

Use this procedure when you need to set or modify the global communication parameters for all RADIUS servers on your device. These settings affect how the switch interacts with RADIUS servers for authentication and authorization.

Before you begin

Beginning in privileged EXEC mode is required to perform these configuration steps.

Follow these steps to configure settings for all RADIUS servers:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **RADIUS server** *server name* command to specify the name for the RADIUS server configuration for Protected Access Credential (PAC) provisioning, and enters RADIUS server configuration mode.

Example:

```
Switch(config)# radius server rsim
```

Step 4 Use the **key** *string* command to specify the shared secret text string used between the switch and all RADIUS servers.

Example:

```
Switch(config-radius-server)# key your_server_key
```

Note
The key is a text string that must match the encryption key used on the RADIUS server. Leading spaces are ignored, but spaces within and at the end of the key are used. If you use spaces in your key, do not enclose the key in quotation marks unless the quotation marks are part of the key.

Step 5 Use the **retransmit** *retries* command to specify the number of times the switch sends each RADIUS request to the server before giving up.

Example:

```
Switch(config-radius-server)# retransmit 5
```

The default value is 3 and the range is 1 to 1000.

Step 6 Use the **timeout** *seconds* command to specify the number of seconds a switch waits for a reply to a RADIUS request before resending the request. The default is 5 seconds; the range is 1 to 1000.

Example:

```
Switch(config-radius-server)# timeout 3
```

Step 7 Use the **end** command to exit global configuration mode and returns to privileged EXEC mode.

Example:

```
Switch(config)# end
```

Configure vendor-specific RADIUS attributes

This task enables the device to recognize and use vendor-specific RADIUS attributes (VSAs) as defined by RADIUS IETF attribute 26. It allows you to configure the device to use accounting and/or authentication attributes for enhanced RADIUS server communication.

Vendor-specific RADIUS attributes are used to extend standard RADIUS functionality, allowing devices to support additional features required by specific vendors. This task is relevant when integrating devices with RADIUS servers that require VSAs for accounting or authentication.

Before you begin

- Ensure you have access to the device with sufficient privileges to enter configuration mode.

Follow these steps to configure vendor-specific RADIUS attributes:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **RADIUS-server vsa send** [**accounting** | **authentication**] command to enable the device to recognize and use VSAs as defined by RADIUS IETF attribute 26.

Example:

```
Switch(config)# radius-server vsa send accounting
```

- (Optional) Use the **accounting** keyword to limit the set of recognized vendor-specific attributes to only accounting attributes.
- (Optional) Use the **authentication** keyword to limit the set of recognized vendor-specific attributes to only authentication attributes.

If you enter this command without keywords, both accounting and authentication vendor-specific attributes are used.

Step 4 Use the **end** command to exit global configuration mode and returns to privileged EXEC mode.

Example:

```
Switch(config)# end
```

After completing these steps, the device will recognize and use vendor-specific RADIUS attributes for accounting and/or authentication, as configured. This ensures proper communication with RADIUS servers requiring VSAs.

Configure the device for vendor-proprietary RADIUS server communication

This task enables the device to communicate with a vendor-proprietary RADIUS server for authentication and access control. It configures the necessary settings to support Protected Access Credential (PAC) provisioning and secure password exchange.

- Allows integration with custom RADIUS implementations.
- Ensures secure communication between the device and the RADIUS server.

Use this task when you need to configure a device to work with a vendor-proprietary RADIUS server, such as for environments requiring custom authentication protocols or PAC provisioning.

Perform this configuration when deploying devices that must authenticate users or network access through a RADIUS server using a non-standard implementation.

- Applicable for network setups requiring enhanced security or custom authentication methods.

Before you begin

Ensure you have access to the device with privileged EXEC mode and know the vendor-proprietary RADIUS server details, including IP address and shared secret.

- Obtain the server name, IP address, and shared secret string from your RADIUS server administrator.

Follow these steps to configure the device for vendor-proprietary RADIUS server communication:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **RADIUS server** *server name* command to specify the name for the RADIUS server configuration for Protected Access Credential (PAC) provisioning, and enters RADIUS server configuration mode.

Example:

```
Switch(config)# radius server rsim
```

Step 4 (Optional) Use the **address** { **ipv4 | ipv6** } *IP address* command to specify the IP address of the RADIUS server.

Example:

```
Switch(config-radius-server)# address ipv4 172.24.25.10
```

Step 5 Use the **non-standard** command to identify that the RADIUS server is using a vendor-proprietary implementation of RADIUS.

Example:

```
Switch(config-radius-server)# non-standard
```

Step 6 Use the **key***string* command to specify the shared secret text string used between the device and the vendor-proprietary RADIUS server. The device and the RADIUS server use this text string to encrypt passwords and exchange responses.

Example:

```
Switch(config-radius-server)# key rad123
```

Step 7 Use the **end** command to exit global configuration mode and returns to privileged EXEC mode.

Example:

```
Switch(config)# end
```

After completing these steps, the device is configured to communicate securely with the vendor-proprietary RADIUS server. Authentication and access control will use the specified server and shared secret.

What to do next

Verify the device can successfully authenticate with the RADIUS server and monitor logs for any errors or failed authentication attempts.

- Test connectivity and authentication to ensure proper configuration.

Configure DSCP marking on a RADIUS server

This task enables you to configure DSCP marking for authentication and accounting on a RADIUS server, allowing you to prioritize RADIUS traffic in your network.

Use this task when you need to set differentiated services code point (DSCP) values for RADIUS authentication and accounting traffic, ensuring that these packets receive appropriate priority in your network.

Before you begin

No specific prerequisites are required for this task.

Follow these steps to configure DSCP marking for authentication and accounting on a RADIUS server:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **RADIUS server** *server_name* command to specify the name for the RADIUS server configuration for Protected Access Credential (PAC) provisioning, and enters RADIUS server configuration mode.

Example:

```
Switch(config)# radius server rsim
```

Step 4 (Optional) Use the **address** { **ipv4 | ipv6** } *IP address* [**auth-port** *auth_port_number* **acct-port** *acct_port_number*] command to specify the IP address of the RADIUS server.

Example:

```
Switch(config-radius-server)# address ipv4 10.1.1.1 auth-port 1645 acct-port 1646
```

- **auth-port** configures the port value for RADIUS authentication server. The default value is 1812.
- **acct-port** configures the port value for RADIUS accounting server. The default value is 1813.

Step 5 Use the **DSCP** {**acct** *dscp_acct_value* | **auth** *dscp_auth_value*} command to configure DSCP marking for authentication and accounting on the RADIUS server.

Example:

```
Switch(config-radius-server)# dscp auth 10 acct 20
```

- **acct** configures RADIUS DSCP marking value for accounting. The valid range is from 1 to 63. The default value is 0.
- **auth** configures RADIUS DSCP marking value for authentication. The valid range is from 1 to 63. The default value is 0.

Step 6 Use the **key** *string* command to specify the shared secret text string used between the device and the vendor-proprietary RADIUS server.

Example:

```
Switch(config-radius-server)# key rad123
```

The device and the RADIUS server use this text string to encrypt passwords and exchange responses.

Step 7 Use the **end** command to exit global configuration mode and returns to privileged EXEC mode.

Example:

```
Switch(config)# end
```

After completing this task, DSCP marking is configured for authentication and accounting on the RADIUS server. RADIUS traffic will be prioritized according to the specified DSCP values.

Configure the source interface and DSCP marking on a RADIUS server group

This task enables you to configure the source interface and DSCP marking for authentication and accounting on a RADIUS server group. This configuration helps control the source address and DSCP values used for RADIUS authentication and accounting traffic.

Use this task when you need to specify a particular source interface and set DSCP marking values for RADIUS authentication and accounting messages sent from your device to the RADIUS server group. This is typically required for network environments where traffic marking and interface selection are important for security or traffic engineering.

Before you begin

Before you begin, ensure you have access to the device CLI with sufficient privileges to configure RADIUS server groups.

Follow these steps to configure the source interface and DSCP marking for authentication and accounting on a RADIUS server group:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **aaa group server RADIUS** *group_name* command to define the RADIUS server group configuration and enters RADIUS server group configuration mode.

Example:

```
Switch(config)# aaa group server radius abc
```

Step 4 Use the **server name** *name* command to associates the RADIUS server to the server group.

Example:

```
Switch(config-sg-radius)# server name serv1
```

Step 5 Use the {**ip** | **ipv6**} **RADIUS source-interface** *type number* command to specify an interface to use for the source address in RADIUS server.

Example:

```
Switch(config-sg-radius)# ipv6 radius source-interface ethernet 0/0
```

Step 6 Use the **DSCP** {**acct** *dscp_acct_value* | **auth** *dscp_auth_value*} command to configure DSCP marking for authentication and accounting on the RADIUS server group.

Example:

```
Switch(config-sg-radius)# dscp auth 10 acct 20Switch
```

- **acct** configures RADIUS DSCP marking value for accounting. The valid range is from 1 to 63. The default value is 0.
- **auth** configures RADIUS DSCP marking value for authentication. The valid range is from 1 to 63. The default value is 0.

Step 7 Use the **end** command to exit global configuration mode and returns to privileged EXEC mode.

Example:

```
Switch(config)# end
```

After completing these steps, the device uses the specified source interface and DSCP marking values for RADIUS authentication and accounting traffic sent to the configured RADIUS server group.

Configure CoA

This task enables you to configure Change of Authorization (CoA) on a Switch, allowing it to interact with an external policy server and process dynamic authorization requests.

Use this procedure when you need the Switch to accept CoA and disconnect requests from RADIUS clients. This is typically required in environments where dynamic policy enforcement is necessary.

Before you begin

This procedure is required before the Switch can process CoA requests from RADIUS clients.

Follow these steps to configure CoA on a Switch:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Configure AAA and RADIUS

a) Use the **AAA new-model** command to enable AAA.

Example:

```
Switch(config)# aaa new-model
```

b) Use the **AAA server RADIUS dynamic-author** command to configure the Switch as an authentication, authorization, and accounting (AAA) server to facilitate interaction with an external policy server, and enters dynamic authorization local server configuration mode.

Example:

```
Switch(config)# aaa server radius dynamic-author
```

c) Use the **client** {*ip-address* | *name*} [**vrf** *vrfname*] [**server-key** *string*] command to specify a RADIUS client from which a Switch will accept CoA and disconnect requests.

Example:

```
Switch(config-locsvr-da-radius)# client client1 vrf vrf1
```

d) Use the **server-key** [**0** | **7**] *string* command to configure the RADIUS key to be shared between a Switch and RADIUS clients.

Example:

```
Switch(config-locsvr-da-radius)# server-key your_server_key
```

e) Use the **port** *port-number* command to specify the port on which a switch listens for RADIUS requests from configured RADIUS clients.

Example:

```
Switch(config-locsvr-da-radius)# port 25
```

f) Use the **auth-type** {**any** | **all** | **session-key**} command to specify the type of authorization the Switch uses for RADIUS clients.

Example:

```
Switch(config-locsvr-da-radius)# auth-type any
```

The client must match all the configured attributes for authorization.

g) (Optional) Use the **ignore server-key** command to configure the switch to ignore the server-key.

Example:

```
Switch(config-locsvr-da-radius)# ignore server-key
Switch(config-locsvr-da-radius)# exit
```

exit command exits from the dynamic authorization local server configuration mode and returns to global configuration mode.

h) (Optional) Use the **authentication command bounce-port ignore** command to configure the switch to ignore a CoA request to temporarily disable the port hosting a session at the global configuration mode.

Example:

```
Switch(config)# authentication command bounce-port ignore
```

The purpose of temporarily disabling the port is to trigger a DHCP renegotiation from the host when a VLAN change occurs and there is no supplicant on the endpoint to detect the change.

i) (Optional) Use the **authentication command disable-port ignore** command to configure the switch to ignore a nonstandard command requesting that the port hosting a session be administratively shut down

Example:

```
Switch(config)# authentication command disable-port ignore
```

Shutting down the port results in termination of the session. Use standard CLI or SNMP commands to re-enable the port.

Monitor CoA

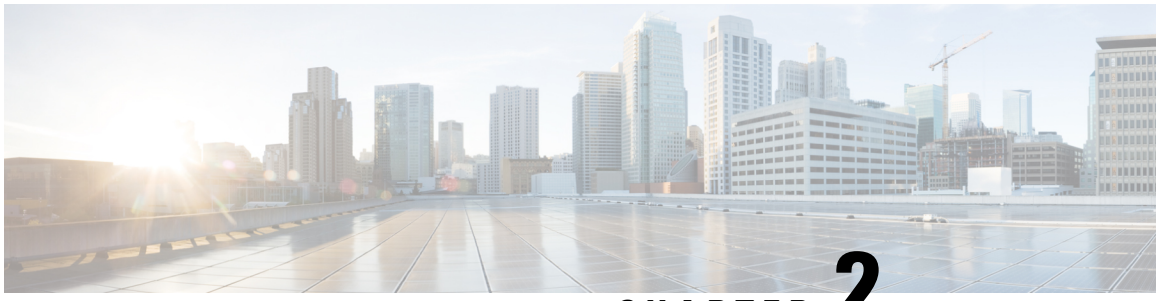
This reference lists commands used to monitor and troubleshoot Change of Authorization (CoA) functionality, including their purposes.

Table 7: Privileged EXEC show commands

Command	Purpose
show AAA attributes protocol RADIUS	Displays AAA attributes of RADIUS commands.

Table 8: Global troubleshooting commands

Command	Purpose
debug RADIUS	Displays information for troubleshooting RADIUS.
debug AAA coa	Displays information for troubleshooting CoA processing.
debug AAA POD	Displays information for troubleshooting POD packets.
debug AAA subsys	Displays information for troubleshooting POD packets.



CHAPTER 2

MACsec Encryption

- MACsec Encryption, on page 45
- MACsec Key Agreement, on page 46
- MACsec MKA using Certificate-based MACsec, on page 50
- Switch-to-Switch MKA MACsec Must Secure Policy, on page 51
- MKA/MACsec for Port Channel, on page 51
- MACsec Cipher Announcement, on page 51
- How to Configure MACsec Encryption, on page 52
- Additional References for MACsec Encryption, on page 95
- Feature History for MACsec Encryption, on page 96
- Overview of PQC for MACsec with EAP-TLS, on page 96
- Key elements of PQC, on page 97
- PQC implementation in EAP-TLS, on page 98
- Prerequisites, on page 100
- Restrictions, on page 100
- Configuration matrix for MACsec with EAP-TLS, on page 100
- Workflow to set up PQC for MACsec with EAP-TLS, on page 101
- Configure EAP-TLS 1.3, on page 101
- Configure PQC for EAP-TLS sessions, on page 102
- Verify PQC for EAP-TLS sessions, on page 102

MACsec Encryption

MACsec is the IEEE 802.1AE standard for authenticating and encrypting packets between two MACsec-capable devices. Catalyst switches support 802.1AE encryption with MACsec Key Agreement (MKA) on switch-to-host links for encryption between the switch and host device. The switch also supports MACsec encryption for switch-to-switch (inter-network device) security using MKA-based key exchange protocol.

Note When switch-to-switch MACSec is enabled, all traffic is encrypted, except the EAP-over-LAN (EAPOL) packets.

Link layer security can include both packet authentication between switches and MACsec encryption between switches (encryption is optional).

Table 9: MACsec Support on Switch Ports

Connections	MACsec support
Switch-to-host	MACsec MKA encryption
Switch-to-switch	MACsec MKA encryption

Cisco TrustSec is meant only for switch-to-switch links and is not supported on switch ports connected to end hosts, such as PCs or IP phones. MKA is supported on switch-to-host facing links as well as switch-to-switch links. Host-facing links typically use flexible authentication ordering for handling heterogeneous devices with or without IEEE 802.1x, and can optionally use MKA-based MACsec encryption. Network Edge Access Topology (NEAT) is used for compact switches to extend security outside the wiring closet.

MACsec Key Agreement

MACsec, defined in 802.1AE, provides MAC-layer encryption over wired networks by using out-of-band methods for encryption keying. The MACsec Key Agreement (MKA) Protocol provides the required session keys and manages the required encryption keys. MKA and MACsec are implemented after successful authentication using certificate-based MACsec or Pre Shared Key (PSK) framework.

A switch using MACsec accepts either MACsec or non-MACsec frames, depending on the policy associated with the MKA peer. MACsec frames are encrypted and protected with an integrity check value (ICV). When the switch receives frames from the MKA peer, it decrypts them and calculates the correct ICV by using session keys provided by MKA. The switch compares that ICV to the ICV within the frame. If they are not identical, the frame is dropped. The switch also encrypts and adds an ICV to any frames sent over the secured port (the access point used to provide the secure MAC service to a MKA peer) using the current session key.

The MKA Protocol manages the encryption keys used by the underlying MACsec protocol. The basic requirements of MKA are defined in 802.1x-REV. The MKA Protocol extends 802.1x to allow peer discovery with confirmation of mutual authentication and sharing of MACsec secret keys to protect data exchanged by the peers.

The EAP framework implements MKA as a newly defined EAP-over-LAN (EAPOL) packet. EAP authentication produces a master session key (MSK) shared by both partners in the data exchange. Entering the EAP session ID generates a secure connectivity association key name (CKN). The switch acts as the authenticator for both uplink and downlink; and acts as the key server for downlink. It generates a random secure association key (SAK), which is sent to the client partner. The client is never a key server and can only interact with a single MKA entity, the key server. After key derivation and generation, the switch sends periodic transports to the partner at a default interval of 2 seconds.

The packet body in an EAPOL Protocol Data Unit (PDU) is referred to as a MACsec Key Agreement PDU (MKPDU). MKA sessions and participants are deleted when the MKA lifetime (6 seconds) passes with no MKPDU received from a participant. For example, if a MKA peer disconnects, the participant on the switch continues to operate MKA until 6 seconds have elapsed after the last MKPDU is received from the MKA peer.

Note Integrity check value (ICV) indicator in MKPDU is optional. ICV is not optional when the traffic is encrypted.

EAPoL Announcements indicate the use of the type of keying material. The announcements can be used to announce the capability of the supplicant as well as the authenticator. Based on the capability of each side, the largest common denominator of the keying material could be used.

MKA Policies

To enable MKA on an interface, a defined MKA policy should be applied to the interface. You can configure these options:

- Policy name, not to exceed 16 ASCII characters.
- Confidentiality (encryption) offset of 0, 30, or 50 bytes for each physical interface

Definition of Policy-Map Actions

This section describes the policy-map actions and its definition:

- Activate: Applies a service template to the session.
- Authenticate: Starts authentication of the session.
- Authorize: Explicitly authorizes a session.
- Set-domain: Explicitly sets the domain of a client.
- Terminate: Terminates the method that is running, and deletes all the method details associated with the session.
- Deactivate: Removes the service-template applied to the session. If not applied, no action is taken.
- Set-timer: Starts a timer and gets associated with the session. When the timer expires, any action that needs to be started can be processed.
- Authentication-restart: Restarts authentication.
- Clear-session: Deletes a session.
- Pause: Pauses authentication.

Rest of the actions as self-explanatory and are associated with authentication.

Virtual Ports

Use virtual ports for multiple secured connectivity associations on a single physical port. Each connectivity association (pair) represents a virtual port. In uplink, you can have only one virtual port per physical port. You cannot simultaneously host secured and unsecured sessions in the same VLAN on the same port. Because of this limitation, 802.1x multiple authentication mode is not supported.

The exception to this limitation is in multiple-host mode when the first MACsec supplicant is successfully authenticated and connected to a hub that is connected to the switch. A non-MACsec host connected to the hub can send traffic without authentication because it is in multiple-host mode. We do not recommend using multi-host mode because after the first successful client, authentication is not required for other clients.

Virtual ports represent an arbitrary identifier for a connectivity association and have no meaning outside the MKA Protocol. A virtual port corresponds to a separate logical port ID. Valid port IDs for a virtual port are

0x0002 to 0xFFFF. Each virtual port receives a unique secure channel identifier (SCI) based on the MAC address of the physical interface concatenated with a 16-bit port ID.

MKA Statistics

Some MKA counters are aggregated globally, while others are updated both globally and per session. You can also obtain information about the status of MKA sessions. See Displaying MKA Statistics for further information.

Key Lifetime and Hitless Key Rollover

A MACsec key chain can have multiple pre-shared keys (PSK) each configured with a key id and an optional lifetime. A key lifetime specifies at which time the key expires. In the absence of a lifetime configuration, the default lifetime is unlimited. When a lifetime is configured, MKA rolls over to the next configured pre-shared key in the key chain after the lifetime is expired. Time zone of the key can be local or UTC. Default time zone is UTC.

You can Key rolls over to the next key within the same key chain by configuring a second key in the key chain and configuring a lifetime for the first key. When the lifetime of the first key expires, it automatically rolls over to the next key in the list. If the same key is configured on both sides of the link at the same time, then the key rollover is hitless, that is, key rolls over without traffic interruption.

On all participating devices, the MACsec key chain must be synchronised by using Network Time Protocol (NTP) and the same time zone must be used. If all the participating devices are not synchronized, the connectivity association key (CAK) rekey will not be initiated on all the devices at the same time.

Note The lifetime of the keys need to be overlapped in order to achieve hitless key rollover.

Default timer-based SAK rekey

A Secure Association Key (SAK) is a core security feature in MACsec (Media Access Control Security) that:

- delivers encryption and authentication for Ethernet frames, ensuring secure data transmission, and
- is dynamically created and managed as secure associations are established between network devices.

Cisco Catalyst IE9300 Rugged Series Switches support default timer-based rekey, with rekey frequencies based on interface speed:

- 1 Gbps: approximately 30 minutes
- 10 Gbps: approximately 3 minutes
- 40 Gbps: approximately 45 seconds

Note The Cisco Catalyst IE9300 Rugged Series Switches doesn't support packet numbering (PN) exhaustion rekey.

Replay Protection Window Size

Replay protection is a feature provided by MACsec to counter replay attacks. Each encrypted packet is assigned a unique sequence number and the sequence is verified at the remote end. Frames transmitted through a Metro Ethernet service provider network are highly susceptible to reordering due to prioritization and load balancing mechanisms used within the network.

A replay window is necessary to support the use of MACsec over provider networks that reorder frames. Frames within the window can be received out of order, but are not replay protected. The default window size is 0, which enforces strict reception ordering. The replay window size can be configured in the range of 0 to 2^{32}- 1.

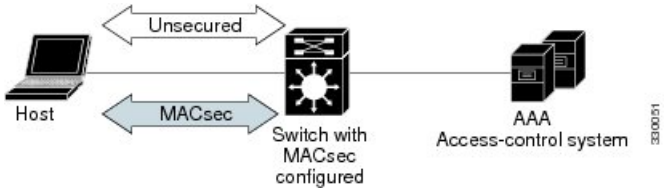
MACsec, MKA and 802.1x Host Modes

You can use MACsec and the MKA Protocol with 802.1x single-host mode, multi-host mode, or Multi Domain Authentication (MDA) mode. Multiple authentication mode is not supported.

Single-Host Mode

The figure shows how a single EAP authenticated session is secured by MACsec by using MKA

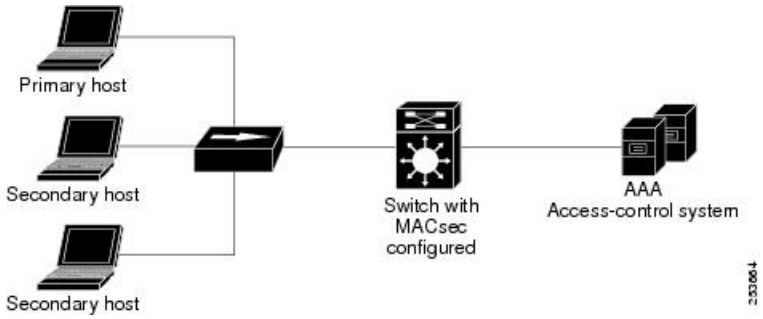
Figure 3: MACsec in Single-Host Mode with a Secured Data Session



Multiple-Host Mode

In standard (not 802.1x REV) 802.1x multiple-host mode, a port is open or closed based on a single authentication. If one user, the primary secured client services client host, is authenticated, the same level of network access is provided to any host connected to the same port. If a secondary host is a MACsec supplicant, it cannot be authenticated and traffic would not flow. A secondary host that is a non-MACsec host can send traffic to the network without authentication because it is in multiple-host mode. The figure shows MACsec in Standard Multiple-Host Unsecure Mode.

Figure 4: MACsec in Multiple-Host Mode - Unsecured



Note Multi-host mode is not recommended because after the first successful client, authentication is not required for other clients, which is not secure.

In standard (not 802.1x REV) 802.1x multiple-domain mode, a port is open or closed based on a single authentication. If the primary user, a PC on data domain, is authenticated, the same level of network access is provided to any domain connected to the same port. If a secondary user is a MACsec supplicant, it cannot be authenticated and traffic would no flow. A secondary user, an IP phone on voice domain, that is a non-MACsec host, can send traffic to the network without authentication because it is in multiple-domain mode.

Multiple-Domain Mode

In standard (not 802.1x REV) 802.1x multiple-domain mode, a port is open or closed based on a single authentication. If the primary user, a PC on data domain, is authenticated, the same level of network access is provided to any domain connected to the same port. If a secondary user is a MACsec supplicant, it cannot be authenticated and traffic would no flow. A secondary user, an IP phone on voice domain, that is a non-MACsec host, can send traffic to the network without authentication because it is in multiple-domain mode.

MACsec MKA using Certificate-based MACsec

MACsec MKA is supported on switch-to-switch links. Using certificate-based MACsec, you can configure MACsec MKA between device uplink ports. Certificate-based MACsec allows mutual authentication and obtains an MSK (master session key) from which the connectivity association key (CAK) is derived for MKA operations. Device certificates are carried, using certificate-based MACsec, for authentication to the AAA server.

Note Certificate-based MACsec is supported for Cisco Catalyst ESS9300 Embedded Series Switch beginning with the Cisco IOS XE 17.13.1 release.

Prerequisites for MACsec MKA Using Certificate-based MACsec

- Ensure that you have a Certificate Authority (CA) server configured for your network.
- Generate a CA certificate.
- Ensure that you have configured Cisco Identity Services Engine (ISE) Release 2.0.
- Ensure that both the participating devices, the CA server, and Cisco Identity Services Engine (ISE) are synchronized using Network Time Protocol (NTP). If time is not synchronized on all your devices, certificates will not be validated.
- Ensure that 802.1x authentication and AAA are configured on your device.

Switch-to-Switch MKA MACsec Must Secure Policy

Must-secure support is enabled on both the ingress and the egress. Must-secure is supported for MKA. With must-secure enabled, only EAPoL traffic will not be encrypted. The rest of the traffic will be encrypted. Unencrypted packets are dropped.



Note Must-secure mode is enabled by default.

MKA/MACsec for Port Channel

MKA/MACsec can be configured on the port members of a port channel. MKA/MACsec is agnostic to the port channel since the MKA session is established between the port members of a port channel.



Note Port channel is supported for PSK-based MACsec but not for certificate-based MACsec.



Note EtherChannel links that are formed as part of the port channel can either be congruent or disparate. That is, the links can either be MACsec-secured or non-MACsec-secured. MKA session between the port members is established even if a port member on one side of the port channel is not configured with MACsec.

We recommend that you enable MKA/MACsec on all the member ports for better security of the port channel.

MACsec Cipher Announcement

Cipher Announcement allows the supplicant and the authenticator to announce their respective MACsec Cipher Suite capabilities to each other. Both the supplicant and the authenticator calculate the largest common supported MACsec Cipher Suite and use the same as the keying material for the MKA session.



Note Only the MACsec Cipher Suite capabilities which are configured in the MKA policy are announced from the authenticator to the supplicant.

There are two types of EAPoL Announcements:

- Unsecured Announcements (EAPoL PDUs) : Unsecured announcements are EAPoL announcements carrying MACsec Cipher Suite capabilities in an unsecured manner. These announcements are used to decide the width of the key used for MKA session prior to authentication.
- Secure Announcements (MKPDUs) : Secure announcements revalidate the MACsec Cipher Suite capabilities which were shared previously through unsecure announcements.

Once the session is authenticated, peer capabilities which were received through EAPoL announcements are revalidated with the secure announcements. If there is a mismatch in the capabilities, the MKA session tears down.

Limitations for MACsec Cipher Announcement

- MACsec Cipher Announcement is supported only on the switch-to-host links.
- The MKA session between the supplicant and the authenticator does not tear down even if the MACsec Cipher Suite capabilities configured on both do not result in a common cipher suite.
- Host to Switch MKA MACsec 256-bit Cipher is not supported on both Network Advantage and Network Essentials.

How to Configure MACsec Encryption

Prerequisites for MACsec Encryption

Prerequisites for MACsec Encryption

- Enable the **ssci-based-on-sci** command while configuring MACsec encryption on the device to allow interoperability with non-Cisco and non-IOS XE devices.
- Ensure that 802.1x authentication and AAA are configured on your device.

Prerequisites for Certificate-Based MACsec

- Ensure that you have a Certificate Authority (CA) server configured for your network.
- Generate a CA certificate.
- Ensure that you have configured Cisco Identity Services Engine (ISE) Release 2.0.
- Ensure that both the participating devices, the CA server, and Cisco Identity Services Engine (ISE) are synchronized using Network Time Protocol (NTP). If time is not synchronized on all your devices, certificates will not be validated.

Restrictions for MACsec Encryption

- MACsec Key Agreement (MKA) is not supported with both stateful and stateless high availability.
- MACsec with MKA is supported only on point-to-point links.
- MACsec configuration is not supported on EtherChannel ports. Instead, MACsec configuration can be applied on the individual member ports of an EtherChannel. To remove MACsec configuration, you must first unbundle the member ports from the EtherChannel, and then remove it from the individual member ports.

- Cisco Catalyst IE9300 Rugged Series Switches support 128-bit MACsec encryption with a Network Essentials license and 256-bit MACsec encryption with a Network Advantage license.
- Certificate-based MACsec is supported only if the access-session is configured as closed or in multiple-host mode. None of the other configuration modes are supported.
- Packet number exhaustion rekey is not supported.
- If the **dot1q tag vlan native** command is configured globally, the dot1x reauthentication will fail on trunk ports.
- MACsec with Precision Time Protocol (PTP) is not supported.
- The **should-secure** access mode is supported on switch-to-switch ports only using PSK authentication.
- PSK fallback key chain is not supported for point-to-multipoint cases.
- PSK fallback key chain is not supported on a high availability setup.
- PSK fallback key chain supports infinite lifetime with one key only.
- The connectivity association key name (CKN) ID used in the fallback key chain must not match any of the CKN IDs used in the primary key chain.
- The following limitations apply only to certificate-based MACsec.
  - The port should be in access mode or trunk mode.
  - MKA is not supported on port channels.
  - Ports with no switch port are not supported.

Recommendations for MACsec Encryption

This section list the recommendations for configuring MACsec encryption:

- Use the confidentiality (encryption) offset as 0 in switch-to-host connections.
- Execute the **shutdown** command, and then the **no shutdown** command on a port, after changing any MKA policy or MACsec configuration for active sessions, so that the changes are applied to active sessions.
- Set the connectivity association key (CAK) rekey overlap timer to 30 seconds or more.

MKA and MACsec Configuration

MACsec is disabled by default. No MKA policies are configured.

Configure an MKA Policy

Perform these steps to configure MACsec Key Agreement (MKA) policy.

Procedure

Step 1 Accessing and creating the MKA policy

a) Use the **enable** command to enable privileged EXEC mode.

Example:

```
Device> enable
```

Enter your password if prompted.

b) Use the **configure terminal** command to enter global configuration mode.

Example:

```
Device# configure terminal
```

c) Use the **mka policy** *policy-name* command to identifies an MKA policy, and enters into the MKA policy configuration mode.

Example:

```
Device(config)# mka policy mka_policy
```

The maximum policy name length is 16 characters.

Note
The default MACsec cipher suite in the MKA policy will always be "GCM-AES-128". If the device supports both "GCM-AES-128" and "GCM-AES-256" ciphers, it is highly recommended to define and use a user defined MKA policy to include both 128 and 256 bits ciphers or only 256 bits cipher, as may be required.

Step 2 Configuring MKA policy parameters

a) Use the **key-server** command to configure MKA key server options.

Example:

```
Device(config-mka-policy)# key-server priority 200
```

It sets priority between 0-255.

Note
When value of key server priority is set to 255, the peer can not become the key server. The key server priority value is valid only for MKA PSK; and not for MKA EAPTLS.

b) Use **sak-rekey interval** *time-interval* to configure the SAK rekey interval in seconds.

Example:

```
Device(config-mka-policy)# sak-rekey interval 50
```

This configuration is optional, with a range from 30 to 65,535.

By default, the SAK rekey interval is set automatically based on interface speed. Any user-defined value must be less than the default interval.

Use the **no** form of this command to stop the SAK rekey timer.

c) Use the **include-icv-indicator** command to enables the ICV indicator in MKPDU.

Example:

```
Device(config-mka-policy)# include-icv-indicator
```

Use the **no** form of this command to disable the ICV indicator.

d) **macsec-cipher-suite** co mand to configure a cipher suite for deriving SAK with 128-bit or 256-bit encryption.

Example:

```
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128
```

The syntax includes: **macsec-cipher-suite** {*gcm-aes-128* | *gcm-aes-256*}

e) Use the **confidentiality-offset** command to set the confidentiality (encryption) offset for each physical interface.

Example:

```
Device(config-mka-policy)# confidentiality-offset 0
```

Note
Offset Value can be 0, 30 or 50. If you are using Anyconnect on the client, it is recommended to use Offset 0.

f) Use the **ssci-based-on-sci** command to computes Short Secure Channel Identifier (SSCI) value.

Example:

```
Device(config-mka-policy)# ssci-based-on-sci
```

(Optional) SSCI computation value is based on Secure Channel Identifier (SCI) value. The higher the SCI value, the lower is the SSCI value.

g) Use the **end** command to exit MKA policy configuration mode and return to privileged EXEC mode.

Example:

```
Device(config-mka-policy)# end
```

h) (Optional) Use the **show mka policy** command to display MKA policy configuration information.

Example:

```
Device# show mka policy
```

Configure Switch-to-host MACsec Encryption

Follow these steps to configure MACsec on an interface with one MACsec session for voice and one for data:

SUMMARY STEPS

1. **enable**
2. **configureterminal**

3. **interface** *type number*
4. **switchport access vlan***vlan-id*
5. **switchport mode access**
6. **macsec**
7. **authentication event linksec fail action authorize vlan** *vlan-id*
8. **authentication host-mode multi-domain**
9. **authentication linksec policy must-secure**
10. **authentication port-control auto**
11. **authentication periodic**
12. **authentication timer reauthenticate**
13. **authentication violation protect**
14. **mka policy** *policy-name*
15. **dot1x pae authenticator**
16. **spanning-tree portfast**
17. **end**
18. **show authentication session interface** *interface-id*
19. **show mka sessions**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter the password if prompted.
Step 2	**configureterminal** **Example:** `Device>configure terminal`	Enters the global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 1/0/1`	Identifies the MACsec interface, and enters interface configuration mode. The interface must be a physical interface.
Step 4	**switchport access vlan***vlan-id* **Example:** `Device(config-if)# switchport access vlan 1`	Configures the access VLAN for the port.
Step 5	**switchport mode access** **Example:** `Device(config-if)# switchport mode access`	Configures the interface as an access port.

	Command or Action	Purpose
Step 6	**macsec** **Example:** `Device(config-if)# macsec`	Enables 802.1ae MACsec on the interface. The macsec command enables MKA MACsec on switch-to-host links only.
Step 7	**authentication event linksec fail action authorize vlan** *vlan-id* **Example:** `Device(config-if)# authentication event linksec fail action authorize vlan 1`	(Optional) Specifies that the switch processes authentication link-security failures resulting from unrecognized user credentials by authorizing a restricted VLAN on the port after a failed authentication attempt.
Step 8	**authentication host-mode multi-domain** **Example:** `Device(config-if)# authentication host-mode multi-domain`	Configures authentication manager mode on the port to allow both a host and a voice device to be authenticated on the 802.1x-authorized port. If not configured, the default host mode is single.
Step 9	**authentication linksec policy must-secure** **Example:** `Device(config-if)# authentication linksec policy must-secure`	Sets the LinkSec security policy to secure the session with MACsec if the peer is available. If not set, the default is *should secure*.
Step 10	**authentication port-control auto** **Example:** `Device(config-if)# authentication port-control auto`	Enables 802.1x authentication on the port. The port changes to the authorized or unauthorized state based on the authentication exchange between the switch and the client.
Step 11	**authentication periodic** **Example:** `Device(config-if)# authentication periodic`	(Optional) Enables or disables re-authentication for this port .
Step 12	**authentication timer reauthenticate** **Example:** `Device(config-if)# authentication timer reauthenticate`	(Optional) Enters a value between 1 and 65535 (in seconds). Obtains re-authentication timeout value from the server. Default re-authentication time is 3600 seconds.
Step 13	**authentication violation protect** **Example:** `Device(config-if)# configure terminal`	Configures the port to drop unexpected incoming MAC addresses when a new device connects to a port or when a device connects to a port after the maximum number of devices are connected to that port. If not configured, the default is to shut down the port.
Step 14	**mka policy** *policy-name* **Example:** `Device(config-if)# mka policy mka_policy`	Applies an existing MKA protocol policy to the interface, and enable MKA on the interface. If no MKA policy was configured (by entering the**mka policy** global configuration command).
Step 15	**dot1x pae authenticator** **Example:**	Configures the port as an 802.1x port access entity (PAE) authenticator.

	Command or Action	Purpose
	`Device(config-if)# dot1x pae authenticator`	
Step 16	**spanning-tree portfast** **Example:** `Device(config-if)# spanning-tree portfast`	Enables spanning tree Port Fast on the interface in all its associated VLANs. When the Port Fast feature is enabled, the interface changes directly from a blocking state to a forwarding state without making the intermediate spanning-tree state changes
Step 17	**end** **Example:** `Device(config)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 18	**show authentication session interface** *interface-id* **Example:** `Device# show authentication session interface GigabitEthernet 1/0/1`	Verifies the authorized session security status.
Step 19	**show mka sessions** **Example:** `Device# show mka sessions`	Verifies the established MKA sessions.

Configure MACsec MKA using PSK

Beginning in privileged EXEC mode, follow these steps to configure MACsec MKA policies using a Pre Shared Key (PSK).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **key chain** *key-chain-name* **macsec**
4. **key** *hex-string*
5. **cryptographic-algorithm** *{aes-128-cmac | aes-256-cmac}*
6. **key-string** *{ [0|6|7] pwd-string | pwd-string}*
7. **lifetime local** [*start timestamp {hh::mm::ss | day | month | year}*] [**duration** *seconds* | *end timestamp {hh::mm::ss | day | month | year}*]
8. **end**

DETAILED STEPS

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	**key chain** *key-chain-name* **macsec** **Example:** `Device(config)# key chain keychain1 macsec`	Configures a key chain and enters the key chain configuration mode.
Step 4	**key** *hex-string* **Example:** `Device(config-key-chain)# key 1000`	Configures a unique identifier for each key in the keychain and enters the keychain's key configuration mode. **Note** For 128-bit encryption, use any value between 1 and 32 hex digit key-string. For 256-bit encryption, use 64 hex digit key-string.
Step 5	**cryptographic-algorithm** *{aes-128-cmac | aes-256-cmac}* **Example:** `Device(config-key-chain)# cryptographic-algorithm aes-128-cmac`	Set cryptographic authentication algorithm with 128-bit or 256-bit encryption.
Step 6	**key-string** *{ [0|6|7] pwd-string | pwd-string}* **Example:** `Device(config-key-chain)# key-string 12345678901234567890123456789012`	Sets the password for a key string. Only hex characters must be entered.
Step 7	**lifetime local** [*start timestamp {hh::mm::ss | day | month | year}*] [**duration** *seconds* | *end timestamp {hh::mm::ss | day | month | year}*] **Example:** `Device(config-key-chain)# lifetime local 12:12:00 July 28 2016 12:19:00 July 28 2016`	Sets the lifetime of the pre shared key.
Step 8	**end** **Example:** `Device(config-key-chain)# end`	Exits key chain configuration mode and returns to privileged EXEC mode.

Configure MACsec MKA on an Interface using PSK

Beginning in privileged EXEC mode, follow these steps to configure MACsec MKA policies on an interface using a Pre Shared Key (PSK).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *interface-id*

4. **macsec network-link**
5. **mka policy** *policy-name*
6. **mka pre-shared-key key-chain** *key-chain name* [**fallback key-chain** *key-chain name*]
7. **macsec replay-protection window-size** *frame number*
8. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config-if)# interface GigabitEthernet 0/0/0`	Enters interface configuration mode.
Step 4	**macsec network-link** **Example:** `Device(config-if)# macsec network-link`	Enables MACsec on the interface.
Step 5	**mka policy** *policy-name* **Example:** `Device(config-if)# mka policy mka_policy`	Configures an MKA policy.
Step 6	**mka pre-shared-key key-chain** *key-chain name* [**fallback key-chain** *key-chain name*] **Example:** `Device(config-if)# mka pre-shared-key key-chain key-chain-name`	Configures an MKA pre-shared-key key-chain name.
Step 7	**macsec replay-protection window-size** *frame number* **Example:** `Device(config-if)# macsec replay-protection window-size 10`	Sets the MACsec window size for replay protection.
Step 8	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

What to do next

It is not recommended to change the MKA policy on an interface with MKA PSK configured when the session is running. However, if a change is required, you must reconfigure the policy as follows:

1. Disable the existing session by removing macsec network-link configuration on each of the participating node using the **no macsec network-link** command
2. Configure the MKA policy on the interface on each of the participating node using the **mka policy policy-name** command.
3. Enable the new session on each of the participating node by using the **macsec network-link** command.

Configuring MACsec MKA Using Certificate-based MACsec

To configure MACsec with MKA on point-to-point links, perform these tasks:

- Configure Certificate Enrollment
  - Generate Key Pairs
  - Configure SCEP Enrollment
  - Configure Certificates Manually
- Configure an Authentication Policy
- Configure certificate-based MACsec Profiles and IEEE 802.1x Credentials
- Configure MKA MACsec using certificate-based MACsec on Interfaces

Generate Key Pairs

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key generate rsa label** *label-name* **general-keys modulus** *size*
4. **end**
5. **show authentication session interface** *interface-id*

DETAILED STEPS

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	**crypto key generate rsa label** *label-name* **general-keys modulus** *size* **Example:** `Device(config)# crypto key generate rsa label general-keys modulus 2048`	Generates a RSA key pair for signing and encryption. You can also assign a label to each key pair using the label keyword. The label is referenced by the trustpoint that uses the key pair. If you do not assign a label, the key pair is automatically labeled <Default-RSA-Key>. If you do not use additional keywords this command generates one general purpose RSA key pair. If the modulus is not specified, the default key modulus of 1024 is used. You can specify other modulus sizes with the modulus keyword.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.
Step 5	**show authentication session interface** *interface-id* **Example:** `Device# show authentication session interface gigabitethernet 0/1/1`	Verifies the authorized session security status.

Configure Enrollment using SCEP

Simple Certificate Enrollment Protocol (SCEP) is a Cisco-developed enrollment protocol that uses HTTP to communicate with the certificate authority (CA) or registration authority (RA). SCEP is the most commonly used method for sending and receiving requests and certificates.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki trustpoint** *server name*
4. **enrollment url** *url name pem*
5. **rsakeypair** *label*
6. **serial-number none**
7. **ip-address none**
8. **revocation-check crl**
9. **auto-enroll** *percent* **regenerate**
10. **exit**
11. **crypto pki authenticate** *name*
12. **end**
13. **show crypto pki certificate** *trustpoint name*

DETAILED STEPS

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	**crypto pki trustpoint** *server name* **Example:** `Device(config)# crypto pki trustpoint ka`	Declares the trustpoint and a given name and enters ca-trustpoint configuration mode.
Step 4	**enrollment url** *url name pem* **Example:** `Device(ca-trustpoint)# enrollment url http://url:80`	Specifies the URL of the CA on which your device should send certificate requests. An IPv6 address can be added in the URL enclosed in brackets. For example: http:// [2001:DB8:1:1::1]:80. The pem keyword adds privacy-enhanced mail (PEM) boundaries to the certificate request.
Step 5	**rsakeypair** *label* **Example:** `Device(ca-trustpoint)# rsakeypair exampleCAkeys`	Specifies which key pair to associate with the certificate. **Note** The **rsakeypair** name must match the trust-point name.
Step 6	**serial-number none** **Example:** `Device(ca-trustpoint)# serial-number none`	The **none** keyword specifies that a serial number will not be included in the certificate request.
Step 7	**ip-address none** **Example:** `Device(ca-trustpoint)# ip-address none`	The **none** keyword specifies that no IP address should be included in the certificate request.
Step 8	**revocation-check crl** **Example:** `Device(ca-trustpoint)# revocation-check crl`	Specifies CRL as the method to ensure that the certificate of a peer has not been revoked.
Step 9	**auto-enroll** *percent* **regenerate** **Example:** `Device(ca-trustpoint)# auto-enroll 90 regenerate`	Enables auto-enrollment, allowing the client to automatically request a rollover certificate from the CA. If auto-enrollment is not enabled, the client must be manually re-enrolled in your PKI upon certificate expiration.

	Command or Action	Purpose
		By default, only the Domain Name System (DNS) name of the device is included in the certificate. Use the percent argument to specify that a new certificate will be requested after the percentage of the lifetime of the current certificate is reached. Use the regenerate keyword to generate a new key for the certificate even if a named key already exists. If the key pair being rolled over is exportable, the new key pair will also be exportable. The following comment will appear in the trustpoint configuration to indicate whether the key pair is exportable: "! RSA key pair associated with trustpoint is exportable." It is recommended that a new key pair be generated for security reasons.
Step 10	**exit** **Example:** `Device(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode and returns to global configuration mode.
Step 11	**crypto pki authenticate** *name* **Example:** `Device(config)# crypto pki authenticate myca`	Retrieves the CA certificate and authenticates it.
Step 12	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.
Step 13	**show crypto pki certificate** *trustpoint name* **Example:** `Device# show crypto pki certificate ka`	Displays information about the certificate for the trust point.

Configure Enrollment Manually

If your CA does not support SCEP or if a network connection between the router and CA is not possible. Perform the following task to set up manual certificate enrollment:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki trustpoint** *server name*
4. **enrollment url** *url name pem*
5. **rsakeypair** *label*
6. **serial-number none**
7. **ip-address none**

8. **revocation-check crl**
9. **exit**
10. **crypto pki authenticate** *name*
11. **crypto pki enroll** *name*
12. **crypto pki import** *name* **certificate**
13. **end**
14. **show crypto pki certificate** *trustpoint name*

DETAILED STEPS

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	**crypto pki trustpoint** *server name* **Example:** `Device# crypto pki trustpoint ka`	Declares the trustpoint and a given name and enters ca-trustpoint configuration mode.
Step 4	**enrollment url** *url name pem* **Example:** `Device(ca-trustpoint)# enrollment url http://url:80`	Specifies the URL of the CA on which your device should send certificate requests. An IPv6 address can be added in the URL enclosed in brackets. For example: http:// [2001:DB8:1:1::1]:80. The pem keyword adds privacy-enhanced mail (PEM) boundaries to the certificate request.
Step 5	**rsakeypair** *label* **Example:** `Device(ca-trustpoint)# rsakeypair exampleCAkeys`	Specifies which key pair to associate with the certificate.
Step 6	**serial-number none** **Example:** `Device(ca-trustpoint)# serial-number none`	The **none** keyword specifies that a serial number will not be included in the certificate request.
Step 7	**ip-address none** **Example:** `Device(ca-trustpoint)# ip-address none`	The **none** keyword specifies that no IP address should be included in the certificate request.

	Command or Action	Purpose
Step 8	**revocation-check crl** **Example:** `Device(ca-trustpoint)# revocation-check crl`	Specifies CRL as the method to ensure that the certificate of a peer has not been revoked.
Step 9	**exit** **Example:** `Device(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode and returns to global configuration mode.
Step 10	**crypto pki authenticate** *name* **Example:** `Device(config)# crypto pki authenticate myca`	Retrieves the CA certificate and authenticates it.
Step 11	**crypto pki enroll** *name* **Example:** `Device(config)# crypto pki enroll myca`	Generates certificate request and displays the request for copying and pasting into the certificate server. Enter enrollment information when you are prompted. For example, specify whether to include the device FQDN and IP address in the certificate request. You are also given the choice about displaying the certificate request to the console terminal. The base-64 encoded certificate with or without PEM headers as requested is displayed.
Step 12	**crypto pki import** *name* **certificate** **Example:** `Device(config)# crypto pki import myca certificate`	Imports a certificate via TFTP at the console terminal, which retrieves the granted certificate. The device attempts to retrieve the granted certificate via TFTP using the same filename used to send the request, except the extension is changed from ".req" to ".crt". For usage key certificates, the extensions "-sign.crt" and "-encr.crt" are used. The device parses the received files, verifies the certificates, and inserts the certificates into the internal certificate database on the switch. **Note** Some CAs ignore the usage key information in the certificate request and issue general purpose usage certificates. If your CA ignores the usage key information in the certificate request, only import the general purpose certificate. The router will not use one of the two key pairs generated.
Step 13	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 14	**show crypto pki certificate** *trustpoint name* **Example:** `Device# show crypto pki certificate  ka`	Displays information about the certificate for the trust point.

Enable 802.1x Authentication and Configure AAA

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **aaa new-model**
4. **dot1x system-auth-control**
5. **radius server** *name*
6. **address** *ip_address* **auth-port** *port_number* **acct-port** *port_number*
7. **automate-tester username** *username*
8. **key** *string*
9. **radius-server deadtime** *minutes*
10. **exit**
11. **aaa group server radius** *group_name*
12. **server** *name*
13. **exit**
14. **aaa authentication dot1x default group** *group_name*
15. **aaa authorization network default group** *group_name*

DETAILED STEPS

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	**aaa new-model** **Example:** `device(config)# aaa new-model`	Enables AAA.
Step 4	**dot1x system-auth-control** **Example:**	Enables 802.1X on your device.

	Command or Action	Purpose
	`device(config)# dot1x system-auth-control`	
Step 5	**radius server** *name* **Example:** `device(config)# radius server ISE`	Specifies the name of the RADIUS server configuration for Protected Access Credential (PAC) provisioning and enters RADIUS server configuration mode.
Step 6	**address** *ip_address* **auth-port** *port_number* **acct-port** *port_number* **Example:** `device(config-radius-server)# address ipv4 10.64.72.90 auth-port 1645 acct-port 1646`	Configures the IPv4 address for the RADIUS server accounting and authentication parameters.
Step 7	**automate-tester username** *username* **Example:** `device(config-radius-server)# automate-tester username dummy`	Enables the automated testing feature for the RADIUS server. With this practice, the device sends periodic test authentication messages to the RADIUS server. It looks for a Radius response from the server. A success message is not necessary; a failed authentication suffices, because it shows that the server is alive.
Step 8	**key** *string* **Example:** `device(config-radius-server)# key dummy123`	Configures the authentication and encryption key for all RADIUS communications between the device and the RADIUS server.
Step 9	**radius-server deadtime** *minutes* **Example:** `device(config-radius-server)#radius-server deadtime 2`	Improves RADIUS response time when some servers might be unavailable and skips unavailable servers immediately.
Step 10	**exit**	Returns to global configuration mode.
Step 11	**aaa group server radius** *group_name* **Example:** `device(config)# aaa group server radius ISEGRP`	Groups different RADIUS server hosts into distinct lists and distinct methods, and enters server group configuration mode.
Step 12	**server** *name* **Example:** `device(config)#server ise`	
Step 13	**exit** **Example:** `device(config-radius-server)# exit`	Returns to global configuration mode.
Step 14	**aaa authentication dot1x default group** *group_name* **Example:** `device(config)# aaa authentication dot1x default group ISEGRP`	Sets the default authentication server group for IEEE 802.1x.

	Command or Action	Purpose
Step 15	**aaa authorization network default group** *group_name* **Example:** `device(config)# aaa authorization network default group ISEGRP`	Sets the network authorization default group.

Apply the 802.1x MKA MACsec Configuration on the Interfaces

To apply MKA MACsec using EAP-TLS to interfaces, complete the following steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *interface_id*
4. **macsec network-link**
5. **authentication periodic**
6. **authentication timer reauthenticate interval**
7. **access-session host-mode multi-domain**
8. **access-session closed**
9. **access-session port-control auto**
10. **dot1x pae both**
11. **dot1x credentials profile**
12. **dot1x supplicant eap profile** *profile_name*
13. **dot1x authenticator eap profile** *profile_name*
14. **service-policy type control subscriber** *control_policy_name*
15. **exit**
16. **show macsec interface**
17. **copy running-config startup-config**

DETAILED STEPS

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 te0/1/2`	Identifies the MACsec interface, and enter interface configuration mode. The interface must be a physical interface.
Step 4	**macsec network-link** **Example:** `device(config)# macsec network-link`	Enables MACsec on the interface.
Step 5	**authentication periodic** **Example:** `device(config)# authentication periodic`	Enables reauthentication for this port.
Step 6	**authentication timer reauthenticate interval** **Example:** `device(config)# authentication timer reauthenticate interval`	Sets the reauthentication interval.
Step 7	**access-session host-mode multi-domain** **Example:** `device(config)# access-session host-mode multi-domain`	Allows hosts to gain access to the interface.
Step 8	**access-session closed** **Example:** `device(config)# access-session closed`	Prevents preauthentication access on the interface.
Step 9	**access-session port-control auto** **Example:** `device(config)# access-session port-control auto`	Sets the authorization state of a port.
Step 10	**dot1x pae both** **Example:** `device(config)# dot1x pae both`	Configures the port as an 802.1X port access entity (PAE) supplicant and authenticator.
Step 11	**dot1x credentials profile** **Example:** `device(config)# dot1x credentials profile`	Assigns a 802.1x credentials profile to the interface.
Step 12	**dot1x supplicant eap profile** *profile_name* **Example:** `device(config)# dot1x supplicant eap profile eap1`	Assigns the EAP-TLS profile to the interface.
Step 13	**dot1x authenticator eap profile** *profile_name* **Example:**	Assigns the EAP-TLS profile to use during 802.1x authentication.

	Command or Action	Purpose
	`device(config)# dot1x authenticator eap profile eap1`	
Step 14	**service-policy type control subscriber** *control_policy_name* **Example:** `device(config)# service-policy type control subscriber controlPolicy2`	Applies a subscriber control policy to the interface.
Step 15	**exit** **Example:** `device(config)# exit`	Returns to privileged EXEC mode.
Step 16	**show macsec interface** **Example:** `device# show macsec interface`	Displays MACsec details for the interface.
Step 17	**copy running-config startup-config** **Example:** `device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configure MKA/MACsec for Port Channel using PSK

Beginning in privileged EXEC mode, complete the following steps to configure MKA policies on an interface using a pre-shared key (PSK):

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config-if)# interface gigabitethernet 1/0/3`	Enters interface configuration mode.
Step 4	**macsec network-link** **Example:** `Device(config-if)# macsec network-link`	Enables MACsec on the interface. Supports layer 2 and layer 3 port channels.

	Command or Action	Purpose
Step 5	**mka policy** *policy-name* **Example:** `Device(config-if)# mka policy mka_policy`	Configures an MKA policy.
Step 6	**mka pre-shared-key key-chain** *key-chain name* [**fallback key-chain** *key-chain name*] **Example:** `Device(config-if)# mka pre-shared-key key-chain key-chain-name`	Configures an MKA pre-shared-key key-chain name. **Note** The MKA pre-shared key can be configured on either physical interface or subinterfaces and not on both.
Step 7	**macsec replay-protection window-size** *frame number* **Example:** `Device(config-if)# macsec replay-protection window-size 0`	Sets the MACsec window size for replay protection.
Step 8	**channel-group** *channel-group-number* **mode** {**auto** | **desirable**} | {**active** | **passive**} | {**on**} **Example:** `Device(config-if)# channel-group 3 mode auto active on`	Configures the port in a channel group and sets the mode. **Note** You cannot configure ports in a channel group without configuring MACsec on the interface. You must configure the commands in Step 3, 4, 5 and 6 before this step. The channel-number range is from 1 to 4096. The port channel that is associated with this channel group is automatically created if the port channel does not already exist. For mode, select one of the following keywords: • **auto**: Enables PAgP only if a PAgP device is detected. This places the port into a passive negotiating state, in which the port responds to PAgP packets it receives but does not start PAgP packet negotiation. **Note** The **auto** keyword is not supported when EtherChannel members are from different switches in the switch stack. • **desirable**: Unconditionally enables PAgP. This places the port into an active negotiating state, in which the port starts negotiations with other ports by sending PAgP packets. **Note** The **desirable** keyword is not supported when EtherChannel members are from different switches in the switch stack. • **on**: Forces the port to channel without PAgP or LACP. In the on mode, an EtherChannel exists only when a

	Command or Action	Purpose
		port group in the **on** mode is connected to another port group in the **on** mode. • **active**: Enables LACP only if a LACP device is detected. It places the port into an active negotiating state in which the port starts negotiations with other ports by sending LACP packets. • **passive**: Enables LACP on the port and places it into a passive negotiating state in which the port responds to LACP packets that it receives, but does not start LACP packet negotiation.
Step 9	**end** **Example:** `Device(config-if)# cend`	Exits interface configuration mode and returns to privileged EXEC mode.

Configure Port Channel Logical Interfaces for Layer 2 EtherChannels

To create a port channel interface for a Layer 2 EtherChannel, perform this task:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface port-channel** *channel-group-number* **Example:** `Device(config)# interface port-channel 1`	Creates the port channel interface. **Note** Use the **no** form of this command to delete the port channel interface.
Step 4	**switchport** **Example:** `Device(config-if)# switchport`	Switches an interface that is in Layer 3 mode into Layer 2 mode for Layer 2 configuration.
Step 5	**switchport mode** {**access** | **trunk**} **Example:** `Device(config-if)# switchport mode access`	Assigns all ports as static-access ports in the same VLAN, or configure them as trunks.

	Command or Action	Purpose
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configure Port Channel Logical Interfaces for Layer 3 EtherChannels

To create a port channel interface for a Layer 3 EtherChannel, perform this task:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/2`	Enters interface configuration mode.
Step 4	**no switchport** **Example:** `Device(config-if)# no switchport`	Switches an interface that is in Layer 2 mode into Layer 3 mode for Layer 3 configuration.
Step 5	**ip address** *ip-address subnet_mask* **Example:** `Device(config-if)# ip address 10.2.2.3 255.255.255.254`	Assigns an IP address and subnet mask to the EtherChannel.
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring MACsec Cipher Announcement

Configure an MKA Policy for Secure Announcement

Beginning in privileged EXEC mode, follow these steps to create an MKA Protocol policy to enable secure announcement in MKPDUs. By default, secure announcements are disabled.

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	**mka policy** *policy-name* **Example:** `Device(config)# mka policy mka_policy`	Identifies an MKA policy and enters MKA policy configuration mode. The maximum policy name length is 16 characters. **Note** The default MACsec cipher suite in the MKA policy is GCM-AES-128. If the device supports both "GCM-AES-128" and "GCM-AES-256" ciphers, it is highly recommended to define and use a user defined MKA policy to include both 128 and 256 bits ciphers or only 256 bits cipher, as may be required.
Step 4	**key-server** *priority* **Example:** `Device(config-mka-policy)# key-server priority 200`	Configures MKA key server options and sets priority between 0-255. **Note** When value of key server priority is set to 255, the peer cannot become the key server. The key server priority value is valid only for MKA PSK. This does not apply to MKA EAP-TLS.
Step 5	**send-secure-announcements** **Example:** `Device(config-mka-policy)# send-secure-announcements`	Enables sending of secure announcements. Use the **no** form of the command to disable sending of secure announcements. By default, secure announcements are disabled.
Step 6	**macsec-cipher-suite** {*gcm-aes-128* | *gcm-aes-256*} **Example:** `Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128`	Configures cipher suite for deriving SAK with 128-bit or 256-bit encryption.
Step 7	**end** **Example:** `Device(config-mka-policy)# end`	Exits MKA policy configuration mode and returns to privileged EXEC mode.
Step 8	**show mka policy** **Example:**	Displays MKA policies.

	Command or Action	Purpose
	`Device# show mka policy`	

Configure Secure Announcement Globally

Beginning in privileged EXEC mode, follow these steps to enable secure announcement globally across all the MKA Policies.

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	**mka defaults policy send-secure-announcements** **Example:** `Device(config)# mka defaults policy send-secure-announcements`	Enables sending of secure announcements in MKPDUs across MKA policies. By default, secure announcements are disabled.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configure EAPoL Announcements on an Interface

Beginning in privileged EXEC mode, follow these steps to configure EAPoL Announcement on 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.

	Command or Action	Purpose
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Identifies the MACsec interface, and enters interface configuration mode. The interface must be a physical interface.
Step 4	**eapol annoucement** **Example:** `Device(config-if)# eapol announcement`	Enables EAPoL announcements. Use the **no** form of the command to disable EAPoL announcements. By default, EAPoL announcements are disabled.
Step 5	**end** **Example:** `Device(config-if)# configure terminal`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for MACsec Encryption

Example: Configuring MKA and MACsec

This example shows how to create an MKA policy:

```
Device> enable
Device# configure terminal
Device(config)# mka policy mka_policy
Device(config-mka-policy)# key-server priority 200
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128
Device(config-mka-policy)# confidentiality-offset 30
Device(config-mka-policy)# ssci-based-on-sci
Device(config-mka-policy)#end
```

This example shows how to configure MACsec on an interface:

```
Device> enable
Device# configure terminal
Device(config)# interface GigabitEthernet 1/0/1
Device(config-if)# switchport access vlan 1
Device(config-if)# switchport mode access
Device(config-if)# macsec
Device(config-if)#access-session event linksec fail action authorize vlan 1
Device(config-if)# access-session host-mode multi-domain
Device(config-if)# access-session linksec policy must-secure
Device(config-if)# access-session port-control auto
Device(config-if)#authentication periodic
Device(config-if)# authentication timer reauthenticate
Device(config-if)# authentication violation protect
Device(config-if)#mka policy mka_policy
Device(config-if)# dot1x pae authenticator
Device(config-if)# spanning-tree portfast
Device(config-if)#end
```

Examples: Configuring MACsec MKA Using PSK

This example shows how to configure MACsec MKA using PSK.

```
Device> enable
Device# configure terminal
Device(config)# Key chain keychain1 macsec
Device(config-keychain)# key 1000
Device(config-keychain-key)# cryptographic-algorithm aes-128-cmac
Device(config-keychain-key)# key-string 12345678901234567890123456789012
Device(config-keychain-key)# lifetime local 12:12:00 July 28 2016 12:19:00 July 28 2016
Device(config-keychain-key)# end
```

This example shows how to configure MACsec MKA on an interface using PSK.

```
Device> enable
Device# configure terminal
Device(config)# interface GigabitEthernet 0/0/0
Device(config-if)# mka policy mka_policy
Device(config-if)# mka pre-shared-key key-chain key-chain-name
Device(config-if)# macsec replay-protection window-size 10
Device(config-if)# end
```

MKA-PSK: CKN Behavior Change

Starting Cisco IOS XE Fuji 16.8.1 release, for MKA PSK sessions, the CKN uses exactly the same string as the CKN which is configured as the hex-string for the key, instead of the fixed 32 bytes.

```
Device> enable
Device# configure terminal
Device(config)# key chain abc macsec
Device(config-keychain)# key 11
Device(config-keychain-key)# cryptographic-algorithm aes-128-cmac
Device(config-keychain-key)# key-string 12345678901234567890123456789013
Device(config-keychain-key)# lifetime local 12:21:00 Sep 9 2015 infinite
Device(config-keychain-key)# end
```

The following is sample output of the **show mka session** command for the above configuration:

```
Device# show mka session

Total MKA Sessions....... 1
Secured Sessions... 1
Pending Sessions... 0
=============================================================================================
Interface      Local-TxSCI         Policy-Name      Inherited         Key-Server

Port-ID        Peer-RxSCI          MACsec-Peers     Status            CKN

=============================================================================================
Et0/0          aabb.cc00.6600/0002     icv            NO                NO

2              aabb.cc00.6500/0002 1               Secured           11   *Note that the
 CKN key-string is exactly the same that has been configured for the key as hex-string.*
```

In case of interoperability between two images, where one having the CKN behavior change, and one without the CKN behavior change, the hex-string for the key must be a 64-character hex-string with zero padded for it to work on a device that has an image with the CKN behavior change. See the examples below:

Configuration without CKN key-string behavior change:

```
Device# configure terminal
Device(config)# key chain abc macsec
Device(config-keychain)# key 11
Device(config-keychain-key)# cryptographic-algorithm aes-128-cmac
Device(config-keychain-key)# key-string 12345678901234567890123456789013
Device(config-keychain-key)# lifetime local 12:21:00 Sep 9 2015 infinite
Device(config-keychain-key)# end
```

Configuration with CKN key-string behavior change:

```
Device# configure terminal
Device(config)# key chain abc macsec
Device(config-keychain)# key 1100000000000000000000000000000000000000000000000000000000000000
Device(config-keychain-key)# cryptographic-algorithm aes-128-cmac
Device(config-keychain-key)# key-string 12345678901234567890123456789013
Device(config-keychain-key)# lifetime local 12:21:00 Sep 9 2015 infinite
Device(config-keychain-key)# end
```

Examples: Configuring MACsec MKA using Certificate-based MACsec

This example shows how to configure MACsec MKA using certificate-based MACsec:

```
Device> enable
Device# configure terminal
Device(config)# interface Gigabitethernet 1/0/1
Device(config-if)# macsec network-link
Device(config-if)# authentication periodic
Device(config-if)# authentication timer reauthenticate interval
Device(config-if)#access-session host-mode multi-domain
Device(config-if)# access-session closed
Device(config-if)# access-session port-control auto
Device(config-if)# dot1x pae both
Device(config-if)#dot1x credentials profile
Device(config-if)# dot1x supplicant eap profile profile_eap_tls
Device(config-if)#service-policy type control subscriber sub1
Device(config-if)# end
```

Examples: Configuring MACsec MKA for Port Channel using PSK

Etherchannel Mode — Static/On

The following is sample configuration on Device 1 and Device 2 with EtherChannel Mode on:

```
Device> enable
Device# configure terminal
Device(config)# key chain KC macsec
Device(config-key-chain)# key 1000
Device(config-key-chain)# cryptographic-algorithm aes-128-cmac
Device(config-key-chain)# key-string FC8F5B10557C192F03F60198413D7D45
Device(config-key-chain)# exit
Device(config)# mka policy POLICY
Device(config-mka-policy)# key-server priority 0
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128
Device(config-mka-policy)# confidentiality-offset 0
Device(config-mka-policy)# exit
Device(config)# interface gigabitethernet 1/0/1
Device(config-if)# channel-group 2 mode on
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# channel-group 2 mode on
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# end
```

Layer 2 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

The following is sample output from the**show etherchannel summary** command:

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports

------+-------------+-----------+-----------------------------------------------

2      Po2(RU)           -        Te1/0/1(P)  Te1/0/2(P)
```

Layer 3 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.3 255.255.255.0
```

```
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.4 255.255.255.0
Device(config-if)# no shutdown
Device(config-if)# end
```

The following is sample output from the **show etherchannel summary** command:

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports

------+-------------+-----------+-----------------------------------------------

2      Po2(RU)           -        Te1/0/1(P)   Te1/0/2(P)
```

Etherchannel Mode — LACP

The following is sample configuration on Device 1 and Device 2 with EtherChannel Mode as LACP.

```
Device> enable
Device# configure terminal
Device(config)# key chain KC macsec
Device(config-key-chain)# key 1000
Device(config-key-chain)# cryptographic-algorithm aes-128-cmac
Device(config-key-chain)# key-string FC8F5B10557C192F03F60198413D7D45
Device(config-key-chain)# exit
Device(config)# mka policy POLICY
Device(config-mka-policy)# key-server priority 0
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128
Device(config-mka-policy)# confidentiality-offset 0
Device(config-mka-policy)# exit
Device(config)# interface gigabitethernet 1/0/1
Device(config-if)# channel-group 2 mode active
```

```
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# channel-group 2 mode active
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# end
```

Layer 2 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

The following is sample output from the **show etherchannel summary** command:

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1


------+-------------+-----------+-----------------------------------------------

2      Po2(SU)         LACP      Te1/1/1(P)  Te1/1/2(P)
```

Layer 3 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.3 255.255.255.0
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.4 255.255.255.0
Device(config-if)# no shutdown
Device(config-if)# end
```

The following is sample output from the **show etherchannel summary** command:

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports

------+-------------+-----------+-----------------------------------------------

2      Po2(RU)          LACP      Te1/1/1(P)  Te1/1/2(P)
```

Etherchannel Mode — PAgP

The following is sample configuration on Device 1 and Device 2 with EtherChannel Mode as PAgP:

```
Device> enable
Device# configure terminal
Device(config)# key chain KC macsec
Device(config-key-chain)# key 1000
Device(config-key-chain)# cryptographic-algorithm aes-128-cmac
Device(config-key-chain)# key-string FC8F5B10557C192F03F60198413D7D45
Device(config-key-chain)# exit
Device(config)# mka policy POLICY
Device(config-mka-policy)# key-server priority 0
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128
Device(config-mka-policy)# confidentiality-offset 0
```

```
Device(config-mka-policy)# exit
Device(config)# interface gigabitethernet 1/0/1
Device(config-if)# channel-group 2 mode desirable
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# channel-group 2 mode desirable
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# end
```

Layer 2 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

The following shows a sample output from the **show etherchannel summary** command.

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1

------+-------------+-----------+-----------------------------------------------

2      Po2(SU)          PAgP      Te1/1/1(P)  Te1/1/2(P)
```

Layer 3 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.3 255.255.255.0
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.4 255.255.255.0
Device(config-if)# no shutdown
Device(config-if)# end
```

The following is sample output from the **show etherchannel summary** command:

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports

------+-------------+-----------+-----------------------------------------------

2      Po2(RU)         PAgP      Te1/1/1(P)  Te1/1/2(P)
```

Displaying Active MKA Sessions

The following shows all the active MKA sessions.

```
Device# show mka sessions interface Te1/0/1

===============================================================================
Interface      Local-TxSCI         Policy-Name      Inherited
Key-Server
Port-ID        Peer-RxSCI          MACsec-Peers     Status            CKN
```

```
Te1/0/1         00a3.d144.3364/0025 POLICY           NO                  NO

37              701f.539b.b0c6/0032 1                Secured
1000
```

Examples: Configuring MACsec Cipher Announcement

This example shows how to configure MKA policy for Secure Announcement:

```
Device> enable
Device# configure terminal
Device(config)# mka policy mka_policy
Device(config-mka-policy)# key-server 2
Device(config-mka-policy)# send-secure-announcements
Device(config-mka-policy)#macsec-cipher-suite gcm-aes-128confidentiality-offset 0
Device(config-mka-policy)# end
```

This example shows how to configure Secure Announcement globally:

```
Device> enable
Device# configure terminal
Device(config)# mka defaults policy send-secure-announcements
Device(config)# end
```

This example shows how to configure EAPoL Announcements on an interface:

```
Device> enable
Device# configure terminal
Device(config)# interface GigabitEthernet 1/0/1
Device(config-if)# eapol announcement
Device(config-if)# end
```

The following is sample output from the **show running-config interface** *interface-name* command with EAPoL announcement enabled.

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

switchport mode access
 macsec
 access-session host-mode multi-host
 access-session closed
 access-session port-control auto
 dot1x pae authenticator
 dot1x timeout quiet-period 10
 dot1x timeout tx-period 5
 dot1x timeout supp-timeout 10
 dot1x supplicant eap profile peap
 eapol announcement
 spanning-tree portfast
 service-policy type control subscriber Dot1X
```

The following is sample output from the **show mka sessions interface** *interface-name* **detail** command with secure announcement disabled.

```
Device# show mka sessions interface GigabitEthernet 1/0/1 detail
```

```
MKA Detailed Status for MKA Session
===================================
Status: SECURED - Secured MKA Session with MACsec

Local Tx-SCI..............  204c.9e85.ede4/002b
Interface MAC Address....  204c.9e85.ede4
MKA Port Identifier......  43
Interface Name...........  GigabitEthernet1/0/1
Audit Session ID.........
CAK Name (CKN)...........
0100000000000000000000000000000000000000000000000000000000000000
Member Identifier (MI)...  D46CBEC05D5D67594543CEAE
Message Number (MN)......  89567
EAP Role.................  NA
Key Server...............  YES
MKA Cipher Suite.........  AES-128-CMAC

Latest SAK Status........  Rx & Tx
Latest SAK AN............  0
Latest SAK KI (KN).......  D46CBEC05D5D67594543CEAE00000001 (1)
Old SAK Status...........  FIRST-SAK
Old SAK AN...............  0
Old SAK KI (KN)..........  FIRST-SAK (0)

SAK Transmit Wait Time...  0s (Not waiting for any peers to respond)
SAK Retire Time..........  0s (No Old SAK to retire)

MKA Policy Name..........  p2
Key Server Priority......  2
Delay Protection.........  NO
Replay Protection........  YES
Replay Window Size.......  0
Confidentiality Offset...  0
Algorithm Agility........  80C201
Send Secure Announcement..  DISABLED
SAK Cipher Suite.........  0080C20001000001 (GCM-AES-128)
MACsec Capability........  3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired...........  YES

# of MACsec Capable Live Peers............  1
# of MACsec Capable Live Peers Responded..  1

Live Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
  38046BA37D7DA77E06D006A9  89555       c800.8459.e764/002a   10

Potential Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
```

```
Dormant Peers List:
  MI                          MN          Rx-SCI (Peer)        KS Priority
  -----------------------------------------------------------------------
```

The following is sample output from the **show mka sessions details** command with secure announcement disabled.

```
Device# show mka sessions details

MKA Detailed Status for MKA Session
===================================
Status: SECURED - Secured MKA Session with MACsec

Local Tx-SCI.............. 204c.9e85.ede4/002b
Interface MAC Address.... 204c.9e85.ede4
MKA Port Identifier...... 43
Interface Name........... GigabitEthernet1/0/1
Audit Session ID.........
CAK Name (CKN)...........
0100000000000000000000000000000000000000000000000000000000000000
Member Identifier (MI)... D46CBEC05D5D67594543CEAE
Message Number (MN)...... 89572
EAP Role................. NA
Key Server............... YES
MKA Cipher Suite......... AES-128-CMAC

Latest SAK Status........ Rx & Tx
Latest SAK AN............ 0
Latest SAK KI (KN)....... D46CBEC05D5D67594543CEAE00000001 (1)
Old SAK Status........... FIRST-SAK
Old SAK AN............... 0
Old SAK KI (KN).......... FIRST-SAK (0)

SAK Transmit Wait Time... 0s (Not waiting for any peers to respond)
SAK Retire Time.......... 0s (No Old SAK to retire)

MKA Policy Name.......... p2
Key Server Priority...... 2
Delay Protection......... NO
Replay Protection........ YES
Replay Window Size....... 0
Confidentiality Offset... 0
Algorithm Agility........ 80C201
Send Secure Announcement.. DISABLED
SAK Cipher Suite......... 0080C20001000001 (GCM-AES-128)
MACsec Capability........ 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired........... YES

# of MACsec Capable Live Peers............ 1
# of MACsec Capable Live Peers Responded.. 1

Live Peers List:
  MI                          MN          Rx-SCI (Peer)        KS Priority
```

```
  ----------------------------------------------------------------------
  38046BA37D7DA77E06D006A9  89560       c800.8459.e764/002a   10

Potential Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ----------------------------------------------------------------------

Dormant Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ----------------------------------------------------------------------
```

The following is sample output from the **show mka policy** *policy-name* **detail** command with secure announcement disabled.

```
Device# show mka policy p2 detail

MKA Policy Configuration ("p2")
=======================
MKA Policy Name........ p2
Key Server Priority.... 2
Confidentiality Offset. 0
Send Secure Announcement..DISABLED
Cipher Suite(s)........ GCM-AES-128

Applied Interfaces...
  GigabitEthernet1/0/1
```

Examples: Displaying MKA Information

The following is sample output from the **show mka sessions** command.

```
Device# show mka sessions


Total MKA Sessions....... 1
      Secured Sessions... 1
      Pending Sessions... 0

=============================================================================
Interface      Local-TxSCI         Policy-Name      Inherited
Key-Server
Port-ID        Peer-RxSCI          MACsec-Peers     Status            CKN
=============================================================================
Gi1/0/1        204c.9e85.ede4/002b p2               NO                YES
43             c800.8459.e764/002a 1                Secured
0100000000000000000000000000000000000000000000000000000000000000
```

The following is sample output from the **show mka sessions interface** *interface-name* command.

```
Device# show mka sessions interface GigabitEthernet 1/0/1

Summary of All Currently Active MKA Sessions on Interface
GigabitEthernet1/0/1...

=============================================================================
```

```
Interface      Local-TxSCI         Policy-Name      Inherited
Key-Server
Port-ID        Peer-RxSCI          MACsec-Peers     Status             CKN
=============================================================================
Gi1/0/1        204c.9e85.ede4/002b p2               NO                 YES
43             c800.8459.e764/002a 1                 Secured
0100000000000000000000000000000000000000000000000000000000000000
```

The following is sample output from the **show mka sessions interface** *interface-name* **detail**command.

```
Device# show mka sessions interface GigabitEthernet 1/0/1 detail


MKA Detailed Status for MKA Session
===================================
Status: SECURED - Secured MKA Session with MACsec

Local Tx-SCI.............. 204c.9e85.ede4/002b
Interface MAC Address.... 204c.9e85.ede4
MKA Port Identifier...... 43
Interface Name........... GigabitEthernet1/0/1
Audit Session ID.........
CAK Name (CKN)...........
0100000000000000000000000000000000000000000000000000000000000000
Member Identifier (MI)... D46CBEC05D5D67594543CEAE
Message Number (MN)...... 89567
EAP Role................. NA
Key Server............... YES
MKA Cipher Suite......... AES-128-CMAC

Latest SAK Status........ Rx & Tx
Latest SAK AN............ 0
Latest SAK KI (KN)....... D46CBEC05D5D67594543CEAE00000001 (1)
Old SAK Status........... FIRST-SAK
Old SAK AN............... 0
Old SAK KI (KN).......... FIRST-SAK (0)

SAK Transmit Wait Time... 0s (Not waiting for any peers to respond)
SAK Retire Time.......... 0s (No Old SAK to retire)

MKA Policy Name.......... p2
Key Server Priority...... 2
Delay Protection......... NO
Replay Protection........ YES
Replay Window Size....... 0
Confidentiality Offset... 0
Algorithm Agility........ 80C201
Send Secure Announcement.. DISABLED
SAK Cipher Suite......... 0080C20001000001 (GCM-AES-128)
MACsec Capability........ 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired........... YES

# of MACsec Capable Live Peers............ 1
```

```
# of MACsec Capable Live Peers Responded.. 1

Live Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
  38046BA37D7DA77E06D006A9  89555       c800.8459.e764/002a   10

Potential Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------

Dormant Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
```

The following is sample output from the **show mka sessions details** command:

```
Device# show mka sessions details

MKA Detailed Status for MKA Session
===================================
Status: SECURED - Secured MKA Session with MACsec

Local Tx-SCI............. 204c.9e85.ede4/002b
Interface MAC Address.... 204c.9e85.ede4
MKA Port Identifier...... 43
Interface Name........... GigabitEthernet1/0/1
Audit Session ID.........
CAK Name (CKN)...........
0100000000000000000000000000000000000000000000000000000000000000
Member Identifier (MI)... D46CBEC05D5D67594543CEAE
Message Number (MN)...... 89572
EAP Role................. NA
Key Server............... YES
MKA Cipher Suite......... AES-128-CMAC

Latest SAK Status........ Rx & Tx
Latest SAK AN............ 0
Latest SAK KI (KN)....... D46CBEC05D5D67594543CEAE00000001 (1)
Old SAK Status........... FIRST-SAK
Old SAK AN............... 0
Old SAK KI (KN).......... FIRST-SAK (0)

SAK Transmit Wait Time... 0s (Not waiting for any peers to respond)
SAK Retire Time.......... 0s (No Old SAK to retire)

MKA Policy Name.......... p2
Key Server Priority...... 2
Delay Protection......... NO
Replay Protection........ YES
Replay Window Size....... 0
Confidentiality Offset... 0
Algorithm Agility........ 80C201
```

```
Send Secure Announcement.. DISABLED
SAK Cipher Suite..... 0080C20001000001 (GCM-AES-128)
MACsec Capability..... 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired..... YES

# of MACsec Capable Live Peers..... 1
# of MACsec Capable Live Peers Responded.. 1

Live Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
  38046BA37D7DA77E06D006A9  89560       c800.8459.e764/002a   10

Potential Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------

Dormant Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
```

The following is sample output from the **show mka policy** command:

```
Device# show mka policy


MKA Policy Summary...

Policy            KS       Delay   Replay  Window Conf   Cipher
 Interfaces
Name              Priority Protect Protect Size   Offset Suite(s)
 Applied
================================================================================
*DEFAULT POLICY*  0        FALSE   TRUE    0      0      GCM-AES-128

p1                1        FALSE   TRUE    0      0      GCM-AES-128

p2                2        FALSE   TRUE    0      0      GCM-AES-128
 Gi1/0/1
```

The following is sample output from the **show mka policy** *policy-name* command:

```
Device# show mka policy p2


MKA Policy Summary...

Policy            KS       Delay   Replay  Window Conf   Cipher
 Interfaces
Name              Priority Protect Protect Size   Offset Suite(s)
 Applied
================================================================================
p2                2        FALSE   TRUE    0      0      GCM-AES-128
 Gi1/0/1
```

The following is sample output from the **show mka policy** *policy-name* **detail** command:

```
Device# show mka policy p2 detail

MKA Policy Configuration ("p2")
==========================
MKA Policy Name........ p2
Key Server Priority.... 2
Confidentiality Offset. 0
Send Secure Announcement..DISABLED
Cipher Suite(s)........ GCM-AES-128

Applied Interfaces...
  GigabitEthernet1/0/1
```

The following is sample output from the **show mka statistics interface** *interface-name* command:

```
Device# show mka statistics interface GigabitEthernet 1/0/1


MKA Statistics for Session
==========================
Reauthentication Attempts.. 0

CA Statistics
   Pairwise CAKs Derived... 0
   Pairwise CAK Rekeys..... 0
   Group CAKs Generated.... 0
   Group CAKs Received..... 0

SA Statistics
   SAKs Generated.......... 1
   SAKs Rekeyed............ 0
   SAKs Received........... 0
   SAK Responses Received.. 1

MKPDU Statistics
   MKPDUs Validated & Rx... 89585
      "Distributed SAK".. 0
      "Distributed CAK".. 0
   MKPDUs Transmitted...... 89596
      "Distributed SAK".. 1
      "Distributed CAK".. 0
```

The following is sample output from the **show mka summary** command:

```
Device# show mka summary

Total MKA Sessions....... 1
      Secured Sessions... 1
      Pending Sessions... 0

================================================================================
Interface      Local-TxSCI         Policy-Name      Inherited
```

```
Key-Server
Port-ID        Peer-RxSCI          MACsec-Peers     Status             CKN
=============================================================================
Gi1/0/1        204c.9e85.ede4/002b p2               NO                 YES
43             c800.8459.e764/002a 1                 Secured
01000000000000000000000000000000000000000000000000000000000000000000


MKA Global Statistics
=====================
MKA Session Totals
   Secured..................... 1
   Reauthentication Attempts.. 0

   Deleted (Secured).......... 0
   Keepalive Timeouts......... 0

CA Statistics
   Pairwise CAKs Derived...... 0
   Pairwise CAK Rekeys........ 0
   Group CAKs Generated....... 0
   Group CAKs Received........ 0

SA Statistics
   SAKs Generated............. 1
   SAKs Rekeyed............... 0
   SAKs Received.............. 0
   SAK Responses Received..... 1

MKPDU Statistics
   MKPDUs Validated & Rx...... 89589
      "Distributed SAK"..... 0
      "Distributed CAK"..... 0
   MKPDUs Transmitted......... 89600
      "Distributed SAK"..... 1
      "Distributed CAK"..... 0

MKA Error Counter Totals
========================
Session Failures
   Bring-up Failures.............. 0
   Reauthentication Failures........ 0
   Duplicate Auth-Mgr Handle........ 0

SAK Failures
   SAK Generation................. 0
   Hash Key Generation............ 0
   SAK Encryption/Wrap............ 0
   SAK Decryption/Unwrap.......... 0
   SAK Cipher Mismatch............ 0
```

```
CA Failures
   Group CAK Generation..... 0
   Group CAK Encryption/Wrap..... 0
   Group CAK Decryption/Unwrap..... 0
   Pairwise CAK Derivation..... 0
   CKN Derivation..... 0
   ICK Derivation..... 0
   KEK Derivation..... 0
   Invalid Peer MACsec Capability... 0
MACsec Failures
   Rx SC Creation..... 0
   Tx SC Creation..... 0
   Rx SA Installation..... 0
   Tx SA Installation..... 0

MKPDU Failures
   MKPDU Tx..... 0
   MKPDU Rx Validation..... 0
   MKPDU Rx Bad Peer MN..... 0
   MKPDU Rx Non-recent Peerlist MN.. 0
```

Additional References for MACsec Encryption

Standards and RFCs

Standard/RFC	Title
IEEE 802.1AE-2006	*Media Access Control (MAC) Security*
IEEE 802.1X-2010	*Port-Based Network Access Control*
IEEE 802.1AEbw-2013	*Media Access Control (MAC) Security (Amendment to IEEE 802.1AE-2006)—Extended Packet Numbering (XPN)*
IEEE 802.1Xbx-2014	*Port-Based Network Access Control (Amendment to IEEE 802.1X-2010)*
RFC 4493	*The AES-CMAC Algorithm*

Technical Assistance

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

Feature History for MACsec Encryption

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

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

Release	Feature	Feature Information
Cisco IOS XE 17.13.1	Certificate-based MACsec encryption	Support for this feature was introduced for the Cisco Catalyst ESS9300 Embedded Series Switch in this release.
Cisco IOS XE Cupertino 17.8.x	MACsec encryption	MACsec is the IEEE 802.1AE standard for authenticating and encrypting packets between two MACsec-capable devices. Support for this feature was introduced for Cisco Catalyst IE9300 Rugged Series Switches in this release.

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

Overview of PQC for MACsec with EAP-TLS

Post Quantum Cryptography (PQC) describes cryptographic algorithms that remain secure against attacks from quantum computers. Quantum computers can break many classical encryption methods. Widely used encryption methods—such as Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH)—are vulnerable as quantum computing technology advances.

Implementing quantum-safe encryption now is critical for the long-term confidentiality and integrity of sensitive communications, particularly for government, military, and financial sectors.

PQC for MACsec with EAP-TLS

MACsec encryption with EAP-TLS authentication establishes secure communication between switches by using certificate-based mutual authentication to derive keys for MACsec encryption. This process ensures that data traffic on the MACsec link is encrypted and decrypted and maintains integrity and confidentiality throughout the session.

Post Quantum Cryptography for MACsec with EAP-TLS provides line-rate Layer 2 encryption with quantum-safe mutual authentication. This section provides the configuration information for MACsec with EAP-TLS sessions.

Key elements of PQC

PQC is implemented by using the algorithms available as part of Module-lattice based Cryptography. Module-lattice based Cryptography is a quantum-resistant public-key cryptography that uses structured, high-dimensional grid points called lattices to secure data. This approach combines module-lattice based Cryptography (ML-KEM) with an elliptic curve function such as p256.

This combination provides enhanced security by protecting against both classical and quantum attacks while maintaining compatibility and performance with existing cryptographic protocols.

This table lists the supported algorithms and the areas in which each algorithm can be used.

Table 10: Key elements of PQC

Algorithm	What this algorithm means	Security level
• mlkem512 • mlkem768 • mlkem1024	ML-KEM is a PQC algorithm based on module-lattice cryptography, providing resistance against quantum attacks.	The security level of this implementation is balanced and is best suited for deployments where performance metrics are critical.
• p256_mlkem512 • p384_mlkem768 • p521_mlkem1024	ML-KEM is a PQC algorithm based on module-lattice cryptography, providing resistance against quantum attacks. p256, p384, and p521 refer to the traditional (classical) component. The algorithm uses Elliptic Curve Diffie-Hellman (ECDH) with the NIST P-256, P-384, or P-521 curves to maintain security against current non-quantum threats.	The security level of this implementation is the highest and is best suited for deployments where maximum security is needed.

Algorithm	What this algorithm means	Security level
• x25519_mlkem512 • x448_mlkem768 • X25519MLKEM76	ML-KEM is a PQC algorithm based on module-lattice cryptography, providing resistance against quantum attacks. X448 and X25519 are widely used classical elliptic curve Diffie-Hellman key exchange algorithms. This hybrid PQC key exchange algorithm is used to secure key exchanges against quantum and classical threats in Cisco's PQC-enabled network protocols.	The security level of this implementation is the highest and is best suited for deployments where maximum security is needed.
• SecP256r1MLKEM768 • SecP384r1MLKEM1024	secp384r1 and secp521r1 are standard, classical Elliptic Curve Diffie-Hellman (ECDH) groups used for key exchange during the TLS 1.3 handshake. ML-KEM is a PQC algorithm based on module-lattice cryptography, providing resistance against quantum attacks.	The security level of this implementation is good and is best suited for most deployments.
• secp256r1 • secp384r1 • secp521r1	Secp256r1, secp384r1, and secp521r1 are standard, classical ECDH groups used for key exchange during the TLS 1.3 handshake. These are the standard, non-quantum-resistant cryptographic curves that the system uses to ensure interoperability and maintain security when PQC is not required or not supported by the peer.	If a PQC-compliant handshake cannot be established (for example, when the pqc-type is set to all or non-pqc), the system falls back to using these classic groups. This approach is best suited for deployments where PQC is not chosen.

PQC implementation in EAP-TLS

PQC in Extensible Authentication Protocol Transport Layer Security (EAP-TLS) is implemented through negotiated key-exchange configuration. TLS 1.3 is mandatory for this implementation. This implementation secures Ethernet traffic through MACsec encryption. It combines IEEE 802.1X port-based authentication and EAP-TLS certificates for mutual authentication and automated key derivation.

MACsec encryption with EAP-TLS secures switch-to-switch communication. Certificate-based mutual authentication derives cryptographic keys, which are managed by the MACsec Key Agreement (MKA) protocol to ensure data confidentiality and integrity across Ethernet interfaces.

MACsec encryption uses EAP-TLS authentication to establish secure communication between switches. Certificate-based mutual authentication is leveraged to derive keys for MACsec encryption.

Summary

The key components involved in the process are

- Switches (authenticator/supplicant): Systems that perform MACsec encryption and participate in 802.1X authentication, acting as either the authenticator (facilitates authentication) or the supplicant (seeks authentication).
- Authentication server: An entity that provides authentication services to an authenticator, verifying supplicant credentials and facilitating EAP-TLS communication.
- Certificate Authority (CA) server: Issues and manages digital certificates used for mutual authentication in EAP-TLS.
- EAP-TLS (Extensible Authentication Protocol-Transport Layer Security): The authentication method used for mutual authentication between the authentication server and the client (supplicant) using certificates.
- Master Session Key (MSK): A cryptographic key generated upon successful EAP-TLS authentication.
- Connectivity Association Key (CAK): Derived from the MSK, this key is used by the MACsec Key Agreement (MKA) protocol. CAK is used to derive Secure Association Key (SAK), which performs the actual encryption and authentication of data packets on the link.
- Connectivity Association Key Name (CKN): Derived from the EAP session ID. This name identifies the CAK.

Workflow

These stages describe how MACsec encryption using EAP-TLS authentication works:

1. Initiation: A supplicant switch initiates 802.1X port-based authentication on a physical Ethernet interface with an authenticator switch.
2. Mutual authentication (EAP-TLS): The authentication server and the supplicant switch perform mutual authentication using digital certificates via the EAP-TLS method. This requires both devices to have valid certificates issued by a trusted Certificate Authority.
3. Master session key generation: Upon successful EAP-TLS authentication, a Master Session Key (MSK) is generated.
4. Key derivation: The MSK is then used to derive the Connectivity Association Key (CAK), and the Connectivity Association Key Name (CKN) is derived from the EAP session ID.
5. MACsec Key Agreement (MKA): The derived CAK and CKN are utilized by the MKA protocol to establish and maintain secure MACsec encryption between the switches on the interface.

Result

This process enables robust MACsec encryption between two switches, ensuring data confidentiality and integrity on Ethernet interfaces through secure, certificate-based authentication and automated key management.

Prerequisites

This list specifies the prerequisites before you configure PQC for MACSec with EAP-TLS.

- Ensure MACsec is enabled on the authenticator's (the switch that facilitates authentication) interface and ensure that the supplicant (the switch that seeks authentication) is capable of MACsec.

Restrictions

This list provides the limitations for configuring PQC in MACsec with EAP-TLS.

- In Cisco IOS XE 26.1.1, PQC for EAP-TLS is not supported in Federal Information Processing Standards (FIPS) mode.
- PQC for EAP-TLS is not supported with TLS versions 1.0 and 1.2.
- Certificate Revocation List (CRL) revocation is not supported for MACsec with EAP-TLS.
- PQC is supported with only local authentication for MACsec EAP -TLS.

Configuration matrix for MACsec with EAP-TLS

The protocol and TLS version matrix lists the combinations required for PQC to operate as expected for EAP-TLS. In this matrix, both device 1 and device 2 are switches.

Device 1 protocol and version	Device 2 protocol and version	Negotiated TLS version	Authentication status
EAP-TLS – 1.3	EAP-TLS – 1.3	1.3	Authentication successful
EAP-TLS – all	EAP-TLS – all	1.3	Authentication successful
EAP-TLS-1.2	EAP-TLS-all	1.2	Authentication successful
EAP-TLS – 1.3	EAP-TLS – 1.2		Authentication failure
EAP-TLS – 1.0	EAP-TLS – 1.3		Authentication failure

Supported key-exchange type for the algorithms

The table specifies the key-exchange type supported for each algorithm. The table lists these algorithms used to verify whether PQC is supported on your platform.

- **All**: all PQC, non-PQC and hybrid algorithms are supported.
- **Hybrid**: combined PQC and non-PQC algorithms are supported.

- **Non-PQC**: these are the classic cryptographic algorithms without PQC.
- **PQC**: PQC algorithms.

Version or key-exchange-type	Classic	PQC	Hybrid	All
1.3	Supported	Supported	Supported	Supported
1.2	Supported	Not supported	Not supported	Supported
1.0	Supported	Not supported	Not supported	Supported

Workflow to set up PQC for MACsec with EAP-TLS

This section describes how an authenticator and a supplicant can complete EAP-TLS authentication using ML-KEM for the TLS key exchange. This process results in the establishment of a secure MACsec session.

The steps to set up this configuration are:

1. Enable EAP-TLS with support for PQC ML-KEM on the supplicant as well as the authenticator.
2. Install and trust valid client and server certificates on the respective devices.
3. Connect the supplicant to the authenticator's MACsec-enabled port.
4. Initiate the authentication process by sending traffic from the supplicant or enabling the interface.
5. Verify PQC for EAP-TLS sessions.

Configure EAP-TLS 1.3

Follow these steps to configure TLS 1.3. If the peer device supports TLS version 1.3, authentication succeeds through EAP-TLS 1.3; otherwise, authentication fails.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 2 Use the **access-session tls-version** *version* command to configures the TLS 1.3 version.

Example:

```
Switch (config)# access-session tls-version <1.3>
```

Sample configuration

```
Switch (config)# access-session tls-version ?
  1.0  Version 1.0
  1.2  Version 1.2
  1.3  Version 1.3
  all  All TLS version
```

By default, **all** is configured on a device. When both the devices support TLS 1.3, TLS 1.3 negotiation is successful. Else, an older version of TLS version is enabled. In this scenario, PQC does not work.

Configure PQC for EAP-TLS sessions

Follow these steps to enable PQC for EAP-TLS sessions.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 2 Use the **access-session pqc-type** *pqc* to configure the PQC type.

Example:

```
Switch (config)# access-session pqc-type pqc
```

Note
If a device does not support PQC, the system displays this warning message:

```
Switch(config)# access-session pqc-type pqc
Warning! This platform only supports non-pqc method
```

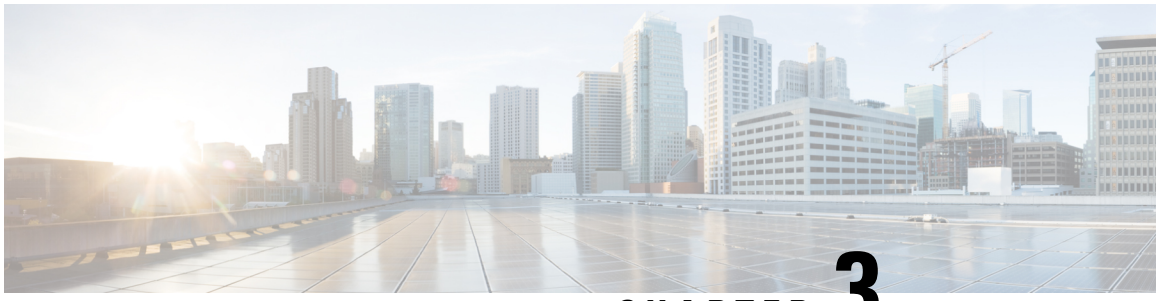
Sample configuration

```
Switch(config)# access-session pqc-type ?
  all        All pqc, non-pqc and hybrid algorithms will be supported
  hybrid     Combined PQC and NON-PQC algorithms (Key dervived with combination of both)
  non-pqc    Classic cryptographic algorithms
  pqc        Post-Quantum Cryptographic algorithms
```

Verify PQC for EAP-TLS sessions

Use these commands to verify whether PQC implementation is successful for EAP-TLS sessions.

Command	Purpose	What to look out for
Device(config)# access-session tls-version ?	Verify the TLS version	The TLS version should read 1.3 for this functionality to work.
show running-config all | i pqc-type	Verify the configured PQC type	
show dot1x interface <intf> detail	Verify the EAP profile is set up and the EAP method	The EAP method should read TLS.



CHAPTER 3

Post Quantum Cryptography for SSHv2 sessions

- Implementation of Post Quantum Cryptography, on page 105
- Overview of PQC for SSHv2 sessions, on page 105
- Factors to consider for deploying PQC, on page 106
- Protocol requirements, on page 106
- Restrictions for PQC, on page 106
- PQC negotiation scenarios , on page 106
- Configure PQC for SSHv2 sessions, on page 107
- Verify PQC for SSHv2 sessions, on page 108
- Key elements of PQC, on page 108

Implementation of Post Quantum Cryptography

Cisco utilizes a hybrid approach to implement Post Quantum Cryptography (PQC). The hybrid approach to PQC combines a quantum-resistant algorithm, such as the module-lattice based ML-KEM (e.g., mlkem768), with a classical elliptic curve key exchange algorithm like NIST P-256 or X25519, along with a hash function such as SHA-256.

This combination provides enhanced security by protecting against both classical and quantum attacks while maintaining compatibility and performance with existing cryptographic protocols.

Cisco implements this hybrid approach in protocols like SSH allowing configurations that enforce the use of ML-KEM algorithms for quantum resistance without falling back to classical-only key exchanges.

Overview of PQC for SSHv2 sessions

PQC refers to cryptographic algorithms designed to be secure against attacks from quantum computers which can break many classical encryption methods. Current widely used encryption methods—such as Diffie-Hellman (DH), and Elliptic Curve Diffie-Hellman (ECDH) — are vulnerable to being broken as quantum computing technology advances.

Implementing quantum-safe encryption now is critical for the long-term confidentiality and integrity of sensitive communications, particularly for government, military, and financial sectors.

Factors to consider for deploying PQC

Review these factors before deploying PQC.

Backward compatibility with classical algorithms
If a device is only configured for PQC and the SSH client does not support PQC, the SSH session will fail and no connection will be established.
Impact on computational time
The key size in PQC implementations may have an impact on the computational time due to the mathematical complexity and processing requirements of quantum-resistant algorithms. Larger key sizes generally increase the computational workload. PQC implementations must consider hardware acceleration and software optimizations to mitigate the increased computational time due to larger key sizes.

Protocol requirements

Protocol requirements

SSHv2 :

To establish secure, encrypted, and remote command-line access to servers and network devices over unsecured network use SSH version 2.0. PQC cannot be implemented on SSHv1 due to severe security vulnerabilities and it does not support the modern, complex key exchange mechanisms required to resist quantum computer attacks.

Key Pairs :

Ensure that RSA key pair or EC key pair for the SSH server is generated using crypto key generate command.

Restrictions for PQC

Implementing PQC is not supported on devices that operate in Federal Information Processing Standards (FIPS) mode.

PQC negotiation scenarios

During the SSH protocol connection setup, the client and server negotiate which cryptographic algorithms to use for securing the session. This negotiation process is a basic example of how algorithm negotiation works in SSH and is applicable to any similar negotiation process.

The algorithms are considered in the order they are configured on the device, meaning the priority or preference of algorithms is determined by the sequence in which they are listed in the configuration. The negotiation proceeds by trying the algorithms in that configured order until a mutually supported algorithm is found and agreed upon for use in the session. The server must support at least one algorithm from the client's list for successful negotiation.

Table 11: sfds

Client configuration	Server configuration	Negotiated algorithm

Classical, Hybrid	Classical, Hybrid	Classical
Hybrid, Classical	Hybrid, Classical	Hybrid
Classical, Hybrid	Hybrid	Hybrid
Hybrid, Classical	Classical	Classical
Classical	Classical, Hybrid	Classical
Hybrid	Classical, Hybrid	Hybrid
Classical	Hybrid	None
Hybrid	Classical	None
Mixed (No overlap)	Mixed (No overlap)	None

Configure PQC for SSHv2 sessions

Use these commands to set up devices as an SSH server and an SSH client. This allows the device to accept connection requests from remote SSH clients, enabling secure remote management of the device through SSH connections.

Note The PQC algorithms are not enabled by default and must be explicitly enabled.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 2 Use the **ip ssh server algorithm kex** *algorithm* command to configure the PQC based key-exchange algorithms that the SSH server supports.

Example:

```
Switch(config)# ip ssh server algorithm kex mlkem768nistp256-sha256
```

Step 3 Use the **ip ssh client algorithm kex** *algorithm* command to configure the PQC based key-exchange algorithms that SSH client supports.

Example:

```
Switch(config)# ip ssh client algorithm kex mlkem768x25519-sha256
```

Verify PQC for SSHv2 sessions

Use this section to understand how to verify PQC implementation for SSHv2.

Command	Purpose	What to look for in output
show ip ssh	Displays the status of the SSH server, its configuration details.	In the output of this command, look for these details that indicate PQC implementation is successful: KEX algorithms: **mlkem1024nistp384-sha384,** **mlkem768nistp256-sha256,** **mlkem768x25519-sha256**

This example shows the response when a PQC algorithm command is entered on a platform that does not support PQC.

```
Switch(config)# ip ssh server algorithm kex mlkem1024nistp384-sha384 ecdh-sha2-nistp256
Switch(config)#
000205: Jun 18 16:41:43.086: %SSH-3-KEX_PQC_NOT_SUPP: PQ/T hybrid key exchange algorithms
are not supported on this platform.
Switch(config)#
```

Key elements of PQC

PQC is implemented by using the algorithms available as part of Module-lattice based Cryptography. Module-lattice based Cryptography is a type of quantum-resistant public-key cryptography that uses structured, high-dimensional grid points (lattices) to secure data. Cisco implements PQC into protocols like SSHusing Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM).

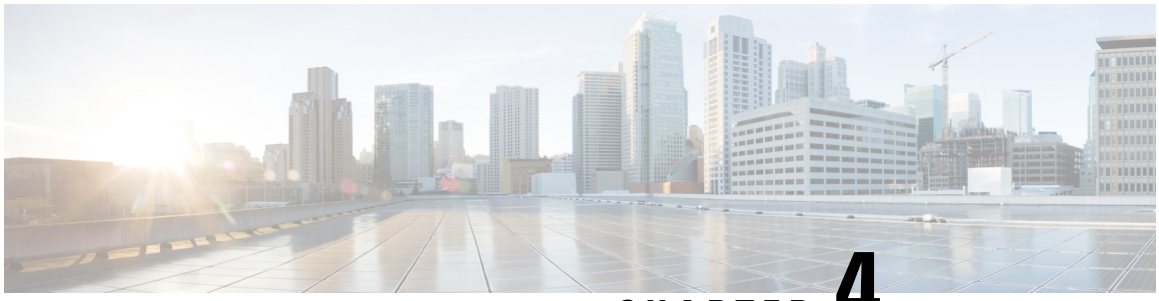
These algorithms are resistant to quantum attacks and are used for secure key establishment, encryption, and digital authentication or signatures.

This table lists the areas in which each algorithm can be used:

Table 12: Key elements of PQC

ML-KEM algorithm	What this algorithm means	Security level
mlkem768x25519-sha256	ML-KEM 768 is a PQC algorithm based on module-lattice cryptography, providing resistance against quantum attacks. X25519 is a widely used classical elliptic curve Diffie-Hellman key exchange algorithm. SHA-256 is the hash function used for cryptographic hashing in the key exchange process. In summary, mlkem768x25519-sha256 is a hybrid PQC key exchange algorithm combining the ML-KEM 768-bit lattice-based scheme with classical X25519 and SHA-256, used to secure key exchanges against quantum and classical threats in Cisco's PQC-enabled network protocols	The security level of this implementation is balanced and is best suited for deployments where performance metrics are critical.
mlkem768nistp256-sha256	ML-KEM 768 is a PQC algorithm based on module-lattice cryptography, providing resistance against quantum attacks. NIST P-256 is a widely used classical elliptic curve Diffie-Hellman key exchange algorithm standardized by NIST. SHA-256 is the hash function used for cryptographic hashing in the key exchange process. In summary, mlkem768x25519-sha256 is a hybrid PQC key exchange algorithm combining the ML-KEM 768-bit lattice-based scheme with classical X25519 and SHA-256, used to secure key exchanges against quantum and classical threats in Cisco's PQC-enabled network protocols	The security level of this implementation is good and is best suited for most deployments.

ML-KEM algorithm	What this algorithm means	Security level
mlkem1024nistp384-sha384	ML-KEM 768 is a PQC algorithm based on module-lattice cryptography, providing resistance against quantum attacks. 1024 is the highest security parameter set, equivalent to the security level of AES-256. NIST P-384 is the traditional (classical) component. It uses Elliptic Curve Diffie-Hellman (ECDH) with the NIST P-384 curve to maintain security against current non-quantum threats. In summary, mlkem1024nistp384-sha384 represents a hybrid cryptographic proposal combining a quantum-resistant ML-KEM algorithm with classical elliptic curve cryptography (NIST P-384) and SHA-384 hashing. This hybrid approach ensures security against both classical and quantum attacks by negotiating keys using both classical and post-quantum algorithms.	The security level of this implementation is the highest and is best suited for deployments where maximum security is needed.



CHAPTER 4

Network Edge Access Topology

- 802.1x supplicant and authenticator switches with network edge access topology, on page 111
- Guidelines and limitations for NEAT configuration, on page 113
- Configure an authenticator switch with NEAT, on page 113
- Configure a supplicant switch with NEAT, on page 115
- Configuration verification, on page 117
- Feature History, on page 119

802.1x supplicant and authenticator switches with network edge access topology

An 802.1x supplicant and authenticator switch with network edge access topology is a LAN access control solution that

- uses the 802.1x standard[1] to authenticate clients before granting access to LAN services
- extends identity and authentication outside the wiring closet using Network Edge Access Topology (NEAT), and
- supports secure connectivity and trunk configuration for devices connected through layer 2 ports.

[1]: The 802.1x standard defines a client-server-based access control and authentication protocol that prevents unauthorized clients from connecting to a LAN through publicly accessible ports unless they are properly authenticated.

Network edge access topology features

The NEAT feature allows any device to authenticate on the port, propagating client MAC and VLAN information between supplicant and authenticator switches using CISP. NEAT and CISP are supported only on layer 2 ports, not on layer 3 ports. NEAT can be configured on Cisco Catalyst IE9300 Rugged series Switches.

- 802.1x switch supplicant: Configure a switch to act as a supplicant to another switch using the 802.1x supplicant feature. This is useful when a switch is outside a wiring closet and connected to an upstream switch through a trunk port. The supplicant switch authenticates with the upstream switch for secure connectivity. After successful authentication, the port mode changes from access to trunk in the authenticator switch. On the supplicant switch, manually configure the trunk when enabling CISP.

- If the access VLAN is configured on the authenticator switch, it becomes the native VLAN for the trunk port after successful authentication.

When connecting a supplicant switch to an authenticator switch with BPDU guard enabled, the authenticator port may be error-disabled if it receives STP BPDU packets before authentication. You can control traffic exiting the supplicant port during authentication. Entering the **dot1x supplicant controlled transient** global configuration command temporarily blocks the supplicant port during authentication to prevent the authenticator port from shutting down before authentication completes. If authentication fails, the supplicant port opens. Entering the **no dot1x supplicant controlled transient** global configuration command opens the supplicant port during authentication, which is the default behavior.

Use the **dot1x supplicant controlled transient** command on a supplicant switch when BPDU guard is enabled on the authenticator switch port with the **spanning-tree bpduguard enable** interface configuration command.

Note If you globally enable BPDU guard on the authenticator switch by using the **spanning-tree portfast bpduguard default** global configuration command, entering the **dot1x supplicant controlled transient** command on the supplicant switch does not prevent the BPDU violation.

You can enable MDA or multiauth mode on the authenticator switch interface that connects to one or more supplicant switches. Multihost mode is not supported on the authenticator switch interface.

When you reboot an authenticator switch with single-host mode enabled on the interface, the interface may move to err-disabled state before authentication. To recover from err-disabled state, flap the authenticator port to activate the interface again and initiate authentication.

Use the **dot1x supplicant force-multicast** global configuration command on the supplicant switch for NEAT to work in all host modes.

- Host authorization: Ensures that only traffic from authorized hosts connecting to the switch with supplicant is allowed on the network. The switches use CISP to send the MAC addresses connecting to the supplicant switch to the authenticator switch.
- Auto enablement: Automatically enables trunk configuration on the authenticator switch, allowing user traffic from multiple VLANs coming from supplicant switches. Configure the cisco-av-pair as *device-traffic-class=switch* at the ISE. (You can configure this under the *group* or the *user* settings.)

Figure 5: Authenticator and supplicant switch using CISP

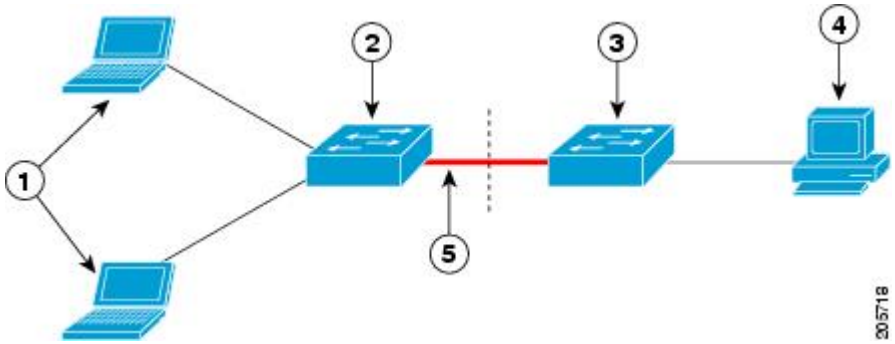


Table 13: Network edge access topology components

1	Workstations (clients)
2	Supplicant switch (outside wiring closet)
3	Authenticator switch
4	Cisco ISE
5	Trunk port

Note The **switchport nonegotiate** command is not supported on supplicant and authenticator switches with NEAT. This command should not be configured at the supplicant side of the topology. If configured on the authenticator side, the internal macros automatically remove this command from the port.

For more information about 802.1x, including configuration information, see Configuring IEEE 802.1x Port-Based Authentication.

Guidelines and limitations for NEAT configuration

Guidelines and limitations

- A Radius server such as Cisco's Identity Server Engine (ISE) is required.
- CISP and NEAT are supported only on L2 ports, not on L3 ports.
- NEAT and 802.1x are not supported on EtherChannel ports.
- NEAT is not supported on dynamic ports.
- MACsec is supported with NEAT.
- NEAT can operate with PTP.
- MAB and NEAT are mutually exclusive. You cannot enable MAB when NEAT is enabled on an interface, and you should not enable NEAT when MAB is enabled on an interface.

Configure an authenticator switch with NEAT

This task enables you to configure a switch as an authenticator using NEAT, allowing secure network access for devices connected outside the wiring closet.

- Ensures that the switch can authenticate connected supplicant switches and dynamically set trunk interfaces after authentication.

Configuring this feature requires that one switch outside a wiring closet is configured as a supplicant and is connected to an authenticator switch.

Note

- The *cisco-av-pairs* must be configured as *device-traffic-class=switch* on the ISE, which sets the interface as a trunk after the supplicant is successfully authenticated.

This configuration is typically performed when you need to extend authentication services to switches located outside the main wiring closet, ensuring secure access and dynamic interface configuration.

Before you begin

Before you begin, ensure you have access to the switch in privileged EXEC mode and that the supplicant switch is physically connected to the authenticator switch.

- Verify that the ISE is configured with the required *cisco-av-pairs* for device-traffic-class.

Follow these steps to configure an authenticator switch with NEAT:

Procedure

Step 1 Use the **enable** command to enables privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

Step 2 Use the **configure terminal** command to enters global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **CISP enable** command to enable CISP.

Example:

```
Switch(config)# cisp enable
```

Step 4 Use the **interface** *interface-id* command to specify the port to be configured, and enters interface configuration mode.

Example:

```
Switch(config)# interface gigabitethernet 1/0/2
```

Step 5 Use the **switchport mode access**command to sets the port mode to **access**.

Example:

```
Switch(config-if)# switchport mode access
```

Sets the port mode to **access**.

Step 6 Use the **authentication port-control auto** command to set the port-authentication mode to auto.

Example:

```
Switch(config-if)# authentication port-control auto
```

Step 7 Use the **dot1x PAE authenticator** command to configure the interface as a port access entity (PAE) authenticator.

Example:

```
Switch(config-if)# dot1x pae authenticator
```

Step 8 Use the **spanning-tree portfast** command to enable the interface to quickly transition to spanning-tree forwarding state for an interface which is a member of multiple VLANs.

Example:

```
Switch(config-if)# spanning-tree portfast trunk
```

Use this command only when you are sure that the switch-to-switch connection is not part of a layer2 loop.

Step 9 Use the **end** command to exit interface configuration mode and returns to privileged EXEC mode.

Example:

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

Configure a supplicant switch with NEAT

This task configures a switch as a supplicant using NEAT, enabling secure authentication and trunking for network access.

- Ensures the switch can participate in 802.1X authentication as a supplicant.
- Allows NEAT to operate on the supplicant switch in all host modes.

Use this task when you need to configure a switch to act as a supplicant in a NEAT-enabled environment.

Beginning in privileged EXEC mode, follow these steps to configure a switch as a supplicant:

- Applicable for environments requiring secure port-based authentication and trunking.

Before you begin

Ensure you are in privileged EXEC mode before starting the configuration.

- Have access to the switch console.

Follow these steps to configure a supplicant switch with NEAT:

Procedure

Step 1 Global configuration modes

a) Use the **enable** comamnd to enables privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

b) Use the **configure terminal** command to enters global configuration mode.

Example:

```
Switch# configure terminal
```

Step 2 Configure CISP, EAP profile, and 802.1X credentials

a) Use the **CISP enable** command to enable CISP.

Example:

```
Switch(config)# cisp enable
```

b) Use the **EAP profile** *profile-name* command to create an Extensible Authentication Protocol (EAP) profile and enters EAP profile configuration mode.

Example:

```
Switch(config)# eap profile CISP
```

c) Use the **method** *type* command to specify the EAP authentication method.

Example:

```
Switch(config-eap-profile)# method md5
```

d) Use the **exit** command to exit EAP profile configuration mode.

Example:

```
Switch(config-eap-profile)# exit
```

e) Use the **dot1x credentials** *profile* command to create 802.1x credentials profile. This must be attached to the port that is configured as supplicant.

Example:

```
Switch(config)# dot1x credentials test
```

f) Use the **username** *suppswitch*command to create a username.

Example:

```
Switch(config)# username suppswitch
```

g) Use the **password** *password* command to creates a password for the new username.

Example:

```
Switch(config)# password myswitch
```

h) Use the **dot1x supplicant force-multicast** command to force the switch to send only multicast EAPOL packets when it receives either unicast or multicast packets.

Example:

```
Switch(config)# dot1x supplicant force-multicast
```

This command also allows NEAT to work on the supplicant switch in all host modes.

Step 3 Interface configuration

a) Use the **interface** *interface-id* command to specify the port to be configured, and enters interface configuration mode.

Example:

```
Switch(config)# interface gigabitethernet1/0/1
```

b) Use the **switchport trunk encapsulation dot1q** command to set the port to trunk mode.

Example:

```
Switch(config-if)# switchport trunk encapsulation dot1q
```

c) Use the **switchport mode trunk** command to configure the interface as a VLAN trunk port.

Example:

```
Switch(config-if)# switchport mode trunk
```

d) Use the **dot1x PAE supplicant** command to configure the interface as a port access entity (PAE) supplicant.

Example:

```
Switch(config-if)# dot1x pae supplicant
```

e) Use the **dot1x credentials** *profile-name* command to attach the 802.1x credentials profile to the interface.

Example:

```
Switch(config-if)# dot1x credentials test
```

f) Use the **dot1x supplicant EAP profile** *profile-name* command to assign the EAP-TLS profile to the 802.1X interface.

Example:

```
Switch(config-if)# dot1x supplicant eap profile cisp
```

g) Use the **end** command to exit interface configuration mode and returns to privileged EXEC mode.

Example:

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

Configuration verification

Use show and debug commands to verify and troubleshoot Client Information Signalling Protocol (CISP) and Network Edge Access Topology (NEAT) configuration.

Use these show commands to verify information about Client Information Signalling Protocol (CISP) and Network Edge Access Topology (NEAT) configuration:

- show CISP interface <interface name>
- show CISP clients
- show CISP summary
- show CISP registrations

This is an example for **show CISP** commands. GigabitEthernet 1/0/1 is configured as Authenticator, and GigabitEthernet 1/0/2 is configured as Supplicant.

```
Auth# show cisp interface Gi1/0/2

CISP Status for interface Gi1/0/2
 ---------------------------------------
Version: 1
Mode: Supplicant Peer
Mode: Authenticator
Supp State: Idle

Auth# <userinput>show cisp clients</userinput>

Authenticator Client Table:
```

```
 ------------------------
 MAC Address VLAN Interface
 --------------------------------
 0050.5695.4de8 1 Gi1/0/10
6c03.09e7.3947 1 Gi1/0/10
6c03.09e7.3954 11 Gi1/0/10
6c03.09e7.4485 1 Gi1/0/10
 9077.ee4a.8567 1 Gi1/0/10
e41f.7ba1.bbd4 1 Gi1/0/10

Supplicant Client Table:
------------------------
MAC Address VLAN Interface
--------------------------------
9077.ee4a.856b 11 Vl11
9077.ee4a.8572 1 Ap1/1
e41f.7bc7.2f03 1 Gi1/0/9

Auth# show cisp summary

CISP is running on the following interface(s):
---------------------------------------------
Gi1/0/2 (Authenticator)

Supp# show cisp summary

CISP is running on the following interface(s):
---------------------------------------------
Gi1/0/1 (Supplicant)

Auth# show cisp registrations

Interface(s) with CISP registered user(s):
-----------------------------------------
Gi1/0/2
Auth Mgr (Authenticator)

Supp# show cisp registration

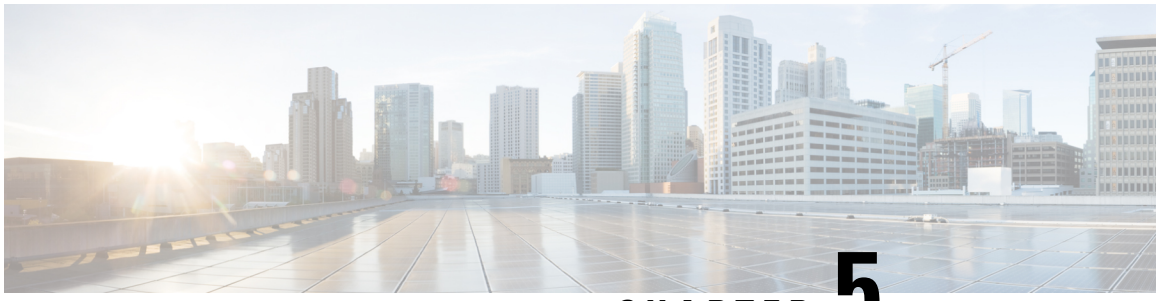
Interface(s) with CISP registered user(s):
-----------------------------------------
Gi1/0/1
802.1x Sup (Supplicant)
```

Use these debug commands to troubleshoot CISP and NEAT:

- debug access-session errors
- debug access-session event
- debug dot1x errors
- debug dot1x packets
- debug dot1x events

Feature History

Feature Name	Release	Feature Information
Network Edge Access Topology (NEAT)	Cisco IOS XE 17.8.1	Initial support on Cisco Catalyst IE9300 Rugged Series Switches.



CHAPTER 5

FlexLink+

- FlexLink+ redundancy functions, on page 121
- Guidelines and restrictions, on page 122
- Configure active port for Flex Link+, on page 122
- Configure standby port for FlexLink+, on page 123
- Configure VLAN load balancing on FlexLink+, on page 124
- Configure time for VLAN load balancing on FlexLink+, on page 125
- Configure manual preemption for VLAN load balancing on FlexLink+, on page 125
- Configure FlexLink+ topology change messages, on page 126

FlexLink+ redundancy functions

FlexLink+ is a Layer 2 redundancy feature that

- provides a pair of interfaces where one acts as a backup to the other
- offers an alternative to the Spanning Tree Protocol (STP) for simple link redundancy, and
- supports switchover on both physical interfaces and EtherChannel bundles.

FlexLink+ provides redundancy by using one Layer 2 link as primary and another as backup. It can also improve bandwidth through VLAN load balancing, letting both links forward traffic at the same time for different VLANs while still maintaining redundancy.

In load balancing mode, both ports forward traffic for separate VLANs simultaneously, which boosts bandwidth and maintains redundancy.

Table 14: Feature history

Feature name	Release information	Description
FlexLink+	26.2.1	This feature provides link redundancy between Layer 2 interfaces without requiring Spanning Tree Protocol. It supports failover for active-standby configurations, VLAN load balancing across links, automatic or manual preemption, and physical interfaces or port channels. FlexLink+ is ideal for industrial networks requiring simple, reliable uplink redundancy with minimal configuration complexity.

Guidelines and restrictions

Guidelines

- Use the **spanning-tree portfast trunk** command on uplink interfaces connected to FlexLink+ ports to bypass STP listening and learning states.
- Enable either MSTP or Rapid PVST on upstream switches for networks that extend across a FlexLink+ node to prevent traffic blackholing. For more information abour configuring MSTP, refer Configuring Multiple Spanning-Tree Protocol and for more information about configuring Rapid PVST, refer Spanning-Tree Interoperability and Backward Compatibility.

Restrictions

- FlexLink+ supports physical Layer 2 interfaces and EtherChannel.
- VLAN load balancing requires the edge port to terminate the segment.

Configure active port for Flex Link+

Define the primary (active) interface in a FlexLink+ pair.

Before you begin

Configure the node connected to the FlexLink+ active link as the root bridge.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 2 Use the **interface** *interface-id* command to enter interface configuration mode.

Example:

```
Switch(config)# interface GigabitEthernet1/0/1
```

Step 3 Use the **switchport trunk allowed vlan** *vlan-list* command to configure the allowed VLANs on the interface.

Example:

```
Switch(config-if)# switchport trunk allowed vlan 10 20
```

Step 4 Use the **switchport mode trunk** command to set the interface to trunk mode.

Example:

```
Switch(config-if)# switchport mode trunk
```

Step 5 Use the **rep segment** *segment-id* **edge no-neighbor primary** command to define the port as the primary edge port.

Example:

```
Switch(config-if)# rep segment 2 edge no-neighbor primary
```

Configure standby port for FlexLink+

Define the standby interface in a FlexLink+ pair.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 2 Use the **interface** *interface-id* command to specify the interface.

Example:

```
Switch(config)# interface GigabitEthernet1/0/1
```

Step 3 Use the **switchport trunk allowed vlan** *vlan-list* command to configure the allowed VLANs on the interface.

Example:

```
Switch(config-if)# switchport trunk allowed vlan 10 20
```

Step 4 Use the **switchport mode trunk** command to set the interface to trunk mode.

Example:

```
Switch(config-if)# switchport mode trunk
```

Step 5 Use the **rep segment** *segment-id* **edge no-neighbor preferred** command to define the port as the standby edge port.

Example:

```
Switch(config-if)# rep segment 2 edge no-neighbor preferred
```

The **preferred** keyword influences which port becomes the blocking port, but the system selects the actual blocking port according to the timing of link state changes, such as when a port comes up later.

Configure VLAN load balancing on FlexLink+

This procedure enables traffic distribution across a FlexLink+ pair.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 2 Use the **interface** *interface-id* command to enter interface configuration mode.

Example:

```
Switch(config)# interface GigabitEthernet1/0/1
```

Step 3 Use the **rep block port** *port-number* **vlan** *vlan-id* command to define the VLANs that are blocked on the standby port.

Example:

```
Switch(config-if)# rep block port 2 vlan 1-50
```

This command blocks VLANs 1 to 50 on the standby port, which means these VLANs remain active on the primary port.

Step 4 Use the **switchport mode trunk** command to set the interface to trunk mode.

Example:

```
Switch(config-if)# switchport mode trunk
```

Step 5 Use the **exit** command to exit interface configuration mode.

Example:

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

Step 6 Use the **interface** *interface-id* command to access the standby interface configuration.

Example:

```
Switch(config)# interface GigabitEthernet1/1
```

Step 7 Use the **rep segment** *segment-id* **edge no-neighbor** command to confirm the standby interface role in the segment.

Example:

```
Switch(config)# rep segment 10 edge no-neighbor
```

Configure time for VLAN load balancing on FlexLink+

You can configure the time to trigger VLAN load balancing automatically after a link recovers. For more information about preempt delay, see VLAN load balancing.

This configuration is valid on the primary edge port only.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 2 Use the **interface** *interface-id* command to enter interface configuration mode.

Example:

```
Switch(config)# interface GigabitEthernet1/0/1
```

Step 3 Use the **rep preempt delay** *seconds* command to configure the preemption delay time.

Example:

```
Switch(config-if)# rep preempt delay 100
```

The valid range for the delay value is 15 to 300 seconds.

Configure manual preemption for VLAN load balancing on FlexLink+

Manual preemption defines the process for manually restoring the configured VLAN load balancing state on a segment.

Configure manual preemption as described in the following procedure. For more information about VLAN load balancing using manual preemption, see VLAN load balancing.

Procedure

Step 1 Use the **enable** command to enter privileged EXEC mode.

Example:

```
Switch> enable
```

Step 2 Use the **rep preempt segment** *segment-id* command to manually trigger VLAN load balancing preemption for the specified segment.

Example:

```
Switch# rep preempt segment 10
```

Confirm the command when prompted to proceed with the preemption action.

Configure FlexLink+ topology change messages

Notify upstream devices of topology changes within the FlexLink+ segment.

Configure FlexLink+ in a larger network domain to ensure traffic convergence.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 2 Use the **interface** *interface-id* command to enter interface configuration mode.

Example:

```
Switch(config)# interface GigabitEthernet1/0/1
```

Step 3 Use the **rep stcn stp** command to enable propagation of topology change notifications on the active interface.

Example:

```
Switch(config-if)# rep stcn stp
```

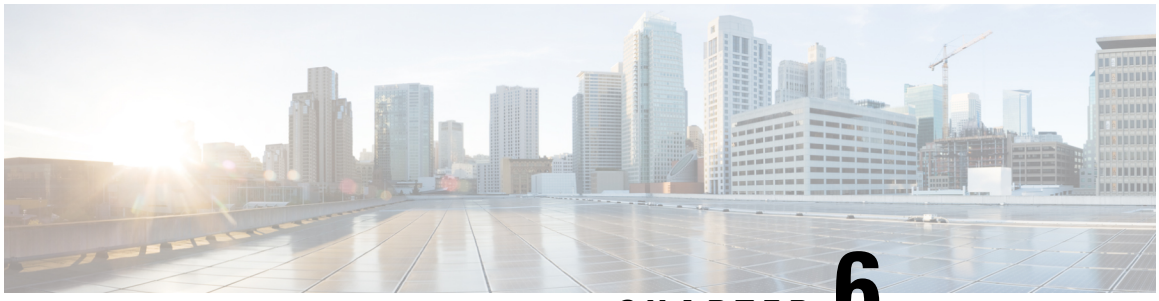
Configure the **rep stcn stp** command on the FlexLink+ interface to send topology change notifications to upstream STP, MSTP, and REP devices when a topology change occurs in the FlexLink+ segment. If you do not configure this command, upstream devices do not receive notifications about link failures or topology changes in the FlexLink+ segment, which can cause traffic blackholing.

Note
Enable this command on each interface configured for FlexLink+.

Step 4 Use the **exit** command to exit interface configuration mode.

Example:

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



CHAPTER 6

Wired dynamic PVLAN

- Wired dynamic PVLANs, on page 127
- Restrictions for wired dynamic PVLAN, on page 129
- Configure wired dynamic PVLAN, on page 129

Wired dynamic PVLANs

A wired dynamic PVLAN is a network isolation feature that

- uses a private VLAN with AAA authorization to isolate clients and provide Zero-Trust
- blocks peer-to-peer communication within a subnet or VLAN, and
- assigns a single wired data client per port interface to ensure point-to-point blocking.

PVLAN isolation and client communication

Hosts connected to Switch A can communicate only with the promiscuous trunk port on the switch. The PVLAN can be extended to span across multiple switches by adding intermediate switches. If there is a switch (Switch C) between Switch A and Switch B, layer 2 trunk ports need to be configured on the intermediate links. In the case of a community VLAN, packets can be seen on other hosts within the same community VLAN.

When a host is connected to a switch port with a cable, it is placed into an isolated PVLAN where it cannot discover any other hosts. The host is then authenticated by the RADIUS server. If the port is placed in closed mode and is not authenticated, only Extensible Authentication Protocol over LAN (EAPoL) packets are allowed. Once the port is authenticated, it is placed into an isolated VLAN dynamically.

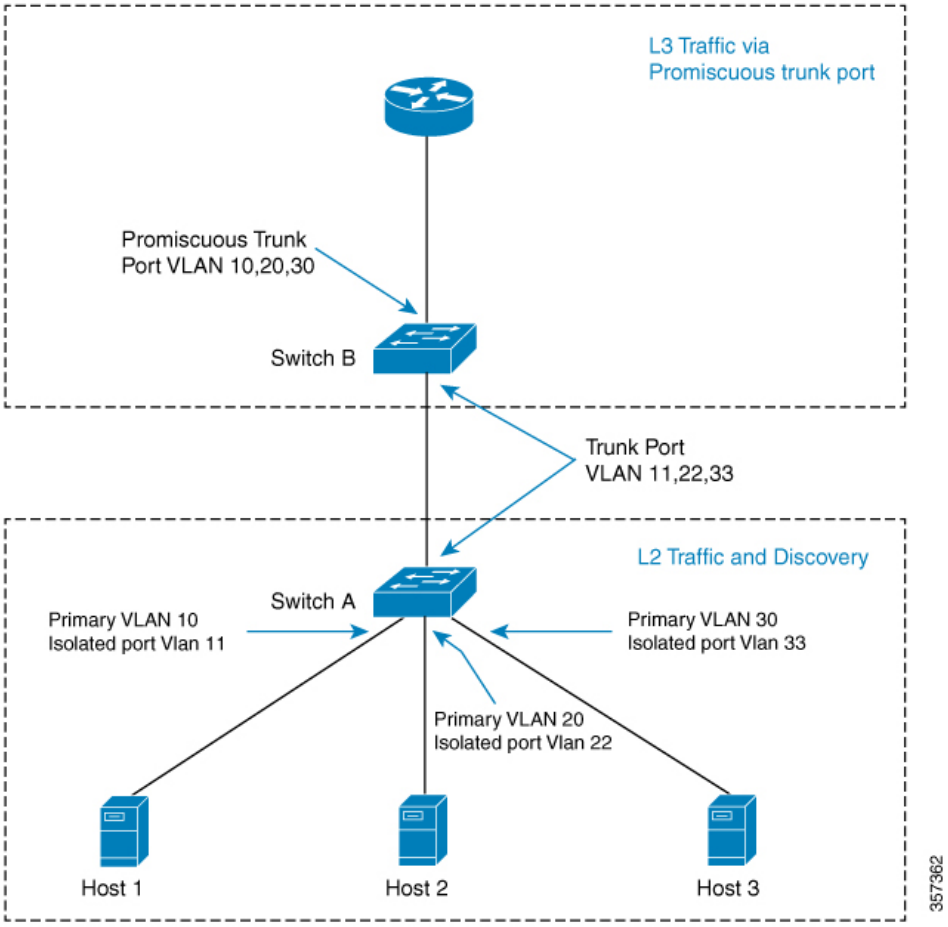
When the host first authenticates with the RADIUS server, it sends the name of a dynamic interface template to be applied to the host's port. This interface template contains the configurations to enable the PVLAN primary and secondary VLANs on the port. With the template applied to the host, the switchport mode changes, causing the port to flap from access mode to PVLAN mode.

Note Traffic from multiple clients on the same interface will not be blocked.

Note The interface template with the same name as referred by AAA Authorization needs to be configured on the switch.

When the interface template is applied, the port will physically go down for a time period set by the sticky timer and come up again. When the RADIUS server sends the interface template a second time, it is ignored as the conversion has been completed. The port is then assigned to a PVLAN which keeps it isolated. The host completes authorization and comes up to ready state.

Figure 6: PVLAN topology diagram



Configuring sticky timer for interface template

Configure the keep time for which the interface template information is retained before it is removed from the port using the **access-session interface-template sticky timer***time* command.

PVLAN isolation compared to private rooms

PVLAN isolation is similar to placing each client in a private room, where they cannot see or communicate with other clients except through a central access point.

Restrictions for wired dynamic PVLAN

- High availability is not supported with wired dynamic PVLAN.
- Voice VLAN configuration cannot co-exist with this feature.
- Local Web Authentication (LWA) and Central Web Authentication (CWA) cannot be used with this feature.
- All wired clients using the dynamic PVLAN interface template will be programmed as data clients.
- Only interfaces with existing Access or PVLAN Host switchport mode support PVLAN template.
- Identity Based Networking Services 2.0 (IBNS 2.0) must be used for dynamic template support.

Configure wired dynamic PVLAN

Configure wired dynamic PVLAN to segment network traffic and enhance security by isolating ports and associating VLANs. This task enables the creation and association of primary and secondary VLANs, and applies interface templates for authenticated clients.

- Establish isolated and primary VLANs for port isolation.
- Associate secondary VLANs with the primary VLAN.
- Apply interface templates for authenticated clients using RADIUS or ISE.

Use this task to configure wired dynamic PVLAN on a user switch, such as switch A in the topology. This configuration is relevant when you need to segment ports for security and apply dynamic templates based on client authentication.

Before you begin

Ensure that dot1x AAA is configured on the user switch.

Verify that the user switch supports PVLAN and interface templates

Procedure

Step 1 Configure private VLAN

a) Use the **enable** command to enter privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password if prompted.

b) Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

c) (Optional) Use the **VLAN** *vlan-id* command to create or select the isolated VLAN.

Example:

```
Switch(config)# vlan 200
```

The VLAN ID range is 2 to 1001 and 1006 to 4094.

d) Use the **private-VLAN isolated** command to designate the VLAN as an isolated VLAN.

Example:

```
Switch(config-vlan)# private-vlan isolated
```

e) Use the **exit** command to return to global configuration mode.

Example:

```
Switch(config-vlan)# exit
```

f) Use the **VLAN** *vlan-id* command to create or select the primary VLAN.

Example:

```
Switch(config)# vlan 100
```

g) Use the **private-VLAN primary** command to designate the VLAN as a primary VLAN.

Example:

```
Switch(config-vlan)# private-vlan primary
```

h) Use the **private-VLAN association** [**add** | **remove**] *secondary_vlan_list* command to associate secondary VLANs with the primary VLAN.

Example:

```
Switch(config-vlan)# private-vlan association 200
```

- The *secondary_vlan_list* parameter cannot contain spaces. It can contain multiple comma-separated items. Each item can be a single private-VLAN ID or a hyphenated range of private-VLAN IDs.
- The *secondary_vlan_list* parameter can contain multiple community VLAN IDs but only one isolated VLAN ID.
- Enter a *secondary_vlan_list*, or use the **add** keyword with a *secondary_vlan_list* to associate secondary VLANs with a primary VLAN.
- Use the **remove** keyword with a *secondary_vlan_list* to clear the association between secondary VLANs and a primary VLAN.
- The command does not take effect until you exit VLAN configuration mode.

i) Use the **exit** command to return to global configuration mode.

Example:

```
Switch(config-vlan)# exit
```

Step 2 Configure interface

a) Use the **template** *template-name* command to create a user template and enter template configuration mode.

Example:

```
Switch(config)# template PVLAN100_200_CFG
```

b) Use the **switchport mode private-VLAN host** command to configure a layer 2 port as a PVLAN host port on the template.

Example:

```
Switch(config-template)# switchport mode private-vlan host
```

c) Use the **switchport private-VLAN host-association** *primary_vlan_id secondary_vlan_id* command to configure the association of a layer 2 port with a PVLAN on the template.

Example:

```
Switch(config-template)# switchport private-vlan host-association 100 200
```

d) Use the **exit** command to return to global configuration mode.

Example:

```
Switch(config-template)# exit
```

e) Use the **access-session interface-template sticky timer** *time* command to configure the global sticky timer for the template.

Example:

```
Switch(config)# access-session interface-template sticky timer 60
```

Configures the keep time of the template globally. Once the last client leaves, the template will be removed from the port after the configured keep time.

Note
It is recommended that you set the sticky timer to 60 seconds.

f) Use the **interface** *interface-ID* command to enter interface configuration mode for the specified interface.

Example:

```
Switch(config)# interface GigabitEthernet1/0/1
```

g) Use the **access-session interface-template sticky timer** *time* command to configure the sticky timer on the specific interface.

Example:

```
Switch(config-if)# access-session interface-template sticky timer 60
```

Configures the keep time of the template on the interface. Once the last client leaves, the template will be removed from the port after the configured keep time.

Note
It is recommended that you set the sticky timer to 60 seconds.

h) Use the **end** command to return to privileged EXEC mode.

Example:

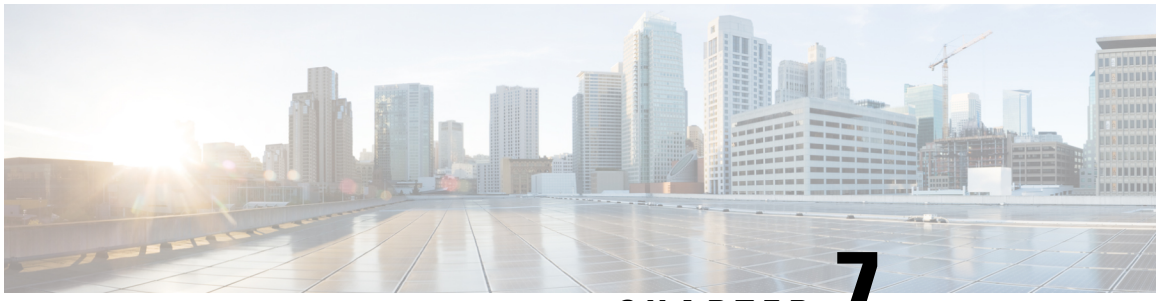
```
Switch(config-if)# end
```

What to do next

Once you comple the wired dynamic PVLAN configuration, configure the Identity Services Engine (ISE) or any other RADIUS server to assign the template to the client's port interface after the client has been authenticated successfully.

If you are using the ISE, go to the Policy > Policy Elements > Authorization > Authorization Profile page. Check the Interface Template check box and enter the name of the template to be assigned to the client interface.

If you are using a different RADIUS server, the attribute Cisco-AVpair="interface:template=name" must be pushed to the switch after the initial client authentication has been completed.



CHAPTER 7

IPv4 ACLs

- Restrictions for IPv4 access control lists, on page 133
- Information About IPv4 Access Control Lists, on page 135
- How to Configure IPv4 Access Control Lists, on page 148
- IPv4 ACL monitoring commands, on page 167
- Configuration Examples for IPv4 Access Control Lists, on page 168

Restrictions for IPv4 access control lists

General network security restrictions for ACLs

- Not all commands that accept a numbered ACL accept a named ACL. ACLs for packet filters and route filters on interfaces can use a name. VLAN maps also accept a name.
- A standard ACL and an extended ACL cannot have the same name.
- Though visible in the command-line help strings, **appletalk** is not supported as a matching condition for the **deny** and **permit** MAC access-list configuration mode commands.
- ACLs cannot be configured on management ports.
- ACL wildcard is not supported in downstream client policy.
- When you apply a scale ACL to an interface that does not program TCAM for a protocol and the ACLs that have been unloaded, it can impact the existing normal movement of traffic for other protocols. The restriction is applicable to .
- Router ACL is enforced on all types of traffic, including CPU generated traffic.
- ACL logging in the egress direction are not supported for packets that are generated from the control plane of the device.
- Time-to-live (TTL) classification is not supported on ACLs.
- If a downloadable ACL contains any type of duplicate entries, the entries are not auto merged. As a result, the 802.1X session authorization fails. Ensure that the downloadable ACL is optimized without any duplicate entries, for example port-based and name-based entries for the same port.
- Egress ACL lookup is not supported for injected traffic that is forwarded by the software.
- ACLs support only Layer 3 interfaces (such as routed interfaces and VLAN interfaces) and sub-interfaces.

These restrictions apply to IPv4 ACLs on Cisco Catalyst 9500X Series Switches, Cisco Catalyst 9600 Series Supervisor 2 Module, and other supported platforms.

Consider these restrictions to ensure proper ACL configuration and avoid unsupported scenarios or unexpected traffic behavior.

Following these restrictions helps maintain network security, prevents configuration errors, and ensures optimal ACL performance.

Review ACL configuration for compliance with platform-specific restrictions and optimize downloadable ACLs to avoid duplicate entries.

IPv4 ACL network interface restrictions

- When controlling access to an interface, you can use a named or numbered ACL.
- If you apply an ACL to a Layer 2 interface that is a member of a VLAN, the Layer 2 (port) ACL takes precedence over an input Layer 3 ACL applied to the VLAN interface or a VLAN map applied to the VLAN.
- If you apply an ACL to a Layer 3 interface and routing is not enabled on the switch, the ACL only filters packets that are intended for the CPU, such as SNMP, Telnet, or web traffic.
- If the **preauth_ipv4_acl** ACL is configured to filter packets, the ACL is removed after authentication.
- You do not have to enable routing to apply ACLs to Layer 2 interfaces.

These restrictions apply to IPv4 ACLs configured on Layer 2 and Layer 3 network interfaces.

Consider interface-specific behaviors to ensure ACLs are applied correctly and traffic is filtered as intended.

Proper application of ACLs to interfaces ensures accurate traffic filtering and prevents unintended access.

Verify interface type and ACL precedence when configuring access control lists.

MAC ACLs on a Layer 2 interface

After you create a MAC ACL, you can apply it to a Layer 2 interface to filter non-IP traffic coming in that interface. When you apply the MAC ACL, consider these guidelines:

- You can apply no more than one IP access list and one MAC access list to the same Layer 2 interface. The IP access list filters only IP packets, and the MAC access list filters non-IP packets.
- A Layer 2 interface can have only one MAC access list. If you apply a MAC access list to a Layer 2 interface that has a MAC ACL configured, the new ACL replaces the previously configured one.



Note The **MAC access-group** interface configuration command is only valid when applied to a physical Layer 2 interface. You cannot use the command on EtherChannel port channels.

These guidelines apply to MAC ACLs configured on Layer 2 interfaces.

Consider MAC ACL limitations to ensure correct filtering of non-IP traffic and avoid configuration conflicts.

Applying MAC ACLs appropriately ensures only intended traffic is filtered on Layer 2 interfaces.

Apply only one MAC ACL per Layer 2 interface and use the **MAC access-group** command only on physical Layer 2 interfaces.

IP access list entry sequence numbering restriction

- This feature does not support dynamic, reflexive, or firewall access lists.

Applies to IP access list entry sequence numbering feature.

Consider this restriction to avoid unsupported access list types when using sequence numbering.

Using supported access list types ensures proper sequence numbering and avoids configuration errors.

Do not use dynamic, reflexive, or firewall access lists with sequence numbering.

Information About IPv4 Access Control Lists

Access control lists

An access control list is a packet filtering mechanism that

- filters traffic as it passes through a device
- permits or denies packets crossing specified interfaces, and
- applies sequential permit and deny conditions to packets based on access list criteria.

Packet filtering and ACL operation

Packet filtering can help limit network traffic and restrict network use by certain users or devices.

The switch compares packet fields against applied ACLs to verify permissions for forwarding, based on access list criteria.

- Packets are tested one by one against access list conditions.
- The first match determines whether the switch accepts or rejects the packet.
- If no conditions match, the switch rejects the packet.
- If there are no restrictions, the switch forwards the packet; otherwise, it drops the packet.
- The switch can use ACLs on all packets it forwards.

Access control list usage examples

You can use access lists to control which hosts access different parts of a network or to decide which types of traffic are forwarded or blocked at device interfaces. For example, you can allow e-mail traffic to be forwarded but not Telnet traffic.

Packets without ACL restrictions

If you do not configure ACLs, all packets passing through the switch could be allowed onto all parts of the network.

Access control entries

An access control entry is a rule in an access control list that

- appears in an ordered list within the ACL
- specifies *permit* or *deny* actions, and
- defines a set of conditions that a packet must satisfy to match the entry.

ACL supported types

An ACL supported type is a classification of access control lists that

- includes IP ACLs that filter IPv4 traffic, such as TCP, UDP, IGMP, and ICMP
- includes Ethernet ACLs that filter non-IP traffic, and
- includes quality of service (QoS) classification ACLs.

Supported ACLs

A supported ACL is a traffic filtering mechanism that

- includes port ACLs for controlling access to Layer 2 interfaces
- provides router ACLs for managing traffic routed between VLANs, and
- offers VLAN ACLs or VLAN maps for filtering traffic within Layer 2 VLANs.

Supported ACL reference information

The switch supports three types of ACLs to filter the traffic:

- Port ACLs controls the access to traffic entering a Layer 2 interface. You can apply port ACLs to a Layer 2 interface in each direction to each access list type—IPv4, IPv6, and MAC.
- Router ACLs control the access to traffic routed between VLANs and are applied to Layer 3 interfaces in a specific direction (inbound or outbound).

ACL precedences

An ACL precedence is a filtering order mechanism that

- defines the sequence in which port ACLs, VLAN maps, and router ACLs filter ingress and egress traffic
- prioritizes port ACLs, VLAN map, and router ACL for ingress traffic, and
- prioritizes router ACL, VLAN map, and port ACL for egress traffic

Examples of ACL precedence filtering

- When both an input port ACL and a VLAN map are applied, incoming packets received on ports with a port ACL applied are filtered by the port ACL. Other packets are filtered by the VLAN map.
- When an input router ACL and input port ACL exist in a switch virtual interface (SVI), incoming packets received on ports to which a port ACL is applied are filtered by the port ACL. Incoming routed IP packets received on other ports are filtered by the router ACL. Other packets are not filtered.
- When an output router ACL and input port ACL exist in an SVI, incoming packets received on the ports to which a port ACL is applied are filtered by the port ACL. Outgoing routed IP packets are filtered by the router ACL. Other packets are not filtered.
- When a VLAN map, input router ACL, and input port ACL exist in an SVI, incoming packets received on the ports to which a port ACL is applied are only filtered by the port ACL. Incoming routed IP packets received on other ports are filtered by both the VLAN map and the router ACL. Other packets are filtered only by the VLAN map.
- When a VLAN map, output router ACL, and input port ACL exist in an SVI, incoming packets received on the ports to which a port ACL is applied are only filtered by the port ACL. Outgoing routed IP packets are filtered by both the VLAN map and the router ACL. Other packets are filtered only by the VLAN map.

Port ACLs

A port ACL is an access control list that

- is applied to Layer 2 interfaces on a switch
- supports application on physical interfaces and EtherChannel interfaces but not on EtherChannel member interfaces, and
- can be applied in inbound and outbound directions to filter traffic based on access list entries.

Supported access lists for port ACLs

The following access lists are supported for port ACLs:

- Standard IP access lists using source addresses
- Extended IP access lists using source and destination addresses and optional protocol type information
- MAC extended access lists using source and destination MAC addresses and optional protocol type information

The switch examines ACLs on an interface and permits or denies packet forwarding based on how the packet matches the entries in the ACL. In this way, ACLs control access to a network or to part of a network.

When you apply a port ACL to a trunk port, the ACL filters traffic on all VLANs present on the trunk port. When you apply a port ACL to a port with voice VLAN, the ACL filters traffic on both data and voice VLANs.

With port ACLs, you can filter IP traffic by using IP access lists and non-IP traffic by using MAC addresses. You can filter both IP and non-IP traffic on the same Layer 2 interface by applying both an IP access list and a MAC access list to the interface.

Note You can't apply more than one IP access list and one MAC access list to a Layer 2 interface. If an IP access list or MAC access list is already configured on a Layer 2 interface and you apply a new IP access list or MAC access list to the interface, the new ACL replaces the previously configured one.

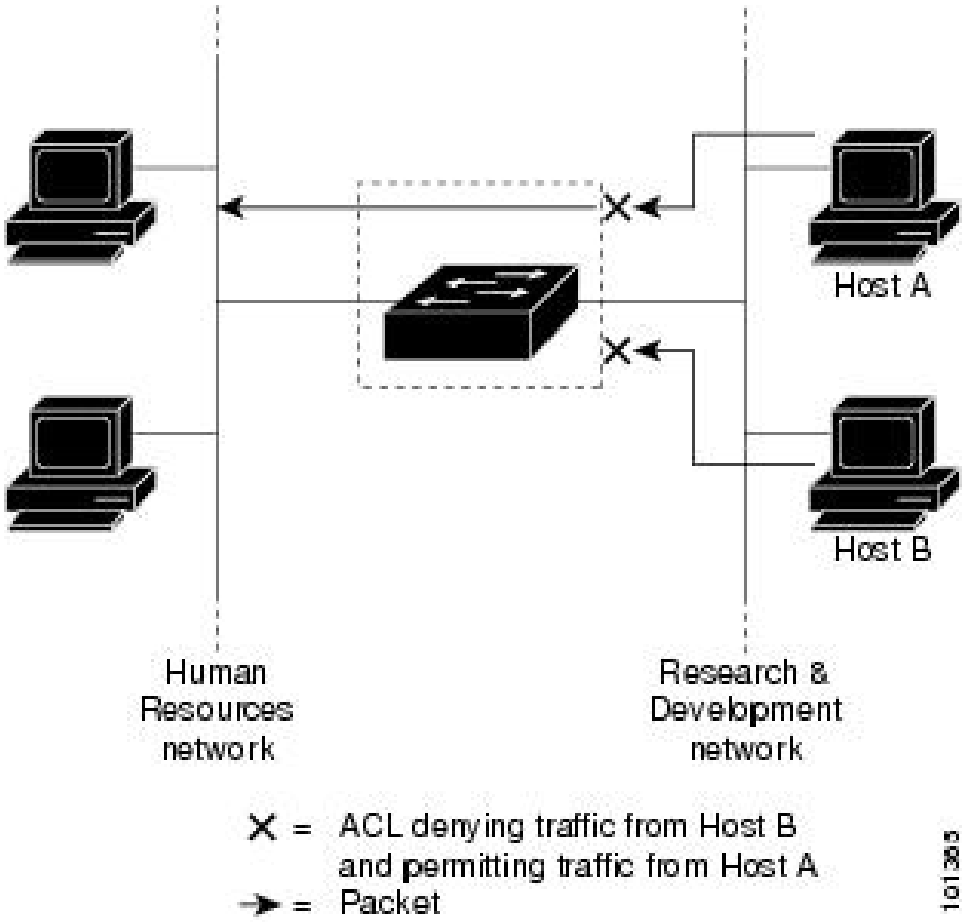
Port ACLs in network traffic control

This is an example of using port ACLs to control access to a network when all workstations are in the same VLAN. ACLs applied at the Layer 2 input would allow Host A to access the Human Resources network, but prevent Host B from accessing the same network. Port ACLs can only be applied to Layer 2 interfaces in the inbound direction.

Figure 7: Using ACLs to control traffic in a network

This is an example of using port ACLs to control access to a network when all workstations are in the same VLAN. ACLs applied at the Layer 2 input would allow Host A to access the Human Resources network, but prevent Host B from accessing the same network. Port ACLs can only be applied to Layer 2 interfaces in the inbound direction.

Figure 8: Using ACLs to control traffic in a network



Router ACLs

A router ACL is a network security mechanism that

- can be applied to switch virtual interfaces (SVIs), physical Layer 3 interfaces, and Layer 3 EtherChannel interfaces
- is applied on interfaces for specific directions (inbound or outbound), and
- permits or denies packet forwarding based on how the packet matches the entries in the ACL.

Router ACL reference information

You can apply one router ACL in each direction on an interface.

The switch supports these access lists for IPv4 traffic:

- Standard IP access lists use source addresses for matching operations.

- Extended IP access lists use source and destination addresses and optional protocol type information for matching operations.

As with port ACLs, the switch examines ACLs associated with features configured on a given interface. As packets enter the switch on an interface, ACLs associated with all inbound features configured on that interface are examined. After packets are routed and before they are forwarded to the next hop, all ACLs associated with outbound features configured on the egress interface are examined.

ACLs can be used to control access to a network or to part of a network.

VLAN maps

A VLAN map is a network traffic control mechanism that

- applies to all packets bridged within a VLAN in the switch
- enforces access control for non-IP protocols through MAC addresses and Ethertype, and
- permits or denies forwarding of packets based on the action specified in the map.

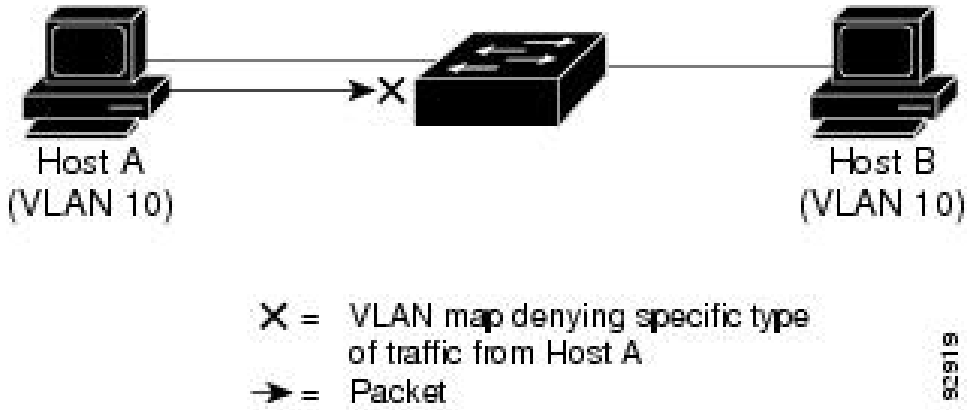
VLAN map reference information

VLAN ACLs or VLAN maps are used to control the network traffic within a VLAN. You can apply VLAN maps to all packets that are bridged within a VLAN in the switch. VACLs are strictly for the security packet filtering and for redirecting traffic to specific physical interfaces. VACLs are not defined by direction (ingress or egress).

All non-IP protocols are access-controlled through MAC addresses and Ethertype using MAC VLAN maps. IP traffic is not access-controlled by MAC VLAN maps. You can enforce VLAN maps only on packets going through the switch; you cannot enforce VLAN maps on traffic between hosts on a hub or on another switch that is connected to this switch.

With VLAN maps, forwarding of packets is permitted or denied, based on the action specified in the map.

Figure 9: Using VLAN maps to control traffic



ACEs and fragmented and unfragmented traffic

An ACE is an access control entry that

- can be applied to all packet fragments if it does not check Layer 4 information
- cannot be applied in the standard manner to most fragments in a fragmented IP packet if it tests Layer 4 information, and
- modifies matching rules when fragments lack Layer 4 information.

Fragmented and unfragmented traffic handling

IP packets can be fragmented as they cross the network. When this happens, only the fragment containing the beginning of the packet contains the Layer 4 information, such as TCP or UDP port numbers, ICMP type and code, and so on. All other fragments are missing this information.

Some ACEs do not check Layer 4 information and therefore can be applied to all packet fragments. ACEs that do test Layer 4 information cannot be applied in the standard manner to most of the fragments in a fragmented IP packet. When the fragment contains no Layer 4 information and the ACE tests some Layer 4 information, the matching rules are modified:

- Permit ACEs that check the Layer 3 information in the fragment (including protocol type, such as TCP, UDP, and so on) are considered to match the fragment regardless of what the missing Layer 4 information might have been.



Note For TCP ACEs with L4 Ops, the fragmented packets will be dropped per RFC 1858.

- Deny ACEs that check Layer 4 information never match a fragment unless the fragment contains Layer 4 information.

Standard and extended IPv4 ACLs

A standard and extended IPv4 ACL is a packet filtering mechanism that

- consists of a sequential collection of permit and deny conditions
- tests packets against access list conditions in order, and
- determines packet acceptance or rejection based on the first match.

Types of IPv4 ACLs

The software supports these types of ACLs for IPv4:

- Standard IP access lists use source addresses for matching operations.
- Extended IP access lists use source and destination addresses for matching operations and optional protocol-type information for finer granularity of control.

The order of conditions in an access list is critical because the device stops testing after the first match. If no conditions match, the device denies the packet.

Restrictions for IPv4 ACL

Restrictions for IPv4 ACL switch features

These ACL-related features are not supported:

- Non-IP protocol ACLs
- IP accounting
- Reflexive ACLs and dynamic ACLs are not supported.

Access list numbers

The number used to denote an access control list (ACL) indicates the type of access list being created. This topic lists the access list numbers, their corresponding types, and whether they are supported in the switch.

This lists the access-list number and corresponding access list type and shows whether or not they are supported in the switch. The switch supports IPv4 standard and extended access lists, numbers 1 to 199 and 1300 to 2699.

Table 15: Access list numbers

Access List Number	Type	Supported
1–99	IP standard access list	Yes
100–199	IP extended access list	Yes
200–299	Protocol type-code access list	No
300–399	DECnet access list	No
400–499	XNS standard access list	No
500–599	XNS extended access list	No
600–699	AppleTalk access list	No
700–799	48-bit MAC address access list	No
800–899	IPX standard access list	No
900–999	IPX extended access list	No
1000–1099	IPX SAP access list	No
1100–1199	Extended 48-bit MAC address access list	No
1200–1299	IPX summary address access list	No
1300–1999	IP standard access list (expanded range)	Yes
2000–2699	IP extended access list (expanded range)	Yes

In addition to numbered standard and extended ACLs, you can also create standard and extended named IP ACLs by using the supported numbers. That is, the name of a standard IP ACL can be 1 to 99; the name of an extended IP ACL can be 100 to 199. The advantage of using named ACLs instead of numbered lists is that you can delete individual entries from a named list.

Numbered standard IPv4 ACLs

A numbered standard IPv4 ACL is an access control list that

- contains an implicit deny statement for all unmatched packets by default
- assumes a mask of 0.0.0.0 if the mask is omitted from an associated IP host address ACL specification, and
- can be applied to VLANs, terminal lines, or interfaces.

Display order and application of numbered standard IPv4 ACLs

Numbered standard IPv4 ACLs display ACEs in ascending order according to their sequence numbers when using the **show IP access-list** *acl_name* or the **show run section** *acl_name* command.

Numbered extended IPv4 ACLs

A numbered extended IPv4 ACL is an access control list that

- uses source and destination addresses for matching operations
- supports optional protocol type information for finer granularity of control, and
- places any additions at the end of the list, without allowing reordering or selective addition/removal of ACEs.

Supported protocols and limitations for numbered extended IPv4 ACLs

Numbered extended IPv4 ACLs allow you to define extended TCP, UDP, ICMP, IGMP, or other IP ACLs. The device supports these IP protocols:

- Authentication Header Protocol (**ahp**)
- Encapsulation Security Payload (**esp**)
- Enhanced Interior Gateway Routing Protocol (**eigrp**)
- Generic routing encapsulation (**gre**)
- Internet Control Message Protocol (**ICMP**)
- Internet Group Management Protocol (**IGMP**)
- Any Interior Protocol (**IP**)
- IP in IP tunneling (**ipinip**)
- KA9Q NOS-compatible IP over IP tunneling (**NOS**)
- Open Shortest Path First routing (**ospf**)
- Payload Compression Protocol (**pcp**)

- Protocol-Independent Multicast (**pim**)
- Transmission Control Protocol (**TCP**)
- User Datagram Protocol (**UDP**)

Some protocols also have specific parameters and keywords that apply to that protocol.

Note ICMP echo-reply cannot be filtered. All other ICMP codes or types can be filtered.

The device does not support dynamic or reflexive access lists. It also does not support filtering based on the type of service (ToS) minimize-monetary-cost bit.

Named IPv4 ACLs

A named IPv4 ACL is an access control list that

- can be identified with an alphanumeric string (a name) rather than a number
- enables configuration of more IPv4 access lists in a device than numbered access lists, and
- has slightly different mode and command syntax compared to numbered access lists.

Guidelines for named IPv4 ACLs

The following guidelines apply when configuring named IPv4 ACLs:

- Numbered ACLs are also available.
- A standard ACL and an extended ACL cannot have the same name.
- You can use standard or extended ACLs (named or numbered) in VLAN maps.

Note The name you give to a standard or extended ACL can also be a number in the supported range of access list numbers. That is, the name of a standard IP ACL can be 1 to 99. The advantage of using named ACLs instead of numbered lists is that you can delete individual entries from a named list.

ACL logging

An ACL logging feature is a device software capability that

- generates logging messages for packets permitted or denied by a standard IP access list
- sends informational messages to the console when packets match the ACL, and
- controls the level of messages logged using **logging console** commands for syslog messages.

Note ACL logging is not supported for ACLs used with Unicast Reverse Path Forwarding (uRPF). ACL logging is supported only for router ACLs and not for port ACLs.

Note Because routing is done in hardware and logging is done in software, if a large number of packets match a *permit* or *deny* ACE containing a **log** keyword, the software might not be able to match the hardware processing rate, and not all packets will be logged.

The first packet that triggers the ACL causes a logging message immediately, and subsequent packets are collected over 5-minute intervals before they are logged. The logging message includes the access list number, whether the packet was permitted or denied, the source IP address, and the number of packets from that source permitted or denied in the prior 5-minute interval.

Note The logging facility might drop some logging message packets if there are too many to be handled or if there is more than one logging message to be handled in 1 second. This behavior prevents the device from crashing due to too many logging packets. Therefore, the logging facility should not be used as a billing tool or an accurate source of the number of matches to an access list.

ACL logging analogy

ACL logging is similar to a security camera that records events when triggered, but may miss some events if too many occur at once or if the recording system cannot keep up.

Hardware and software treatment of IP ACLs

An IP ACL treatment is a processing method that

- performs ACL processing in hardware
- drops all packets on an interface if hardware capacity to store ACL configurations is reached, and
- affects only traffic in a VLAN arriving on a device if an ACL configuration cannot be implemented in hardware due to an out-of-resource condition.

Router ACLs and hardware processing details

Router ACLs may send packets to the CPU due to specific factors.

- Using the **log** keyword
- Generating ICMP unreachable messages

Match counts in command output may not reflect hardware-controlled packets.

When you enter the **show IP access-lists** privileged EXEC command, the match count displayed does not account for packets that are access controlled in hardware. Use the **show platform software fed switch** {

switch_num | **active** | **standby** } **ACL counters hardware** privileged EXEC command to obtain some basic hardware ACL statistics for switched and routed packets.

Router ACLs function as follows:

- The hardware controls permit and deny actions of standard and extended ACLs (input and output) for security access control.
- If **log** has not been specified, the flows that match a *deny* statement in a security ACL are dropped by the hardware if *IP unreachables* is disabled. The flows matching a *permit* statement are switched in hardware.
- Adding the **log** keyword to an ACE in a router ACL causes a copy of the packet to be sent to the CPU for logging only. If the ACE is a *permit* statement, the packet is still switched and routed in hardware.



Note If an ACL configuration cannot be implemented in hardware due to an out-of-resource condition on a device, then only the traffic in that VLAN arriving on that device is affected.

Hardware and software ACL processing example

For example, if the hardware reaches its capacity to store ACL configurations, all packets on that interface are dropped. Using the **log** keyword in a router ACL causes a copy of the packet to be sent to the CPU for logging, while permit statements are still switched in hardware.

Guidelines for VLAN map configuration

VLAN map configuration guidelines

VLAN maps are the only way to control filtering within a VLAN. VLAN maps have no direction. To filter traffic in a specific direction by using a VLAN map, you need to include an ACL with specific source or destination addresses.

- If there is no ACL configured to deny traffic on an interface and no VLAN map is configured, all traffic is permitted.
- Each VLAN map consists of a series of entries. The order of entries in a VLAN map is important. A packet that comes into the device is tested against the first entry in the VLAN map. If it matches, the action specified for that part of the VLAN map is taken. If there is no match, the packet is tested against the next entry in the map.
- If the VLAN map has at least one match clause for the type of packet (IP or MAC) and the packet does not match any of these match clauses, the default is to drop the packet. If there is no match clause for that type of packet in the VLAN map, the default is to forward the packet.
- Logging is not supported for VLAN maps.
- When a device has an IP access list or MAC access list applied to a Layer 2 interface, and you apply a VLAN map to a VLAN that the port belongs to, the port ACL takes precedence over the VLAN map.
- If a VLAN map configuration cannot be applied in hardware, all packets in that VLAN are dropped.

VLAN Maps with Router ACLs

To access control both bridged and routed traffic, you can use VLAN maps only or a combination of router ACLs and VLAN maps. You can define router ACLs on both input and output routed VLAN interfaces, and you can define a VLAN map to access control the bridged traffic.

If a packet flow matches a VLAN-map deny clause in the ACL, regardless of the router ACL configuration, the packet flow is denied.

Note When you use router ACLs with VLAN maps, packets that require logging on the router ACLs are not logged if they are denied by a VLAN map.

If the VLAN map has a match clause for the type of packet (IP or MAC) and the packet does not match the type, the default is to drop the packet. If there is no match clause in the VLAN map, and no action specified, the packet is forwarded if it does not match any VLAN map entry.

Guidelines for VLAN maps and router ACL configuration

Restrictions for VLAN map and router ACL configuration

Time ranges for ACLs

A time range is an ACL configuration feature that

- enables selective application of extended ACLs based on the time of day and week
- allows you to define a time-range name and set times, dates, or days of the week, and
- lets you specify when permit or deny statements in the ACL are in effect.

Benefits of using time ranges

Time ranges provide additional control and flexibility for ACLs.

- You have more control over permitting or denying a user access to resources, such as an application (identified by an IP address/mask pair and a port number).
- You can control logging messages. ACL entries can be set to log traffic only at certain times of the day. Therefore, you can simply deny access without needing to analyze many logs generated during peak hours.

Time-based access lists trigger CPU activity because the new configuration of the access list must be merged with other features and the combined configuration loaded into the hardware memory. For this reason, you should be careful not to have several access lists configured to take effect in close succession (within a small number of minutes of each other).

Note The time range relies on the device system clock; therefore, you need a reliable clock source. We recommend that you use Network Time Protocol (NTP) to synchronize the device clock.

IPv4 ACL interface considerations

An IPv4 ACL interface consideration is a configuration aspect that

- requires a Layer 3 interface to have an IP address before applying the **IP access-group** command
- ensures Layer 3 access groups filter packets that are routed or received by Layer 3 processes on the CPU, and
- does not affect packets bridged within a VLAN.

IPv4 ACL interface processing details

IPv4 ACLs process packets differently depending on whether the ACL is inbound or outbound.

- For inbound ACLs, after receiving a packet, the device checks the packet against the ACL. If the ACL permits the packet, the device continues to process the packet. If the ACL rejects the packet, the device discards the packet.
- For outbound ACLs, after receiving and routing a packet to a controlled interface, the device checks the packet against the ACL. If the ACL permits the packet, the device sends the packet. If the ACL rejects the packet, the device discards the packet.
- By default, the input interface sends ICMP Unreachable messages whenever a packet is discarded, regardless of whether the packet was discarded because of an ACL on the input interface or because of an ACL on the output interface. ICMP Unreachables are normally limited to no more than one every one-half second per input interface, but this can be changed by using the **IP ICMP rate-limit unreachable** global configuration command.
- When you apply an undefined ACL to an interface, the device acts as if the ACL has not been applied to the interface and permits all packets. Remember this behavior if you use undefined ACLs for network security.

How to Configure IPv4 Access Control Lists

Configure IPv4 ACLs

Configure IPv4 ACLs to manage and restrict network traffic based on access conditions. This task helps you define and enforce security policies on your switch.

IPv4 access control lists (ACLs) are used to filter traffic and control access to network resources. Use this task when you need to implement security policies or traffic management on your switch.

These are the steps to use IP ACLs on the switch:

Before you begin

No explicit prerequisites are stated in the source.

Follow these steps to configure IPv4 ACLs:

Procedure

Step 1 Create an ACL by specifying an access list number or name and the access conditions.

Step 2 Apply the ACL to interfaces or terminal lines. You can also apply standard and extended IP ACLs to VLAN maps.

Create a numbered standard ACL

This task enables you to configure a numbered standard IPv4 access control list (ACL) on a Cisco Switch. Numbered standard ACLs allow you to permit or deny traffic based on source IP addresses, enhancing network security and traffic management.

Use this task when you need to control access to network resources by filtering IPv4 packets based on their source addresses. Numbered standard ACLs are typically applied to router interfaces to restrict or allow traffic as required by your network policy.

Before you begin

No specific prerequisites are required to create a numbered standard ACL, but you should have access to the Switch with appropriate privileges.

Perform these steps to create a numbered standard ACL:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **access-list** *access-list-number* {**deny** | **permit**} *source source-wildcard*]

Example:

```
Switch(config)# access-list 2 deny your_host
```

Defines a standard IPv4 access list by using a source address and wildcard.

The *access-list-number* is a decimal number from 1 to 99 or 1300 to 1999.

Enter **deny** or **permit** to specify whether to deny or permit access if conditions are matched.

The *source* is the source address of the network or host from which the packet is being sent specified as:

- The 32-bit quantity in dotted-decimal format.
- The keyword **any** as an abbreviation for *source* and *source-wildcard* of 0.0.0.0 255.255.255.255. You do not need to enter a source-wildcard.
- The keyword **host** as an abbreviation for source and *source-wildcard* of *source* 0.0.0.0.

(Optional) The *source-wildcard* applies wildcard bits to the source.

Note
Logging is supported only on ACLs attached to layer 3 interfaces.

Step 4 **end**

Example:

```
Switch(config)# end
```

Exits global configuration mode and returns to privileged EXEC mode.

Create a numbered extended ACL

This task enables you to configure a numbered extended access control list (ACL) to filter IPv4 traffic based on protocol, source, and destination parameters.

Use this task when you need to control network traffic by permitting or denying packets based on specific criteria such as protocol type, source and destination addresses, and optional parameters like precedence, type of service, or time range.

Before you begin

There are no explicit prerequisites for this task.

Follow these steps to create a numbered extended ACL:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **access-list** *access-list-number* { **deny** | **permit** } *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**tos** tos] [**fragments**] [**time-range** *time-range-name*] [**DSCP** *DSCP*]

Example:

```
Switch(config)# access-list 101 permit ip host 10.1.1.2 any precedence 0 tos 0 log
```

Defines an extended IPv4 access list and the access conditions.

The *access-list-number* is a decimal number from 100 to 199 or 2000 to 2699.

Enter **deny** or **permit** to specify whether to deny or permit the packet if conditions are matched.

Source, source-wildcard, destination, and destination-wildcard can be specified as:

- The 32-bit quantity in dotted-decimal format.
- The keyword **any** for 0.0.0.0 255.255.255.255 (any host).
- The keyword **host** for a single host 0.0.0.0.

The other keywords are optional and have these meanings:

- **precedence**: Enter to match packets with a precedence level specified as a number from 0 to 7 or by name: **routine** (0), **priority** (1), **immediate** (2), **flash** (3), **flash-override** (4), **critical** (5), **internet** (6), **network** (7).
- **fragments**: Enter to check non-initial fragments.
- **tos**: Enter to match by type of service level, specified by a number from 0 to 15 or a name: **normal** (0), **max-reliability** (2), **max-throughput** (4), **min-delay** (8).
- **time-range**: Specify the time-range name.
- **DSCP**: Enter to match packets with the DSCP value specified by a number from 0 to 63, or use the question mark (?) to see a list of available values.

Note
If you enter a **DSCP** value, you cannot enter **tos** or **precedence**. You can enter both a **tos** and a **precedence** value with no **DSCP**.

Step 4 **access-list** *access-list-number* {**deny** | **permit**} **TCP** *source source-wildcard* [*operator port*] *destination destination-wildcard* [*operator port*] [**established**] [**precedence** *precedence*] [**tos** *tos*] [**fragments**] [**time-range** *time-range-name*] [**DSCP** *DSCP*] [*flag*]

Example:

```
Switch(config)# access-list 101 permit tcp any any eq 500
```

Defines an extended TCP access list and the access conditions.

The parameters are the same as those described for an extended IPv4 ACL, with these exceptions:

(Optional) Enter an *operator* and *port* to compare source (if positioned after *source source-wildcard*) or destination (if positioned after *destination destination-wildcard*) port. Possible operators include **eq** (equal), **gt** (greater than), **lt** (less than), **neq** (not equal), and **range** (inclusive range). Operators require a port number (range requires two port numbers separated by a space).

Enter the *port* number as a decimal number (from 0 to 65535) or the name of a TCP port. Use only TCP port numbers or names when filtering TCP.

The other optional keywords have these meanings:

- **established**: Enter to match an established connection. This has the same function as matching on the **ack** or **rst** flag.
- *flag*: Enter one of these flags to match by the specified TCP header bits: **ack** (acknowledge), **fin** (finish), **psh** (push), **rst** (reset), **syn** (synchronize), or **urg** (urgent).

Step 5 **access-list** *access-list-number* { **deny** | **permit** } **UDP** *source source-wildcard* [*operator port*] *destination destination-wildcard* [*operator port*] [**precedence** *precedence*] [**tos** *tos*] [**fragments**] [**time-range** *time-range-name*] [**DSCP** *DSCP*]

Example:

```
Switch(config)# access-list 101 permit udp any any eq 100
```

(Optional) Defines an extended UDP access list and the access conditions.

The UDP parameters are the same as those described for TCP except that the [operator [port]] port number or name must be a UDP port number or name, and the **flag** and **established** keywords are not valid for UDP.

Step 6 **access-list** *access-list-number* { **deny** | **permit** }**ICMP** *source source-wildcard destination destination-wildcard* [*ICMP-type* | [[*ICMP-type ICMP-code*] | [*ICMP-message*]] [**precedence** *precedence*] [**tos** *tos*] [**fragments**] [**time-range** *time-range-name*] [**DSCP** *DSCP*] }

Example:

```
Switch(config)# access-list 101 permit icmp any any 200
```

Defines an extended ICMP access list and the access conditions.

The ICMP parameters are the same as those described for most IP protocols in an extended IPv4 ACL, with the addition of the ICMP message type and code parameters. These optional keywords have these meanings:

- *ICMP-type*: Enter to filter by ICMP message type, a number from 0 to 255.
- *ICMP-code*: Enter to filter ICMP packets that are filtered by the ICMP message code type, a number from 0 to 255.
- *ICMP-message*: Enter to filter ICMP packets by the ICMP message type name or the ICMP message type and code name.

Step 7 **access-list** *access-list-number* { **deny** | **permit** } **IGMP** *source source-wildcard destination destination-wildcard* [*IGMP-type*] [**precedence** *precedence*] [**tos** *tos*] [**fragments**] [**time-range** *time-range-name*] [**DSCP** *DSCP*]

Example:

```
Switch(config)# access-list 101 permit igmp any any 14
```

(Optional) Defines an extended IGMP access list and the access conditions.

The IGMP parameters are the same as those described for most IP protocols in an extended IPv4 ACL, with this optional parameter.

IGMP-type: To match IGMP message type, enter a number from 0 to 15, or enter the message name: **dvmrp**, **host-query**, **host-report**, **pim**, or **trace**.

Step 8 **end**

Example:

```
Switch(config)# end
```

Exits global configuration mode and returns to privileged EXEC mode.

Create a named standard ACL

This task enables you to create a named standard IPv4 access control list (ACL) to filter network traffic based on specified conditions.

Use this procedure when you need to define a standard ACL using a name instead of a number, allowing for more descriptive and manageable access lists. Named ACLs are useful for organizing and identifying access policies in complex network environments.

Before you begin

Perform these steps to create a standard ACL using names:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **ip access-list standard** *name*

Example:

```
Switch(config)# ip access-list standard 20
```

Defines a standard IPv4 access list using a name, and enters access-list configuration mode.

The name can be a number from 1 to 99.

Step 4 Use one of the following:

- **deny** {*source* [*source-wildcard*] | **host** *source* | **any**} [**log**]
- **permit** {*source* [*source-wildcard*] | **host** *source* | **any**} [**log**]

Example:

```
Switch(config-std-nacl)# deny 192.168.0.0 0.0.255.255 255.255.0.0 0.0.255.255
```

or

```
Switch(config-std-nacl)# permit 10.108.0.0 0.0.0.0 255.255.255.0 0.0.0.0
```

In access-list configuration mode, specify one or more conditions denied or permitted to decide if the packet is forwarded or dropped.

- **host** *source* : A source and source wildcard of *source* 0.0.0.0.
- **any** : A source and source wildcard of 0.0.0.0 255.255.255.255.

Step 5 **end**

Example:

```
Switch(config-std-nacl)# end
```

Exits access-list configuration mode and returns to privileged EXEC mode.

Create an extended named ACL

This task enables you to create an extended named access control list (ACL) to filter network traffic based on specific protocols, sources, and destinations. Using named ACLs allows for easier management and modification of access rules.

Use this procedure when you need to define or update extended ACLs using names instead of numbers. Named ACLs provide flexibility and allow for selective removal of entries, which is not possible with numbered ACLs.

Follow these steps to create an extended ACL using names:

Before you begin

No explicit prerequisites are stated. Ensure you have access to the Switch CLI and appropriate privileges if required.

Perform these steps to create an extended named ACL:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **IP access-list extended** *name*

Example:

```
Switch(config)# ip access-list extended 150
```

Defines an extended IPv4 access list using a name, and enters access-list configuration mode.

The name can be a number from 100 to 199.

Step 4 {**deny** | **permit**}*protocol* {*source* [*source-wildcard*] | **host** *source* | **any**} {*destination* [*destination-wildcard*] | host *destination* | **any**} [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log**] [**time-range** *time-range-name*]

Example:

```
Switch(config-ext-nacl)# permit 0 any any
```

In access-list configuration mode, specify the conditions allowed or denied. Use the **log** keyword to get access list logging messages, including violations.

- **host** *source*: A source and source wildcard of *source* 0.0.0.0.
- **host** *destination*: A destination and destination wildcard of *destination* 0.0.0.0.
- **any**: A source and source wildcard or destination and destination wildcard of 0.0.0.0 255.255.255.255.

Step 5 **end**

Example:

```
Switch(config-ext-nacl)# end
```

Exits access-list configuration mode and returns to privileged EXEC mode.

Configure time ranges for ACLs

Configure time ranges for ACLs to control when access control entries are active on your Switch.

Use this task when you need to apply access control lists (ACLs) only during specific times, such as business hours or maintenance windows. Time ranges allow you to automate ACL activation and deactivation based on a schedule.

Before you begin

There are no explicit prerequisites for this task.

Follow these steps to configure a time-range parameter for an ACL:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **time-range** *time-range-name*

Example:

```
Switch(config)# time-range workhours
```

Assigns a meaningful name (for example, *workhours*) to the time range to be created, and enters time-range configuration mode. The name cannot contain a space or quotation mark and must begin with a letter.

Step 4 Use one of the following:

- **absolute** [**start** *time date*] [**end** *time date*]
- **periodic** *day-of-the-week hh:mm to* [*day-of-the-week*] *hh:mm*
- **periodic** {**weekdays** | **weekend** | **daily**} *hh:mm to hh:mm*

Example:

```
Switch(config-time-range)# absolute start 00:00 1 Jan 2006 end 23:59 1 Jan 2006
```

or

```
Switch(config-time-range)# periodic weekdays 8:00 to 12:00
```

Specifies when the function it will be applied to is operational.

- You can use only one **absolute** statement in the time range. If you configure more than one absolute statement, only the one configured last is executed.
- You can enter multiple **periodic** statements. For example, you could configure different hours for weekdays and weekends.

Step 5 **end**

Example:

```
Switch(config-time-range)# end
```

Exits time-range configuration mode and returns to privileged EXEC mode.

Apply an IPv4 ACL to a terminal line

This task enables you to control access to terminal lines by applying a numbered IPv4 ACL, ensuring that only permitted addresses can connect to the Switch through the specified lines.

You can use numbered ACLs to control access to one or more terminal lines. You cannot apply named ACLs to lines. You must set identical restrictions on all the virtual terminal lines because a user can attempt to connect to any of them.

Before you begin

Ensure you have created the required numbered IPv4 access control list (ACL) before applying it to a terminal line.

Follow these steps to restrict incoming and outgoing connections between a virtual terminal line and the addresses in an ACL:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **line** [**console** | **vty**] *line-number*

Example:

```
Switch(config)# line console 0
```

Identifies a specific line to configure, and enter in-line configuration mode.

- **console**: Specifies the console terminal line. The console port is DCE.
- **vty**: Specifies a virtual terminal for remote console access.

The *line-number* is the first line number in a contiguous group that you want to configure when the line type is specified. The range is from 0 to 16.

Step 4 **access-class** *access-list-number* {**in** | **out**}

Example:

```
Switch(config-line)# access-class 10 in
```

Restricts incoming and outgoing connections between a particular virtual terminal line (into a Switch) and the addresses in an access list.

Step 5 **end**

Example:

```
Switch(config-line)# end
```

Exits line configuration mode and returns to privileged EXEC mode.

After completing these steps, the specified IPv4 ACL is applied to the terminal line, restricting access according to the ACL rules.

Apply an IPv4 ACL to an interface

Apply an IPv4 access control list (ACL) to a network interface to restrict or permit traffic based on defined rules by enforcing security policies by controlling access to network interfaces.

This task is relevant when you need to control access to a network interface using IPv4 ACLs.

Perform this configuration in privileged EXEC mode to ensure proper access control on Layer 2 or Layer 3 interfaces.

- Applicable to both port ACLs (Layer 2) and router ACLs (Layer 3).

Before you begin

Ensure you have an existing IPv4 ACL configured before applying it to an interface.

- Access to the Switch with privileged EXEC mode is required.

Follow these steps to apply an IPv4 ACL to an interface:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *interface-id*

Example:

```
Switch(config)# interface gigabitethernet1/0/1
```

Identifies a specific interface for configuration, and enters interface configuration mode.

The interface can be a Layer 2 interface (port ACL), or a Layer 3 interface (router ACL).

Step 4 **ip access-group** {*access-list-number* | *name*} {**in** | **out**}

Example:

```
Switch(config-if)# ip access-group 2 in
```

Controls access to the specified interface.

Step 5 **end**

Example:

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

Exits interface configuration mode and returns to privileged EXEC mode.

Configure a named MAC extended ACL

Configure a named MAC extended ACL to filter non-IPv4 traffic based on MAC addresses. This task allows you to control access to a VLAN or Layer 2 interface for non-IP protocols.

- Enable granular filtering of traffic using MAC address criteria.

You can filter non-IPv4 traffic on a VLAN or on a Layer 2 interface by using MAC addresses and named MAC extended ACLs.

The procedure is similar to that of configuring other extended named ACLs.

- Use this task when you need to restrict or permit traffic based on MAC addresses for non-IP protocols.

Before you begin

No explicit prerequisites are stated. Ensure you have access to the Switch and appropriate privileges to configure ACLs.

- Switch access and privilege required for configuration.

Follow these steps to configure a named MAC extended ACL:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **MAC access-list extended** *name*

Example:

```
Switch(config)# mac access-list extended mac1
```

Defines an extended MAC access list using a name.

Step 4 {**deny** | **permit**} {**any** | **host** *source MAC address* | *source MAC address mask*} {**any** | **host** *destination MAC address* | *destination MAC address mask*} [*type mask* | **LSAP** *LSAP mask* | **aarp** | **amber** | **dec-spanning** | **decnet-iv** | **diagnostic** | **dsm** | **etype-6000** | **etype-8042** | **lat** | **lavc-sca** | **mop-console** | **mop-dump** | **msdos** | **mumps** | **netbios** | **vines-echo** | **vines-IP** | **xns-idp** 0-*65535*] [**cos** *cos*]

Example:

```
Switch(config-ext-macl)# deny any any decnet-iv
```

or

```
Switch(config-ext-macl)# permit any any
```

In extended MAC access-list configuration mode, specifies to **permit** or **deny** any source MAC address, a source MAC address with a mask, or a specific **host** source MAC address and **any** destination MAC address, destination MAC address with a mask, or a specific destination MAC address.

(Optional) You can also enter these options:

- *type mask*—An arbitrary EtherType number of a packet with Ethernet II or SNAP encapsulation in decimal, hexadecimal, or octal with optional mask of *don't care* bits applied to the EtherType before testing for a match.
- **LSAP** *LSAP mask*—An LSAP number of a packet with IEEE 802.2 encapsulation in decimal, hexadecimal, or octal with optional mask of *don't care* bits.
- **aarp** | **amber** | **dec-spanning** | **decnet-iv** | **diagnostic** | **dsm** | **etype-6000** | **etype-8042** | **lat** | **lavc-sca** | **mop-console** | **mop-dump** | **msdos** | **mumps** | **netbios** | **vines-echo** | **vines-IP** | **xns-idp**—A non-IP protocol.
- **cos** *cos*—An IEEE 802.1Q cost of service number from 0 to 7 used to set priority.

Step 5 **end**

Example:

```
Switch(config-ext-macl)# end
```

Exits extended MAC access-list configuration mode and returns to privileged EXEC mode.

Apply a MAC ACL to a layer 2 interface

Applying a MAC ACL to a Layer 2 interface allows you to control access at the data link layer, enhancing network security by filtering traffic based on MAC addresses.

- Restricts traffic to and from a Layer 2 interface using MAC address filtering.
- Helps enforce security policies at the port level.

Use this task when you need to control access to a Layer 2 interface by applying a MAC access list. This is typically required in environments where port-level security is necessary.

Port ACLs are supported in both inbound and outbound directions. The interface must be a physical Layer 2 interface (port ACL).

- Applicable for switches supporting MAC ACLs on Layer 2 interfaces.

Before you begin

No explicit prerequisites are stated. Ensure you have a defined MAC ACL and access to the Switch's CLI.

- MAC ACL must be created prior to applying it to the interface.

Follow these steps to apply a MAC access list to control access to a Layer 2 interface:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *interface-id*

Example:

```
Switch(config)# interface gigabitethernet1/0/1
```

Identifies a specific interface, and enters interface configuration mode. The interface must be a physical Layer 2 interface (port ACL).

Step 4 **MAC access-group** {*name*} {**in** | **out** }

Example:

```
Switch(config-if)# mac access-group mac1 in
```

Controls access to the specified interface by using the MAC access list.

Port ACLs are supported in the outbound and inbound directions.

Step 5 **end**

Example:

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

Exits interface configuration mode and returns to privileged EXEC mode.

Step 6 **show MAC access-group** [**interface** *interface-id*]

Example:

```
Switch# show mac access-group interface gigabitethernet1/1
```

Displays the MAC access list applied to the interface or all Layer 2 interfaces.

After receiving a packet, the Switch checks it against the inbound ACL. If the ACL permits it, the Switch continues to process the packet. If the ACL rejects the packet, the Switch discards it. When you apply an undefined ACL to an interface, the Switch acts as if the ACL has not been applied and permits all packets. Remember this behavior if you use undefined ACLs for network security.

Configure an IPv4 ACL in template mode

This task enables you to configure an IPv4 access control list (ACL) in template mode, allowing you to control access to interfaces using named or numbered access lists.

Note You can configure **IP access-group** command in the template configuration mode. You can configure the **source template** command only once to an interface.

Use this task when you need to apply an IPv4 ACL to an interface using a template for consistent and reusable configuration.

Before you begin

Before you begin, ensure you are in privileged EXEC mode.

Perform these steps to configure an IPv4 ACL in template mode:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **IP access-list extended** *name*

Example:

```
Switch(config)# ip access-list extended 150
```

Defines an extended IPv4 access list using a name, and enters access-list configuration mode.

Enter *name* to define access-list name.

Enter *number* to define extended IP access-list number. The range is from 100 to 199.

Enter *ext_number* to define extended IP access-list number. The expanded range is from 2000 to 2699.

Step 4 **IP access-list extended** {*name*|*number*|*ext_number*}

Example:

```
Switch(config)# ip access-list extended 151
```

Defines an extended IPv4 access list using a name, and enters access-list configuration mode.

Enter *name* to define access-list name.

Enter *number* to define extended IP access-list number. The range is from 100 to 199.

Enter *ext_number* to define extended IP access-list number. The expanded range is from 2000 to 2699.

Step 5 **exit**

Example:

```
Switch(config-ext-nacl)# exit
```

Exits access-list configuration mode.

Step 6 **template**

Example:

```
Switch# template test
```

Creates a user template and enters template configuration mode.

Step 7 **IP access-group** {*access-list-number* | *name*} {**in** | **out**}

Example:

```
Switch(config-template)# ip access-group 150 in
```

Controls access to the specified interface.

Enter *access-list-number* to define the access list. The access list can be a number.

Enter *name* to define the access list. The access list can be a name.

Enter **in** to direct the access list in the incoming direction of the interface.

Enter **out** to direct the access list in the outgoing direction of the interface.

Step 8 **exit**

Example:

```
Switch(config-template)# exit
```

Exits template configuration mode and returns to privileged EXEC mode.

Step 9 **interface** *interface-id*

Example:

```
Switch(config)# interface gigabitethernet1/0/1
```

Identifies a specific interface for configuration, and enters interface configuration mode.

The interface can be a Layer 2 interface (port ACL), or a Layer 3 interface (router ACL).

Step 10 **IP access-group** {*access-list-number* | *name*} {**in** | **out**}

Example:

```
Switch(config-if)# ip access-group 151 out
```

Controls access to the specified interface.

Enter *access-list-number* to define the access list. The access list can be a number.

Enter *name* to define the access list. The access list can be a name.

Enter **in** to direct the access list in the incoming direction of the interface.

Enter **out** to direct the access list in the outgoing direction of the interface.

Step 11 **source template** *name*

Example:

```
Switch(config)# source template test
```

Applies an interface template to a target. The access list 150 is the incoming access list that is configured.

Step 12 **end**

Example:

```
Switch(config)# end
```

Exits global configuration mode and returns to privileged EXEC mode.

Configure a VLAN map

Configure a VLAN map to filter traffic on one or more VLANs by applying standard or extended IPv4 ACLs or named MAC extended ACLs.

Use this task when you need to control traffic within VLANs by matching packets against access lists and specifying actions such as forward or drop. VLAN maps are useful for implementing security and traffic policies at the VLAN level.

Before you begin

Create the standard or extended IPv4 ACLs or named MAC extended ACLs that you want to apply to the VLAN.

Follow these steps to configure a VLAN map and apply it to one or more VLANs:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **VLAN access-map** *name* [**number**]

Example:

```
Switch(config)# vlan access-map map1 20
```

Creates a VLAN map, and give it a name and (optionally) a number. The number is the sequence number of the entry within the map.

When you create VLAN maps with the same name, numbers are assigned sequentially in increments of 10. When modifying or deleting maps, you can enter the number of the map entry that you want to modify or delete.

VLAN maps do not use the specific permit or deny keywords. To deny a packet by using VLAN maps, create an ACL that would match the packet, and set the action to drop. A permit in the ACL counts as a match. A deny in the ACL means no match.

Entering this command changes to access-map configuration mode.

Step 4 **match** {**IP** | **MAC**} **address** {*name* | *number*} [*name* | *number*]

Example:

```
Switch(config-access-map)# match ip address ip2
```

Match the packet (using either the IP or MAC address) against one or more standard or extended access lists. Note that packets are only matched against access lists of the correct protocol type. IP packets are matched against standard or extended IP access lists. Non-IP packets are only matched against named MAC extended access lists.

Note
If the VLAN map is configured with a match clause for a type of packet (IP or MAC) and the map action is drop, all packets that match the type are dropped. If the VLAN map has no match clause, and the configured action is drop, all IP and Layer 2 packets are dropped.

Step 5 Enter one of the following commands to specify an IP packet or a non-IP packet (with only a known MAC address) and to match the packet against one or more ACLs (standard or extended):

- **action** { **forward**}

  ```
  Switch(config-access-map)# action forward
  ```

- **action** { **drop**}

  ```
  Switch(config-access-map)# action drop
  ```

Sets the action for the map entry.

Step 6 **exit**

Example:

```
Switch(config-access-map)# exit
```

Exits access-map configuration mode. and returns to global configuration mode.

Step 7 **VLAN filter** *mapname* **VLAN-list** *list*

Example:

```
Switch(config)# vlan filter map1 vlan-list 20-22
```

Applies the VLAN map to one or more VLAN IDs.

The list can be a single VLAN ID (22), a consecutive list (10-22), or a string of VLAN IDs (12, 22, 30). Spaces around the comma and hyphen are optional.

Step 8 **end**

Example:

```
Switch(config)# end
```

Exits global configuration mode and returns to privileged EXEC mode.

Apply a VLAN map to a VLAN

Apply a VLAN map to one or more VLANs to filter traffic based on defined criteria.

- This task allows you to control traffic flow by associating a VLAN map with selected VLANs.

Use this task when you need to filter traffic on VLANs using a VLAN map.

To apply a VLAN map to one or more VLANs, perform these steps.

- Applicable when configuring VLAN filtering on Cisco Switchs.

Before you begin

No explicit prerequisites are stated in the source.

Perform these steps to apply a VLAN map to a VLAN:

Procedure

Step 1 **enable**

Example:

```
Switch> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Switch# configure terminal
```

Enters global configuration mode.

Step 3 **VLAN filter** *mapname* **VLAN-list** *list*

Example:

```
Switch(config)# vlan filter map 1 vlan-list 20-22
```

Applies the VLAN map to one or more VLAN IDs.

The list can be a single VLAN ID (22), a consecutive list (10-22), or a string of VLAN IDs (12, 22, 30). Spaces around the comma and hyphen are optional.

Step 4 **end**

Example:

```
Switch(config)# end
```

Exits global configuration mode and returns to privileged EXEC mode.

After completing these steps, the VLAN map is applied to the specified VLANs. Traffic is filtered according to the rules defined in the VLAN map.

IPv4 ACL monitoring commands

You can monitor IPv4 ACLs by displaying the ACLs that are configured on the device, and displaying the ACLs that have been applied to interfaces and VLANs.

When you use the **IP access-group** interface configuration command to apply ACLs to a Layer 2 or 3 interface, you can display the access groups on the interface. You can also display the MAC ACLs applied to a Layer 2 interface. You can use the privileged EXEC commands as described in this table to display this information.

Table 16: Commands for displaying access lists and access groups

Command	Purpose
show access-lists [*number* | *name*]	Displays the contents of one or all current IP and MAC address access lists or a specific access list (numbered or named).
show IP access-lists [*number* | *name*]	Displays the contents of all current IP access lists or a specific IP access list (numbered or named).
show IP interface *interface-id*	Displays detailed configuration and status of an interface. If IP is enabled on the interface and ACLs have been applied by using the **IP access-group** interface configuration command, the access groups are included in the display.

Command	Purpose
show running-config [**interface** *interface-id*]	Displays the contents of the configuration file for the device or the specified interface, including all configured MAC and IP access lists and which access groups are applied to an interface.
show MAC access-group [**interface** *interface-id*]	Displays MAC access lists applied to all Layer 2 interfaces or the specified Layer 2 interface.

Configuration Examples for IPv4 Access Control Lists

ACLs in small networked offices

An ACL in a small networked office is a traffic control mechanism that

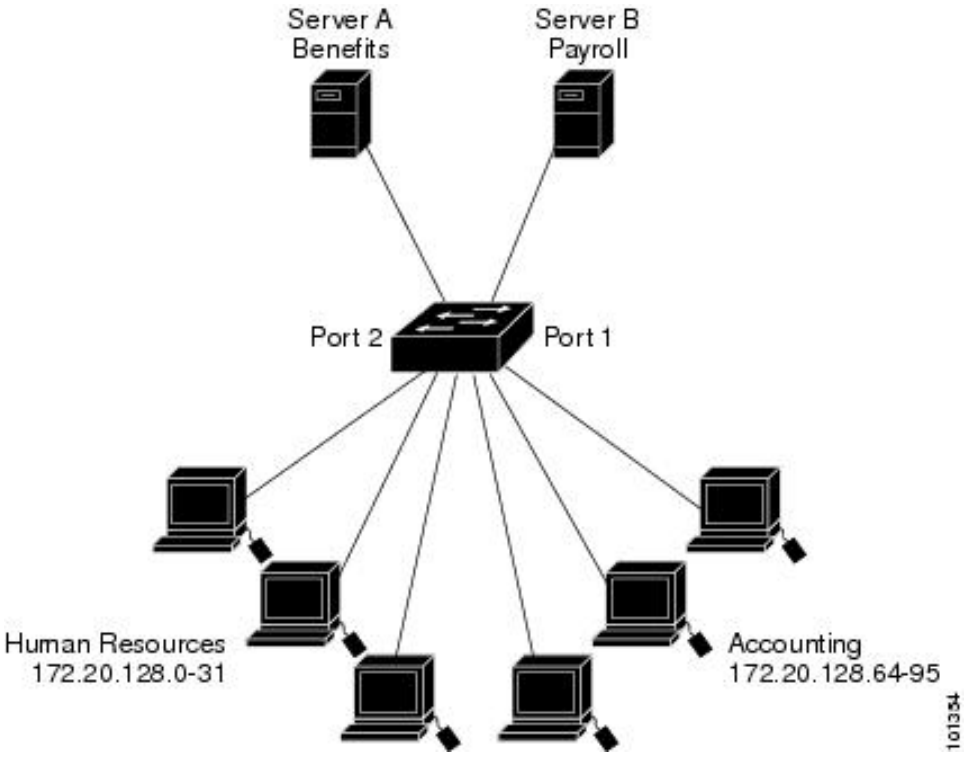
- enables filtering of traffic between devices
- allows creation of standard and extended ACLs, and
- controls access to servers and ports based on defined rules.

Router ACL traffic control methods

Router ACLs can be used in two ways to control traffic in a small networked office.

- Create a standard ACL to filter traffic coming to the server from Port 1.
- Create an extended ACL to filter traffic coming from the server into Port 1.

Figure 10: Using router ACLs to control traffic



Configure ACLs in a small networked office

This task demonstrates how to configure standard and extended ACLs to control traffic flow in a small networked office. The examples show how to permit or restrict traffic based on source and destination addresses.

Use this task when you need to filter network traffic to or from specific switches or subnets in a small office network. The examples provided are applicable to scenarios where you want to control access to resources such as servers or departmental networks.

Before you begin

Ensure you have access to the Switch CLI and are in privileged EXEC mode.

Perform these steps to configure ACLs in a small networked office:

Procedure

Step 1 Configure a standard ACL to permit traffic only from Accounting's source addresses (172.20.128.64 to 172.20.128.95) to Server B, and apply it to routed Port 1.

Example:

```
Switch> enable
                        Switch# configure terminal
                        Switch(config)# access-list 6 permit 172.20.128.64 0.0.0.31
                        Switch(config)# exit
                        Switch# show access-lists
```

```
Standard IP access list 6
10 permit 172.20.128.64, wildcard bits 0.0.0.31

Switch# configure terminal
Switch(config)# interface gigabitethernet1/0/1
Switch(config-if)# ip access-group 6 out
Switch(config-if)# end
```

The standard ACL filters traffic coming into Server B from a port, permitting only traffic from the specified Accounting source addresses. The ACL is applied outbound on routed Port 1.

Step 2 Configure an extended ACL to permit traffic from any source (such as Server B) only to the Accounting destination addresses (172.20.128.64 to 172.20.128.95), and apply it inbound on routed Port 1.

Example:

```
Switch(config)# access-list 106 permit ip any 172.20.128.64 0.0.0.31
Switch(config)# exit
Switch# show access-lists

Extended IP access list 106
10 permit ip any 172.20.128.64 0.0.0.31


Switch# configure terminal
Switch(config)# interface gigabitethernet1/0/1
Switch(config-if)# ip access-group 106 in
Switch(config-if)# end
```

The extended ACL filters traffic coming from Server B into a port, permitting it only to the specified Accounting destination addresses. For extended ACLs, specify the protocol (IP) before the source and destination information.

Configure numbered ACLs

This task enables you to configure numbered access control lists (ACLs) to selectively permit or deny traffic for specific hosts and subnets in a Class A network.

Numbered ACLs are used to filter network traffic based on IP addresses. In this example, network 10.0.0.0 is a Class A network whose second octet specifies a subnet, with a subnet mask of 255.255.0.0. The ACL is applied to packets entering a port.

Before you begin

Ensure you have access to the Switch CLI and appropriate privileges to configure ACLs.

Perform these steps to configure numbered ACLs:

Procedure

Configure access list 2 to permit and deny traffic for specific hosts and subnets, and apply the ACL to an interface.

Example:

```
Switch> enable
Switch# configure terminal
Switch(config)# access-list 2 permit 10.48.0.3
Switch(config)# access-list 2 deny 10.48.0.0 0.0.255.255

Switch(config)# access-list 2 permit 10.0.0.0
0.255.255.255

Switch(config)# interface gigabitethernet1/0/1
Switch(config-if)# ip access-group 2 in
Switch(config-if)# end
```

Using access list 2, the switch accepts one address on subnet 48 and rejects all others on that subnet. The last line of the list shows that the switch accepts addresses on all other network 10.0.0.0 subnets.

After completing these steps, the ACL will be applied to packets entering the specified port, permitting and denying traffic as defined in access list 2.

Configure extended access control lists

Configure extended access control lists (ACLs) to control network traffic by specifying rules for protocols, ports, and IP addresses. This task enables you to permit or restrict traffic based on detailed criteria.

Extended ACLs are used in scenarios where you need to allow or deny traffic based on protocol type, port numbers, and source or destination IP addresses. For example, you may want to permit incoming mail traffic to a dedicated mail host while restricting other types of connections.

Procedure

Step 1 Configure an extended ACL to permit incoming TCP connections with destination ports greater than 1023, permit incoming TCP connections to the SMTP port of host 172.16.0.0, and permit incoming ICMP messages for error feedback.

Example:

```
Switch> enable
      Switch# configure terminal
      Switch(config)# access-list 102 permit tcp any 172.16.0.0 0.0.255.255 gt 1023
      Switch(config)# access-list 102 permit tcp any host 172.16.1.2 eq 25
      Switch(config)# access-list 102 permit icmp any any
      Switch(config)# interface gigabitethernet1/0/1
      Switch(config-if)# ip access-group 102 in
      Switch(config-if)# end
```

The first line permits any incoming TCP connections with destination ports greater than 1023. The second line permits incoming TCP connections to the SMTP port of host 172.16.0.0. The third line permits incoming ICMP messages for error feedback.

Step 2 Configure an extended ACL to allow hosts on your network to form TCP connections to any host on the Internet, while restricting incoming TCP connections to only the mail (SMTP) port of a dedicated mail host.

Example:

```
Switch> enable
      Switch# configure terminal
```

```
Switch(config)# access-list 102 permit tcp any 172.16.0.0 0.0.255.255 eq 23
Switch(config)# access-list 102 permit tcp any 172.16.0.0 0.0.255.255 eq 25
Switch(config)# interface gigabitethernet1/0/1
Switch(config-if)# ip access-group 102 in
Switch(config-if)# end
```

Any host on the network can form TCP connections to any host on the Internet. Incoming mail packets from the Internet have a destination port of 25, and only the mail host accepts mail connections on port 25.

SMTP uses TCP port 25 on one end of the connection and a random port number on the other end. The same port numbers are used throughout the life of the connection. Mail packets coming in from the Internet have a destination port of 25. Because the secure system of the network always accepts mail connections on port 25, the incoming services are separately controlled.

Step 3 Configure an extended ACL for a Class B network with the address 172.16.0.0 and mail host address 172.16.1.2, using the **established** keyword to permit established TCP connections.

Example:

```
Switch> enable
      Switch# configure terminal
      Switch(config)# access-list 102 permit tcp any 172.16.0.0 0.0.255.255 established
      Switch(config)# access-list 102 permit tcp any host 172.16.1.2 eq 25
      Switch(config)# interface gigabitethernet1/0/1
      Switch(config-if)# ip access-group 102 in
      Switch(config-if)# end
```

The network is a Class B network with the address 172.16.0.0, and the mail host address is 172.16.1.2. The **established** keyword is used only for TCP to show an established connection. A match occurs if the TCP datagram has the ACK or RST bits set, which show that the packet belongs to an existing connection. Gigabit Ethernet interface 1 is the interface that connects the Switch to the Internet.

After completing these steps, the Switch will permit or restrict network traffic according to the specified extended ACL rules, providing granular control over protocols, ports, and IP addresses.

Configure named ACLs

Configure named ACLs to filter network traffic and control access based on defined rules. This task demonstrates how to create, apply, and manage standard and extended ACLs.

- Enable granular control over network traffic using named ACLs.
- Apply ACLs to interfaces for inbound and outbound filtering.
- Delete individual ACEs from named ACLs as needed.

Use this task when you need to implement access control policies on Cisco switches by creating, applying, or modifying named ACLs. Named ACLs allow for easier management and identification compared to numbered ACLs.

These examples are relevant for scenarios where you must permit or deny specific types of traffic, such as filtering by IP address, protocol, or port, and when you need to update ACLs by removing individual entries.

- Applicable to Layer 3 interfaces for both inbound and outbound traffic filtering.
- Useful for network administrators managing security and access policies.

Before you begin

Ensure you have access to the switch CLI and appropriate privileges to configure ACLs. Identify the traffic patterns and access requirements for your network before creating or modifying ACLs.

- Switch must be running Cisco IOS and support named ACLs.
- Know the IP addresses, protocols, and ports to be filtered.

Follow these steps to configure named ACLs:

Procedure

Step 1 Create a standard ACL named *internet_filter* to allow traffic from a specific source address.

Example:

```
Switch> enable
      Switch# configure terminal
      Switch(config)# ip access-list standard Internet_filter
      Switch(config-ext-nacl)# permit 10.2.3.4
      Switch(config-ext-nacl)# exit
      Switch(config-ext-nacl)# end
```

The *internet_filter* ACL allows all traffic from the source address 10.2.3.4.

The standard ACL *internet_filter* is created and ready to be applied to an interface.

Step 2 Create an extended ACL named *marketing_group* to permit and deny specific traffic types.

Example:

```
Switch> enable
      Switch# configure terminal
      Switch(config)# ip access-list extended marketing_group
      Switch(config-ext-nacl)# permit tcp any 172.16.0.0 0.0.255.255 eq telnet
      Switch(config-ext-nacl)# deny tcp any any
      Switch(config-ext-nacl)# permit icmp any any
      Switch(config-ext-nacl)# deny udp any 172.16.0.0 0.0.255.255 lt 1024
      Switch(config-ext-nacl)# deny ip any any log
      Switch(config-ext-nacl)# end
```

The *marketing_group* ACL allows any TCP Telnet traffic to 172.16.0.0/16, denies other TCP traffic, permits ICMP, denies UDP traffic to 172.16.0.0/16 with destination port less than 1024, denies other IP traffic, and logs the result.

The extended ACL *marketing_group* is created with multiple permit and deny rules.

Step 3 Apply the *internet_filter* and *marketing_group* ACLs to a Layer 3 interface.

Example:

```
Switch> enable
      Switch# configure terminal
      Switch(config)# interface gigabitethernet1/0/2
      Switch(config-if)# no switchport
      Switch(config-if)# ip address 10.0.5.1 255.255.255.0
      Switch(config-if)# ip access-group Internet_filter out
      Switch(config-if)# ip access-group marketing_group in
```

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

The *internet_filter* ACL is applied to outgoing traffic and the *marketing_group* ACL is applied to incoming traffic on the Layer 3 port.

ACLs are active on the interface, filtering traffic as specified.

Step 4 Delete an individual ACE from a named ACL.

Example:

```
Switch> enable
      Switch# configure terminal
      Switch(config)# ip access-list extended border-list
      Switch(config-ext-nacl)# no permit ip host 10.1.1.3 any
      Switch(config-ext-nacl)# end
```

This example shows how to delete a specific ACE from the named access list *border-list*.

The specified ACE is removed from the ACL.

After completing these steps, named ACLs are configured, applied to interfaces, and individual ACEs can be deleted as needed. Traffic is filtered according to the defined rules, enhancing network security and access control.

ACL logging examples

This reference presents examples of ACL logging on routers, showing how the **log** and **log-input** keywords generate informational messages for matched packets and how log entries differ based on ACL type and keyword usage.

Two variations of logging are supported on router ACLs. The **log** keyword sends an informational logging message to the console about the packet that matches the entry; the **log-input** keyword includes the input interface in the log entry.

In this example, standard named access list *stan1* denies traffic from 10.1.1.0 0.0.0.255, allows traffic from all other sources, and includes the **log** keyword.

```
Switch> enable
                Switch# configure terminal
                Switch(config)# ip access-list standard stan1
                Switch(config-std-nacl)# deny 10.1.1.0 0.0.0.255 log
                Switch(config-std-nacl)# permit any log
                Switch(config-std-nacl)# exit
                Switch(config)# interface gigabitethernet1/0/1

                Switch(config-if)# ip access-group stan1 in
                Switch(config-if)# end
                Switch# show logging

                Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
                Console logging: level debugging, 37 messages logged
                Monitor logging: level debugging, 0 messages logged
                Buffer logging: level debugging, 37 messages logged
                File logging: disabled
                Trap logging: level debugging, 39 message lines logged

                Log Buffer (4096 bytes):
```

```
00:00:48: NTP: authentication delay calculation problems

<output truncated>

00:09:34:%SEC-6-IPACCESSLOGS:list stan1 permitted 0.0.0.0 1 packet
00:09:59:%SEC-6-IPACCESSLOGS:list stan1 denied 10.1.1.15 1 packet
00:10:11:%SEC-6-IPACCESSLOGS:list stan1 permitted 0.0.0.0 1 packet
```

This example is a named extended access list *ext1* that permits ICMP packets from any source to 10.1.1.0 0.0.0.255 and denies all UDP packets.

```
Switch> enable
                Switch# configure terminal
                Switch(config)# ip access-list extended ext1
                Switch(config-ext-nacl)# permit icmp any 10.1.1.0 0.0.0.255 log
                Switch(config-ext-nacl)# deny udp any any log
                Switch(config-std-nacl)# exit
                Switch(config)# interface gigabitethernet1/0/1

                Switch(config-if)# ip access-group ext1 in
                Switch(config)# end
```

This is an example of a log for an extended ACL:

```
01:24:23:%SEC-6-IPACCESSLOGDP:list ext1 permitted icmp 10.1.1.15 -> 10.1.1.61 (0/0), 1
packet
             01:25:14:%SEC-6-IPACCESSLOGDP:list ext1 permitted icmp 10.1.1.15 -> 10.1.1.61
 (0/0), 7 packets
                01:26:12:%SEC-6-IPACCESSLOGP:list ext1 denied udp 0.0.0.0(0) ->
255.255.255.255(0), 1 packet
                01:31:33:%SEC-6-IPACCESSLOGP:list ext1 denied udp 0.0.0.0(0) ->
255.255.255.255(0), 8 packets
```

Note that all logging entries for IP ACLs start with %SEC-6-IPACCESSLOG with minor variations in format depending on the kind of ACL and the access entry that has been matched.

This is an example of an output message when the **log-input** keyword is entered:

```
00:04:21:%SEC-6-IPACCESSLOGDP:list inputlog permitted icmp 10.1.1.10 (Vlan1 0001.42ef.a400)
 ->
                10.1.1.61 (0/0), 1 packet
```

A log message for the same sort of packet using the **log** keyword does not include the input interface information:

```
00:05:47:%SEC-6-IPACCESSLOGDP:list inputlog permitted icmp 10.1.1.10 -> 10.1.1.61 (0/0), 1
 packet
```

Configuration examples for ACEs and fragmented and unfragmented traffic

This reference demonstrates how access control entries (ACEs) interact with fragmented and unfragmented traffic, using configuration examples and packet analysis.

Consider access list 102, configured with these commands, applied to three fragmented packets:

```
Switch> enable
    Switch# configure terminal
    Switch(config)# access-list 102 permit tcp any host 10.1.1.1 eq smtp
```

```
Switch(config)# access-list 102 deny tcp any host 10.1.1.2 eq telnet
Switch(config)# access-list 102 permit tcp any host 10.1.1.2
Switch(config)# access-list 102 deny tcp any any
Switch(config)# end
```

Note In the first and second ACEs in the examples, the *eq* keyword after the destination address means to test for the TCP-destination-port well-known numbers equaling Simple Mail Transfer Protocol (SMTP) and Telnet, respectively.

- Packet A is a TCP packet from host 10.2.2.2., port 65000, going to host 10.1.1.1 on the SMTP port. If this packet is fragmented, the first fragment matches the first ACE (a permit) as if it were a complete packet because all Layer 4 information is present. The remaining fragments also match the first ACE, even though they do not contain the SMTP port information, because the first ACE only checks Layer 3 information when applied to fragments. The information in this example is that the packet is TCP and that the destination is 10.1.1.1.
- Packet B is from host 10.2.2.2, port 65001, going to host 10.1.1.2 on the Telnet port. If this packet is fragmented, the first fragment matches the second ACE (a deny) because all Layer 3 and Layer 4 information is present. The remaining fragments in the packet do not match the second ACE because they are missing Layer 4 information. Instead, they match the third ACE (a permit).

  Because the first fragment was denied, host 10.1.1.2 cannot reassemble a complete packet, so packet B is effectively denied. However, the later fragments that are permitted will consume bandwidth on the network and resources of host 10.1.1.2 as it tries to reassemble the packet.
- Fragmented packet C is from host 10.2.2.2, port 65001, going to host 10.1.1.3, port ftp. If this packet is fragmented, the first fragment matches the fourth ACE (a deny). All other fragments also match the fourth ACE because that ACE does not check any Layer 4 information and because Layer 3 information in all fragments shows that they are being sent to host 10.1.1.3, and the earlier permit ACEs were checking different hosts.

Configure time ranges with ACLs

This task enables you to configure time ranges and apply them to access control lists (ACLs) for controlling network traffic during defined periods, such as work hours or holidays.

- Define time ranges for work hours and holidays.
- Apply time ranges to extended ACLs to permit or deny traffic based on the specified schedule.

Use this task when you need to restrict or allow network access during specific times, such as company holidays or regular work hours. Time ranges can be referenced in ACLs to automate access control based on the calendar.

This is relevant for organizations that require scheduled access policies, such as denying traffic during holidays or permitting traffic during work hours.

- Applicable to both numbered and named extended ACLs.

Before you begin

Ensure you have access to the Switch CLI and appropriate privileges to configure ACLs and time ranges.

- Identify the periods (work hours, holidays) for which you want to define time ranges.

Follow these steps to configure time ranges and apply them to ACLs:

Procedure

Step 1 Verify configured time ranges for *workhours* and configure January 1, 2006, as a company holiday.

Example:

```
Switch# show time-range

                        time-range entry: new_year_day_2003 (inactive)
                        absolute start 00:00 01 January 2006 end 23:59 01 January 2006
                        time-range entry: workhours (inactive)
                        periodic weekdays 8:00 to 12:00
                        periodic weekdays 13:00 to 17:00
```

This step shows how to verify the time ranges configured for work hours and holidays.

Step 2 Apply a time range in an extended ACL to deny TCP traffic during holidays and permit TCP traffic during work hours.

Example:

```
Switch> enable
                         Switch# configure terminal
                        Switch(config)# access-list 188 deny tcp any any time-range new_year_day_2006

                         Switch(config)# access-list 188 permit tcp any any time-range workhours
                         Switch(config)# exit
                         Switch# show access-lists

                         Extended IP access list 188
                         10 deny tcp any any time-range new_year_day_2006 (inactive)
                         20 permit tcp any any time-range workhours (inactive)
```

This step demonstrates how to create and verify an extended access list that uses time ranges to control TCP traffic based on the defined schedule.

Step 3 Use named ACLs to permit and deny traffic based on time ranges.

Example:

```
Switch> enable
                         Switch# configure terminal
                         Switch(config)# ip access-list extended deny_access
                         Switch(config-ext-nacl)# deny tcp any any time-range new_year_day_2006
                         Switch(config-ext-nacl)# exit
                         Switch(config)# ip access-list extended may_access
                         Switch(config-ext-nacl)# permit tcp any any time-range workhours
                         Switch(config-ext-nacl)# end
                         Switch# show ip access-lists

                         Extended IP access list lpip_default
                         10 permit ip any any
                         Extended IP access list deny_access
                         10 deny tcp any any time-range new_year_day_2006 (inactive)
                         Extended IP access list may_access
```

```
10 permit tcp any any time-range workhours (inactive)
```

This step shows how to configure named extended ACLs to permit or deny traffic using time ranges.

After completing these steps, time ranges are configured and applied to ACLs. The Switch will permit or deny traffic according to the defined schedules, such as work hours or holidays.

Configure a time range for an IP access control list

This task enables you to apply time-based restrictions to an IP access control list (ACL), allowing you to control network traffic based on specific days and hours.

Use this task when you need to restrict HTTP traffic during business hours on weekdays and permit UDP traffic only during weekends within defined time ranges. This is useful for organizations that require scheduled access control for different protocols.

Procedure

Deny HTTP traffic on IP on Monday through Friday between 8:00 a.m. and 6:00 p.m. Allow UDP traffic only on Saturday and Sunday from noon to 8:00 p.m.

Example:

```
Switch> enable
                                                  Switch# configure terminal
                                                  Switch(config)# time-range no-http
                                                  Switch(config)# periodic weekdays 8:00 to 18:00
                                                  Switch(config)# time-range udp-yes
                                                  Switch(config)# periodic weekend 12:00 to 20:00
                                                  Switch(config)# ip access-list extended strict
                                                  Switch(config-ext-nacl)# deny tcp any any eq www
time-range no-http
                                                Switch(config-ext-nacl)# permit udp any any time-range
 udp-yes
                                                  Switch(config-ext-nacl)# exit
                                                  Switch(config)# interface gigabitethernet1/0/1

                                                  Switch(config-if)# ip access-group strict in
                                                  Switch(config-if)# end
```

This configuration example shows how to use time ranges in an IP ACL to control protocol access based on days and hours.

After completing this task, HTTP traffic will be denied during weekdays from 8:00 a.m. to 6:00 p.m., and UDP traffic will be allowed only on weekends from noon to 8:00 p.m. The ACL will enforce these time-based restrictions as configured.

Include comments in ACLs

Configure comments in ACLs to document and clarify the purpose of each entry, making the ACL easier to understand and maintain.

Use this task when you need to add descriptive remarks to IP standard or extended ACLs for better documentation and easier management. Remarks can be placed before or after permit or deny statements, but consistency is recommended for clarity.

Before you begin

Ensure you are in global configuration mode on the Switch before adding or removing remarks in ACLs.

Follow these steps to include comments in ACLs:

Procedure

Step 1 Use the **remark** keyword to add a comment to an IP standard or extended ACL entry.

You can add remarks to any IP standard or extended ACL. Each remark line is limited to 100 characters. Place remarks consistently before or after permit or deny statements to avoid confusion.

Example:

```
Switch> enable
      Switch# configure terminal
      Switch(config)# access-list 1 remark Permit only user1 workstation through
      Switch(config)# access-list 1 permit 171.69.2.88
      Switch(config)# access-list 1 remark Do not allow user2 through
      Switch(config)# access-list 1 deny 171.69.3.13
      Switch(config)# end
```

Step 2 To remove a remark, use the **no** form of the command.

Step 3 For named IP ACLs, use the **remark** access-list configuration command to add a comment.

Example:

```
Switch> enable
      Switch# configure terminal
      Switch(config)# ip access-list extended telnetting
      Switch(config-ext-nacl)# remark Do not allow subnet1 subnet to telnet out
      Switch(config-ext-nacl)# deny tcp host 171.69.2.88 any eq telnet
      Switch(config-ext-nacl)# end
```

Step 4 To remove a remark from a named ACL, use the **no** form of the command.

Create an ACL and a VLAN map to deny a packet

This task shows how to configure an ACL and a VLAN map to deny packets that match the ACL criteria. It is useful for controlling network traffic and enhancing security by dropping unwanted packets.

Use this procedure when you need to restrict certain types of traffic on your network by creating an ACL and applying it to a VLAN map. In this example, TCP packets are denied based on the ACL configuration.

Before you begin

Ensure you have access to the switch CLI with sufficient privileges to configure access lists and VLAN maps.

Follow these steps to create an ACL and a VLAN map to deny a packet:

Procedure

Create an ACL and a VLAN map to deny TCP packets as shown in the following example.

Example:

```
Switch> enable

        Switch# configure terminal

        Switch(config)# ip access-list extended ip1

        Switch(config-ext-nacl)# permit tcp any any

        Switch(config-ext-nacl)# exit

        Switch(config)# vlan access-map map_1 10

        Switch(config-access-map)# match ip address ip1

        Switch(config-access-map)# action drop

        Switch(config-access-map)# end
```

In the first map, any packets that match the *ip1* ACL (TCP packets) are dropped. The *ip1* ACL permits any TCP packet and no other packets. Because there is a match clause for IP packets in the VLAN map, the default action is to drop any IP packet that does not match any of the match clauses.

After completing this task, TCP packets matching the ACL are dropped by the VLAN map, and any IP packet not matching the ACL is also dropped by default.

Create an ACL and a VLAN map to permit a packet

This task enables you to configure an access control list (ACL) and a VLAN map to permit UDP packets and forward any packets that match the specified ACL.

Use this task when you need to control packet forwarding in your network by permitting specific UDP packets and dropping others that do not match the ACL criteria. This is relevant in scenarios where you want to allow only certain types of traffic through a VLAN.

Before you begin

No prerequisites are explicitly stated for this configuration.

Follow these steps to create an ACL and a VLAN map to permit a packet:

Procedure

Create an ACL and a VLAN map to permit UDP packets and forward any packets that match the ACL.

Example:

```
Switch> enable
                        Switch# configure terminal
                        Switch(config)# ip access-list extended ip2
                        Switch(config-ext-nacl)# permit udp any any
                        Switch(config-ext-nacl)# exit
                        Switch(config)# vlan access-map map_1 20
                        Switch(config-access-map)# match ip address ip2
                        Switch(config-access-map)# action forward
                        Switch(config-access-map)# exit
```

This example shows how to create a VLAN map to permit a packet. ACL *ip2* permits UDP packets and any packets that match the *ip2* ACL are forwarded. In this map, any IP packets that did not match any of the previous ACLs (that is, packets that are not TCP packets or UDP packets) would get dropped.

After completing this task, UDP packets and any packets that match the ACL are forwarded, while packets that do not match any ACLs are dropped.

Configure default action for VLAN map to drop IP packets and forward MAC packets

Configure a VLAN map to control packet forwarding and dropping based on protocol type. The task achieves selective forwarding and dropping of packets using access lists and VLAN access maps.

- Enable granular control over network traffic between VLANs.
- Prevent unwanted access to servers on other VLANs.

This task is relevant when you need to restrict access between VLANs, such as denying access to a server on another VLAN, by configuring default actions for IP and MAC packets.

Used with standard ACL 101 and extended named access lists **IGMP-match** and **TCP-match**, the VLAN map will have the following results:

- Forward all UDP packets.
- Drop all IGMP packets.
- Forward all TCP packets.
- Drop all other IP packets.
- Forward all non-IP packets.

Before you begin

Ensure you have access to the Switch CLI and appropriate privileges to configure VLAN maps and access lists.

- Standard access list 101 must be defined.
- Extended named access lists **IGMP-match** and **TCP-match** must be available.

Follow these steps to configure default actions for VLAN map to drop IP packets and forward MAC packets:

Procedure

Configure the VLAN map with default actions for IP and MAC packets using access lists.

Example:

```
Switch> enable
      Switch# configure terminal
      Switch(config)# access-list 101 permit udp any any
      Switch(config)# ip access-list extended igmp-match
      Switch(config-ext-nacl)# permit igmp any any
      Switch(config)# action forward
      Switch(config-ext-nacl)# permit tcp any any
      Switch(config-ext-nacl)# exit
      Switch(config)# vlan access-map drop-ip-default 10
      Switch(config-access-map)# match ip address 101
      Switch(config-access-map)# action forward
      Switch(config-access-map)# exit
      Switch(config)# vlan access-map drop-ip-default 20
      Switch(config-access-map)# match ip address igmp-match
      Switch(config-access-map)# action drop
      Switch(config-access-map)# exit
      Switch(config)# vlan access-map drop-ip-default 30
      Switch(config-access-map)# match ip address tcp-match
      Switch(config-access-map)# action forward
      Switch(config-access-map)# end
```

The configuration example demonstrates how to use access lists and VLAN access maps to selectively forward or drop packets based on protocol type.

After completing the configuration, the VLAN map will forward UDP, TCP, and non-IP packets, and drop IGMP and other IP packets as specified. Access to servers on other VLANs will be denied according to the defined rules.

What to do next

Verify the VLAN map operation by testing connectivity and packet forwarding between VLANs.

- Monitor traffic to ensure only permitted packets are forwarded.
- Check logs for dropped packets to confirm the configuration is working as intended.

Configure default action of dropping MAC packets and forwarding IP packets

This task configures a VLAN map with a default action to drop MAC packets and forward IP packets, using MAC extended access lists to specify exceptions for certain hosts and protocols.

Use this configuration when you need to restrict non-IP traffic on a VLAN, allowing only specific MAC hosts or protocols while ensuring all IP packets are forwarded. This is useful for environments where non-IP traffic should be limited for security or performance reasons.

Before you begin

No explicit prerequisites are stated. Ensure you are in privileged EXEC mode and have access to the Switch CLI.

Follow these steps to configure a VLAN map that drops MAC packets by default and forwards IP packets:

Procedure

Configure MAC extended access lists and VLAN access-map to control packet forwarding.

In this example, the VLAN map has a default action of drop for MAC packets and a default action of forward for IP packets. Used with MAC extended access lists **good-hosts** and **good-protocols**, the map will have the following results:

- Forward MAC packets from hosts 0000.0c00.0111 and 0000.0c00.0211
- Forward MAC packets with decnet-iv or vines-IP protocols
- Drop all other non-IP packets
- Forward all IP packets

Example:

```
Switch> enable
      Switch# configure terminal
      Switch(config)# mac access-list extended good-hosts
      Switch(config-ext-macl)# permit host 000.0c00.0111 any
      Switch(config-ext-macl)# permit host 000.0c00.0211 any
      Switch(config-ext-nacl)# exit
      Switch(config)# action forward
      Switch(config-ext-macl)# mac access-list extended good-protocols
      Switch(config-ext-macl)# permit any any vines-ip
      Switch(config-ext-nacl)# exit
      Switch(config)# vlan access-map drop-mac-default 10
      Switch(config-access-map)# match mac address good-hosts
      Switch(config-access-map)# action forward
      Switch(config-access-map)# exit
      Switch(config)# vlan access-map drop-mac-default 20
      Switch(config-access-map)# match mac address good-protocols
      Switch(config-access-map)# action forward
      Switch(config-access-map)# end
```

After completing this configuration, the VLAN map will forward MAC packets from specified hosts and protocols, drop all other non-IP packets, and forward all IP packets as intended.

Configure a VLAN map with default action to drop all packets

This task enables you to configure a VLAN access map with a default action to drop all packets except those explicitly permitted by access lists.

- Helps enforce strict packet filtering by forwarding only specified TCP and MAC packets.

Use this task when you need to restrict network traffic on a VLAN, allowing only certain TCP and MAC packets to be forwarded while dropping all other packets.

The configuration uses access lists **TCP-match** and **good-hosts** to define which packets are forwarded.

- Applicable in environments requiring granular control over VLAN traffic.

Before you begin

Ensure that access lists **TCP-match** and **good-hosts** are already configured as referenced in Examples 2 and 3.

- Access to device configuration mode is required.

Follow these steps to configure a VLAN map with default action to drop all packets:

Procedure

Configure the VLAN access map with a default action to drop all packets, forwarding only specified TCP and MAC packets.

Example:

```
Device> enable
            Device# configure terminal
            Device(config)# vlan access-map drop-all-default 10
            Device(config-access-map)# match ip address tcp-match
            Device(config-access-map)# action forward
            Device(config-access-map)# exit
            Device(config)# vlan access-map drop-all-default 20
            Device(config-access-map)# match mac address good-hosts
            Device(config-access-map)# action forward
            Device(config-access-map)# end
```

In this example, the VLAN map has a default action of drop for all packets (IP and non-IP). Used with access lists **TCP-match** and **good-hosts**, the map will have the following results:

- Forward all TCP packets
- Forward MAC packets from hosts 0000.0c00.0111 and 0000.0c00.0211
- Drop all other IP packets
- Drop all other MAC packets

After completing this task, only TCP packets matching **TCP-match** and MAC packets matching **good-hosts** will be forwarded. All other IP and MAC packets will be dropped by default.

Example: Using VLAN Maps in a Network

Configure a wiring closet VLAN map to drop HTTP traffic

This task enables you to configure a VLAN map on Switch A to prevent HTTP traffic from Host X (10.1.1.32) to Host Y (10.1.1.34) from being bridged to Switch B.

Use this configuration when you need to restrict specific traffic, such as HTTP, between hosts in a wiring closet environment. This is useful for enforcing security policies or limiting access between switchs connected to different switches.

Before you begin

Ensure you have access to Switch A and are in privileged EXEC mode.

Perform these steps to configure a VLAN map to drop HTTP traffic from Host X to Host Y:

Procedure

Step 1 Define the IP access list *HTTP* that permits TCP traffic on the HTTP port from Host X to Host Y.

Example:

```
Switch> enable
                        Switch# configure terminal
                        Switch(config)# ip access-list extended http
                        Switch(config-ext-nacl)# permit tcp host 10.1.1.32 host 10.1.1.34 eq www
                        Switch(config-ext-nacl)# end
```

Step 2 Create VLAN access map *map2* to drop traffic matching the *HTTP* access list and forward all other IP traffic.

Example:

```
Switch> enable
                        Switch# configure terminal
                        Switch(config)# vlan access-map map2 10
                        Switch(config-access-map)# match ip address http
                        Switch(config-access-map)# action drop
                        Switch(config-access-map)# exit
                        Switch(config)# ip access-list extended match_all
                        Switch(config-ext-nacl)# permit ip any any
                        Switch(config-ext-nacl)# exit
                        Switch(config)# vlan access-map map2 20
                        Switch(config-access-map)# match ip address match_all
                        Switch(config-access-map)# action forward
                        Switch(config-access-map)# end
```

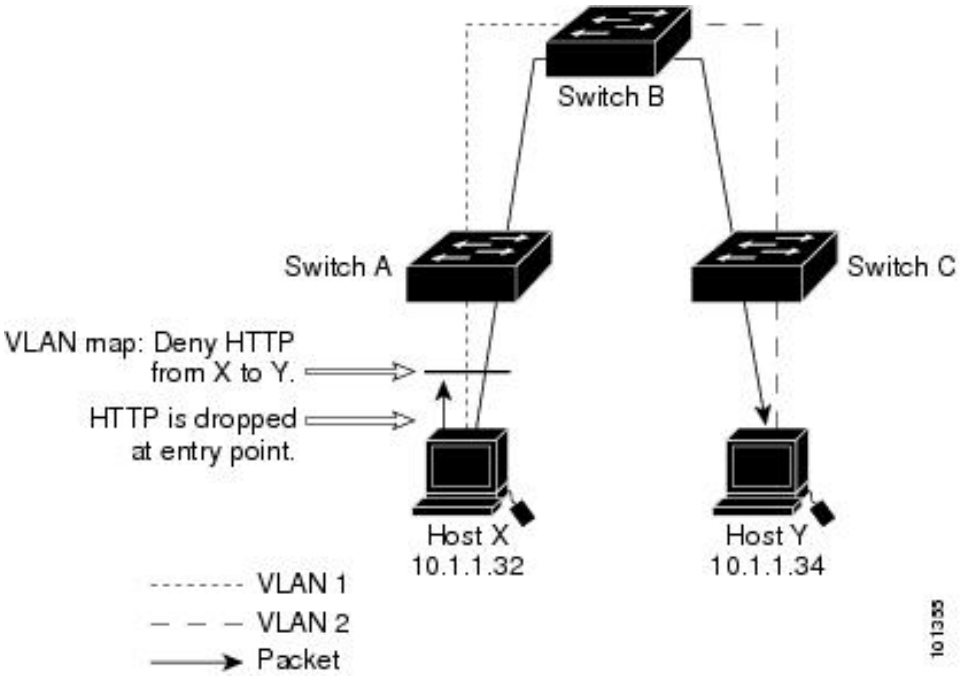
Step 3 Apply VLAN access map *map2* to VLAN 1.

Example:

```
Switch> enable
                        Switch# configure terminal
                        Switch(config)# vlan filter map2 vlan 1
                        Switch(config)# end
```

Step 4 Review the wiring closet configuration diagram for reference.

Figure 11: Wiring Closet Configuration



After completing these steps, HTTP traffic from Host X (10.1.1.32) to Host Y (10.1.1.34) will be dropped at Switch A and not bridged to Switch B. All other IP traffic will be forwarded as usual.

Restricting access to servers on other VLANs

A restriction of access to servers on other VLANs is a network security configuration that

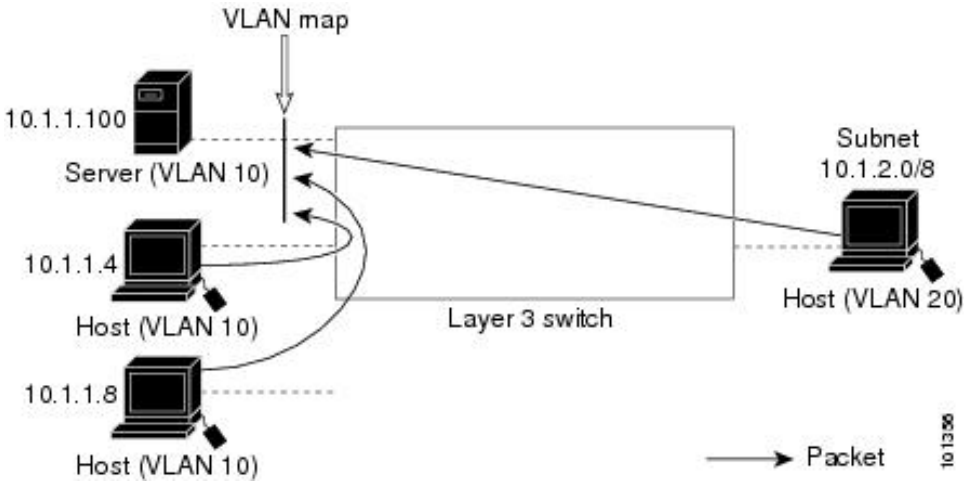
- denies access to specific hosts or subnets from reaching a server on a different VLAN
- applies VLAN maps or access control lists to enforce the restriction, and
- targets individual hosts or entire subnets based on network requirements.

Configuration details for restricting access to servers on other VLANs

You can restrict access to a server on another VLAN. For example, server 10.1.1.100 in VLAN 10 needs to have access denied to these hosts:

- Hosts in subnet 10.1.2.0/8 in VLAN 20 should not have access.
- Hosts 10.1.1.4 and 10.1.1.8 in VLAN 10 should not have access.

Figure 12: Restricting access to servers on other VLANs



Deny access to a SERVER on another VLAN

This task enables you to deny access to a SERVER located on another VLAN by creating and applying VLAN maps and access control lists (ACLs).

Use this task when you need to restrict access between VLANs, such as preventing hosts in a specific subnet or IP addresses from reaching a SERVER on another VLAN. This is commonly required in environments where network segmentation and security are important.

Before you begin

Ensure you have access to the Switch CLI and appropriate privileges to configure VLAN maps and ACLs.

Follow these steps to deny access to a SERVER on another VLAN:

Procedure

Step 1 Create an IP access control list (ACL) to match packets from the specified hosts and subnet.

Example:

```
Switch> enable
                                                  Switch# configure terminal
                                                  Switch(config)# ip access-list extended SERVER1_ACL
                                                 Switch(config-ext-nacl)# permit ip 10.1.2.0 0.0.0.255
 host 10.1.1.100
                                                 Switch(config-ext-nacl)# permit ip host 10.1.1.4 host
 10.1.1.100
                                                 Switch(config-ext-nacl)# permit ip host 10.1.1.8 host
 10.1.1.100
                                                  Switch(config-ext-nacl)# end
```

Step 2 Define a VLAN map using the ACL to drop matching IP packets and forward others.

Example:

```
Switch> enable
                                                  Switch# configure terminal
```

```
Switch(config)# vlan access-map SERVER1_MAP
Switch(config-access-map)# match ip address SERVER1_ACL

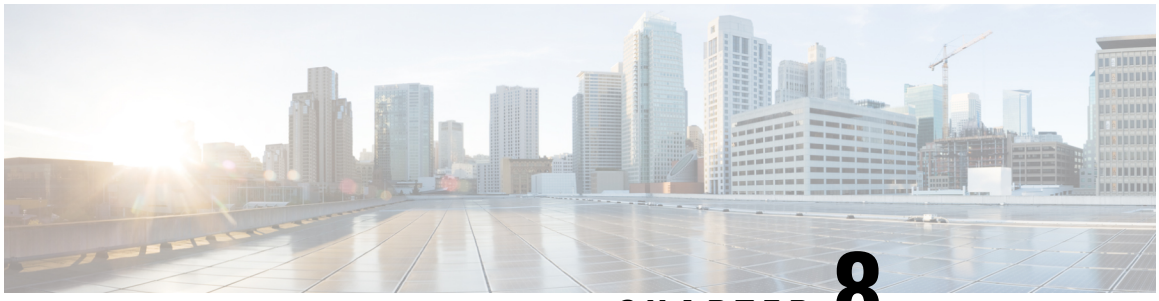
Switch(config-access-map)# action drop
Switch(config)# vlan access-map SERVER1_MAP 20
Switch(config-access-map)# action forward
Switch(config-access-map)# end
```

Step 3 Apply the VLAN map to VLAN 10 to enforce the access restrictions.

Example:

```
Switch> enable
                                                    Switch# configure terminal
                                                   Switch(config)# vlan filter SERVER1_MAP vlan-list 10

                                                    Switch(config)# end
```



CHAPTER 8

IPv6 ACLs

- IPv6 access control lists, on page 189
- Guidelines and restrictions for IPv6 ACLs, on page 189
- Supported ACLs, on page 190
- Types of ACL, on page 190
- ACL precedences, on page 191
- VLAN maps, on page 191
- Interactions with other features and switches, on page 192
- Configure an IPv6 ACL, on page 193
- IPv6 ACL monitoring commands, on page 201

IPv6 access control lists

An IPv6 access control list is a set of rules that

- limit access to a particular interface
- are configured on the device and applied to the management interface and dynamic interfaces, and
- support options such as source, destination, source, and destination ports.

IPv6 access control lists can be used to create preauthentication ACLs for web authentication, allowing certain types of traffic before authentication is complete.

Guidelines and restrictions for IPv6 ACLs

Guidelines

- When configuring an ACL, you can enter any keywords regardless of platform support. When you apply the ACL to an interface that requires hardware forwarding (physical ports or SVIs), the switch checks to determine whether ACL can be supported on the interface or not. If the ACL is not supported on the interface, the ACL is rejected.
- If a downloadable ACL contains any type of duplicate entries, the entries are not auto merged. As a result, the 802.1X session authorization fails. Ensure that the downloadable ACL is optimized without any duplicate entries, for example port-based and name-based entries for the same port.

- When you apply a scale ACL to an interface that does not program TCAM for a protocol and the ACLs that have been unloaded, it can impact the existing normal movement of traffic for other protocols.

Restrictions for IPv6 ACLs

IPv6 supports only named ACLs. With IPv4 ACLs, you can configure standard and extended numbered IP ACLs, named IP ACLs, and MAC ACLs. The switch supports most Cisco IOS-supported IPv6 ACLs with some exceptions.

- The switch does not support matching on these keywords:
  - flowlabel
  - routing header
  - undetermined-transport
  - reflexive ACLs (the reflect keyword)
  - **vrf-also** keyword is mutually exclusive of IPv6 access-class line command.
  - MAC-based ACLs
- If an ACL is applied to an interface and you attempt to add an access control entry (ACE) with an unsupported keyword, the switch does not allow the ACE to be added to the ACL that is currently attached to the interface.
- Time-to-live (TTL) classification is not supported on ACLs.
- Egress ACL lookup is not supported for injected traffic that is forwarded by the software.
- ACLs support only layer 3 interfaces (such as routed interfaces and VLAN interfaces).

Supported ACLs

A supported ACL is a traffic filtering mechanism that includes port ACLs to control access for traffic entering a layer 2 interface.

Types of supported ACLs

The switch supports three types of ACLs to filter the traffic:

- Port ACLs access-control traffic entering a layer 2 interface. You can apply port ACLs to a layer 2 interface in each direction to each access list type—IPv4, IPv6, and MAC.
- Router ACLs access-control traffic routed between VLANs and are applied to Layer 3 interfaces in a specific direction (inbound or outbound).

Types of ACL

The following sections provide information on the types of ACL:

Per-user IPv6 ACLs

A per-user IPv6 ACL is an access control list that

- applies IPv6 access control entries to individual users, and
- configures full access control entries as text strings on Cisco Secure Access Control Server

Filter ID IPv6 ACLs

A filter ID IPv6 ACL is an access control list that

- stores the full ACEs and the ACL name (filter-ID) on the device, and
- configures only the `filter-ID` on the Cisco Secure ACS, and

ACL precedences

An ACL precedence is a filtering order that determines which ACL type filters the packets first for ingress and egress traffic. It

- applies port ACL, and router ACL in ingress path, and
- applies router ACL, and port ACL, in egress path

VLAN maps

A VLAN map is a network traffic control mechanism that

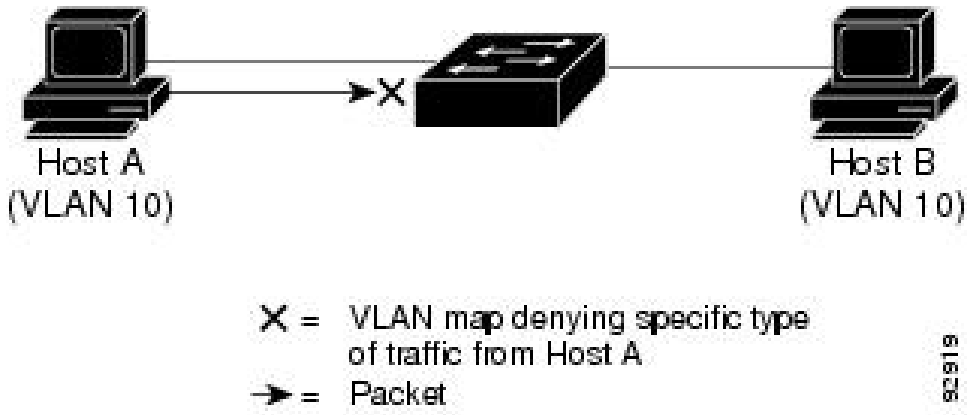
- applies to all packets bridged within a VLAN in the switch
- enforces access control for non-IP protocols through MAC addresses and Ethertype, and
- permits or denies packet forwarding based on the action specified in the map.

VLAN map reference information

VLAN ACLs or VLAN maps are used to control the network traffic within a VLAN. You can apply VLAN maps to all packets that are bridged within a VLAN in the switch. VACLs are strictly for the security packet filtering and for redirecting traffic to specific physical interfaces. VACLs are not defined by direction (ingress or egress).

All non-IP protocols are access-controlled through MAC addresses and Ethertype using MAC VLAN maps. (IP traffic is not access-controlled by MAC VLAN maps.) You can enforce VLAN maps only on packets going through the switch; you cannot enforce VLAN maps on traffic between hosts on a hub or on another switch that is connected to this switch.

Figure 13: Using VLAN maps to control traffic



Interactions with other features and switches

An interaction with other features and switches is a network behavior that

- determines how IPv6 ACLs affect packet routing and bridging
- controls ACL configuration and error handling, and
- manages hardware memory limitations and packet drop statistics.

IPv6 ACL interactions with features and switches

IPv6 ACLs interact with other features and switches in several ways:

- If an IPv6 router ACL is configured to deny a packet, the packet is not routed. A copy of the packet is sent to the Internet Control Message Protocol (ICMP) queue to generate an ICMP unreachable message for the frame.
- If a bridged frame is to be dropped due to a port ACL, the frame is not bridged.
- You can create both IPv4 and IPv6 ACLs on a switch, and you can apply both IPv4 and IPv6 ACLs to the same interface. Each ACL must have a unique name; an error message appears if you try to use a name that is already configured.

  You use different commands to create IPv4 and IPv6 ACLs and to attach IPv4 or IPv6 ACLs to the same Layer 2 or Layer 3 interface. If you use the wrong command to attach an ACL (for example, an IPv4 command to attach an IPv6 ACL), you receive an error message.
- You cannot use MAC ACLs to filter IPv6 frames. MAC ACLs can only filter non-IP frames.
- If the hardware memory is full, packets are dropped on the interface and an unload error message is logged.

  If the hardware memory is full, for any additional configured ACLs, packets are dropped to the CPU, and the ACLs are applied in software. When the hardware is full a message is printed to the console indicating the ACL has been unloaded and the packets will be dropped on the interface.

Configure an IPv6 ACL

Default configuration for IPv6 ACLs

This reference describes the default configuration for IPv6 access control lists (ACLs).

The default IPv6 ACL configuration is as follows:

```
Device# show access-lists preauth_ipv6_acl

IPv6 access list preauth_ipv6_acl (per-user)
permit udp any any eq domain sequence 10
permit tcp any any eq domain sequence 20
permit icmp any any nd-ns sequence 30
permit icmp any any nd-na sequence 40
permit icmp any any router-solicitation sequence 50
permit icmp any any router-advertisement sequence 60
permit icmp any any redirect sequence 70
permit udp any eq 547 any eq 546 sequence 80
permit udp any eq 546 any eq 547 sequence 90
deny ipv6 any any sequence 100
```

Configure IPv6 ACLs

Configure IPv6 ACLs to filter IPv6 traffic and control access to your network switches. This task enables you to define permit and deny conditions for IPv6 packets.

- Improve network security by restricting IPv6 traffic based on protocol, source, and destination addresses.
- Apply granular control over IPv6 packet flow using access control lists.

IPv6 ACLs are used to filter IPv6 traffic and enforce security policies on network switches. This procedure is relevant when you need to restrict or permit IPv6 packets based on specific criteria.

Perform this task when configuring IPv6 security or managing access to resources in an IPv6-enabled environment.

- Use this task to define and verify IPv6 ACLs for routers and switches supporting IPv6.

Before you begin

Ensure you have access to the Switch CLI and privileged EXEC mode. No explicit prerequisites are stated in the source, but access to the Switch is implied.

- Have the IPv6 addresses and protocols you want to filter ready.

Follow these steps to configure IPv6 ACLs:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password, if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **ipv6 access-list** {*list-name* | **log-update threshold** | **role-based** *list-name*} command to define an IPv6 ACL name, and enters IPv6 access list configuration mode.

Example:

```
Switch(config)# ipv6 access-list example_acl_list
```

Step 4 Use the {**deny** | **permit**} protocol {*source-ipv6-prefix/* |*prefix-length* |**any threshold**| **host** *source-ipv6-address*} [operator [*port-number*]] { *destination-ipv6-prefix/ prefix-length* | **any** | **host** *destination-ipv6-address*} [operator [*port-number*]][**dscp** *value*] [**fragments**] [**log**] [**log-input**][**sequence** *value*] [**time-range** *name*] command to specify permit or deny conditions for an IPv6 ACL.

Example:

```
Switch(config-ipv6-acl)# permit tcp 2001:DB8:0300:0201::/32 eq telnet any
```

Table 17: IPv6 ACL configurable paramets and their description

Parameter	Description
ahp, **esp**, **icmp**, **ipv6**, **pcp**, **stcp**, **TCP**, or **UDP**, or an integer in the range 0 to 255 representing an IPv6 protocol number.	Provide any supported IPv6 protocols.
source-ipv6-prefix/prefix-length or *destination-ipv6-prefix/ prefix-length*	Provide the source or destination IPv6 network or class of networks for which to set deny or permit conditions, specified in hexadecimal and using 16-bit values between colons (see RFC 2373)
any	Provide an abbreviation for the IPv6 prefix ::/0
host *source-ipv6-address* or *destination-ipv6-address*	Specify the source or destination IPv6 host address for which to set deny or permit conditions. Specify the values in hexadecimal using 16-bit values between colons.
lt (less than), **gt** (greater than), **eq** (equal), **neq** (not equal), and **range**	These are the supported operators. Specify an operand that compares the source or destination ports of the specified protocol. These are optional parameters.
source-ipv6-prefix/prefix-lengthdestination-ipv6-prefix/prefix-length	Provide the port. If the operator follows the *source-ipv6-prefix/prefix-length* argument, it must match the source port. If the operator follows the *destination-ipv6-prefix/prefix-length* argument, it must match the destination port.

Parameter	Description
port-number	Provide any decimal number from 0 to 65535 or the name of a TCP or UDP port. You can use TCP port names only when filtering TCP. You can use UDP port names only when filtering UDP. It is an optional parameter.
dscp	Specify a value to match a differentiated services code point value against the traffic class value in the Traffic Class field of each IPv6 packet header. The acceptable range is from 0 to 63. It is an optiional parameter.
fragments	Specify to check noninitial fragments. This keyword is visible only if the protocol is ipv6. It is an optiional parameter.
loglog-input	Specify to cause a logging message to be sent to the console about the packet that matches the entry. Enter **log-input** to include the input interface in the log entry. Logging is supported only for router ACLs. It is an optiional parameter.
sequence *value*	Specify the sequence number for the access list statement. The acceptable range is from 1 to 4,294,967,295. It is an optiional parameter.
time-range	Specify a name to enter the time range that applies to the deny or permit statement. It is an optiional parameter.

Step 5 Use the {**deny** | **permit**} **TCP** {*source-ipv6-prefix/prefix-length* | **any** | **host** *source-ipv6-address*} [**operator** [**port-number**]] {*destination-ipv6- prefix/prefix-length* | **any** | **host** *destination-ipv6-address*} [operator [*port-number*]] [**ACK**] [**dscp** *value*] [**established**] [**fin**] [**log**] [**log-input**] [**neq** {**port** | protocol}] [**psh**] [**range** {**port** | **protocol**}] [**RST**] [**sequence** *value*] [**syn**] [**time-range** *name*] [**urg**] command to specify permit or deny conditions for an IPv6 ACL.

Example:

```
Switch(config-ipv6-acl)# deny tcp host 2001:DB8:1::1 any log-input
```

Enter **TCP** for Transmission Control Protocol. The parameters are the same as those described in step 4, with these additional optional parameters:

Table 18: TCP optional paramets

Parameter	Description
ACK	Acknowledgment bit set.

Parameter	Description
established	An established connection. A match occurs if the TCP datagram has the ACK or RST bits set.
fin	Finished bit set; no more data from sender.
neq { *port* |**protocol**}	Matches only packets that are not on a given port number.
RST	Reset bit set.
syn	Synchronize bit set.
urg	Urgent pointer bit set.
psh	Push function bit set.
range { *port* |**protocol**}	Matches only packets in the port number range.

Step 6 Use the **end** command to exit IPv6 access list configuration mode and return to privileged EXEC mode.

Example:

```
Switch(config-ipv6-acl)# end
```

Step 7 (Optional) Use the **show ipv6 access-list** command to verifies that IPv6 ACLs configuration

Example:

```
Switch# show ipv6 access-list
```

Attach an IPv6 ACL to an interface

Attach an IPv6 ACL to an interface to control access and manage traffic flow. This task allows you to restrict inbound or outbound traffic based on IPv6 addresses.

You can apply an ACL to outbound or inbound traffic on Layer 3 interfaces, or to inbound traffic on Layer 2 interfaces. You can also apply ACLs only to inbound management traffic on Layer 3 interfaces.

This task is relevant when you need to control access to an interface using IPv6 ACLs for security or traffic management purposes.

Before you begin

Ensure you have created the IPv6 ACL that you want to attach to the interface.

Follow these steps to attach an IPv6 ACL to an interface:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password, if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **interface** *interface-id* command to identify a Layer 2 interface (for port ACLs) or Layer 3 interface (for router ACLs) on which to apply an access list, and enters interface configuration mode.

Example:

```
Switch(config)# interface gigabitethernet1/0/1
```

Step 4 Use the **no switchport** command to return the interface to the routed-interface status and erases all further layer 2 configuration.

Example:

```
Switch(config-if)# no switchport
```

Step 5 Use the **ipv6 address***ipv6-address* command to configure an IPv6 address on a layer 3 interface (for router ACLs).

Example:

```
Switch(config-if)# ipv6 address 2001:DB8::1
```

Step 6 Use the **ipv6 traffic-filter** *access-list-name* {**in** | **out**} command to apply the access list to incoming or outgoing traffic on the interface.

Example:

```
Switch(config-if)# ipv6 traffic-filter acl1 in
```

Step 7 Use the **end** command to exit IPv6 access list configuration mode and return to privileged EXEC mode.

Example:

```
Switch(config-ipv6-acl)# end
```

Configure an IPv6 ACL in template mode

This task describes how to configure an IPv6 access control list (ACL) in template mode to manage network traffic and enhance security on your Switch interfaces.

Note You can configure **ipv6 traffic-filter** command in the template configuration mode. You can configure the **source template** command only once to an interface.

Use this procedure when you need to apply IPv6 ACLs to interfaces using templates for consistent and scalable configuration. Begin in privileged EXEC mode.

Before you begin

Ensure you have access to the Switch with appropriate privileges and that you are in privileged EXEC mode.

Follow these steps to configure an IPv6 ACL in template mode:

Procedure

Step 1 Use the **ipv6 access-list** {*list-name* | **log-update threshold** | **role-based** *list-name*} command to define an IPv6 ACL name, and enters IPv6 access list configuration mode.

Example:

```
Switch(config)# ipv6 access-list v6acl10
```

Step 2 Use the **ipv6 access-list** {*list-name* | **log-update threshold** | **role-based** *list-name*} command to define an IPv6 ACL name, and enters IPv6 access list configuration mode.

Example:

```
Switch(config-ipv6-acl)#ipv6 access-list v6acl11
```

Step 3 Use the **exit** command to exit access-list configuration mode.

Example:

```
Switch(config-ipv6-acl)#exit
```

Step 4 Use the **template** command to create a user template and enters template configuration mode.

Example:

```
Switch(config)# template test
```

Step 5 Use the **ipv6 traffic-filter** {*access-list-number* | *name*} {**in** | **out**} command to control access to the specified interface.

Example:

```
Switch(config-template)# ipv6 traffic-filter v6acl10 in
```

Table 19: IPv6 Traffic filter parametrs

Parameter	Description
access-list-number	Defines the access list. The access list can be a number.
name	Defines the access list. The access list can be a name
in	Directs the access list in the incoming direction of the interface
out	Direct the access list in the outgoing direction of the interface.

Step 6 Use the **exit** command to exit template configuration mode and return to privileged EXEC mode.

Example:

```
Switch(config-template)# exit
```

Step 7 Use the **interface** *interface-id* command to identify a specific interface for configuration.

Example:

```
Switch(config)# interface gigabitethernet1/0/1
```

The interface can be a layer 2 interface (port ACL), or a layer 3 interface (router ACL).

Step 8 Use the **ipv6 traffic-filter** {*access-list-number* | *name*} {**in** | **out**} command to controls access to the specified interface.

Example:

```
Switch(config-if)# ipv6 traffic-filter v6acl11 out
```

Table 20: IPv6 traffic filter parametrs

Parameter	Description
access-list-number	Defines the access list. The access list can be a number.
name	Defines the access list. The access list can be a name
in	Directs the access list in the incoming direction of the interface
out	Direct the access list in the outgoing direction of the interface.

Step 9 (Optional) Use the **source template** *name* command to apply an interface template to a target.

Example:

```
Switch(config)# source template test
```

Configure a VLAN map

This task enables you to configure a VLAN map, which allows you to filter traffic on VLANs using access control lists (ACLs). By following these steps, you can create, match, and apply VLAN maps to control packet forwarding or dropping within your network.

Use this task when you need to apply ACLs to VLANs for granular traffic control. VLAN maps are used to match packets against ACLs and specify actions such as forwarding or dropping packets based on the match criteria.

VLAN maps do not use explicit permit or deny keywords; instead, the action is determined by the ACL and the map configuration.

Before you begin

Create the IPv6 ACL that you want to apply to the VLAN.

Follow these steps to configure a VLAN map:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password, if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **VLAN access-map** *name* [**number**] command to create a VLAN map.

Example:

```
Switch(config)# vlan access-map map_1 20
```

VLAN map can have a name or (optionally) a number. The number is the sequence number of the entry within the map.

When you create VLAN maps with the same name, numbers are assigned sequentially in increments of 10. When modifying or deleting maps, you can enter the number of the map entry that you want to modify or delete.

VLAN maps do not use the specific permit or deny keywords. To deny a packet by using VLAN maps, create an ACL that would match the packet, and set the action to drop. A permit in the ACL counts as a match. A deny in the ACL means no match.

Step 4 Use the **match** {**IP** | **ipv6** | **MAC**} **address** {*name* | *number*} [*name* | *number*] command to Matches the packet against one or more access lists.

Example:

```
Switch(config-access-map)# match ipv6 address ip_net
```

Note
Packets are only matched against access lists of the correct protocol type. IP packets are matched against IP access lists. Non-IP packets are only matched against named MAC access lists.

If the VLAN map is configured with a match clause for a type of packet (IP or MAC) and the map action is drop, all packets that match the type are dropped. If the VLAN map has no match clause, and the configured action is drop, all IP and Layer 2 packets are dropped.

Step 5 Use one of these commands to specify an IP packet or a non-IP packet (with only a known MAC address) and to match the packet against one or more ACLs:

- **action** { **forward** }

  ```
  Switch(config-access-map)# action forward
  ```

- **action** { **drop** }

  ```
  Switch(config-access-map)# action drop
  ```

Sets the action for the map entry.

Step 6 Use the **VLAN filter** *mapname* **VLAN-list** *list* command to apply the VLAN map to one or more VLAN IDs.

Example:

```
Switch(config)# vlan filter map 1 vlan-list 20-22
```

The list can be a single VLAN ID (22), a consecutive list (10-22), or a string of VLAN IDs (12, 22, 30). Spaces around the comma and hyphen are optional.

Step 7 Use the **end** command to exit IPv6 access list configuration mode and return to privileged EXEC mode.

Example:

```
Switch(config-ipv6-acl)# end
```

Apply a VLAN map to a VLAN

This task allows you to apply a VLAN map to one or more VLANs, enabling traffic filtering and policy enforcement on specified VLANs.

Use this task when you need to control or filter traffic on specific VLANs by associating a VLAN map with those VLAN IDs. This is typically performed during VLAN configuration or when updating network security policies.

Before you begin

Perform these steps to apply a VLAN map to a VLAN:

Procedure

Step 1 Use the **enable** command to enable privileged EXEC mode.

Example:

```
Switch> enable
```

Enter your password, if prompted.

Step 2 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 3 Use the **VLAN filter** *mapname* **VLAN-list** *list* command to apply the VLAN map to one or more VLAN IDs.

Example:

```
Switch(config)# vlan filter map 1 vlan-list 20-22
```

The **VLAN-list** can be a single VLAN ID (22), a consecutive list (10-22), or a string of VLAN IDs (12, 22, 30). Spaces around the comma and hyphen are optional.

Step 4 Use the **end** command to exit IPv6 access list configuration mode and return to privileged EXEC mode.

Example:

```
Switch(config-ipv6-acl)# end
```

IPv6 ACL monitoring commands

This reference lists the privileged EXEC commands used to display information about all configured access lists, all IPv6 access lists, or a specific access list.

Table 21: Show ACL commands

Command	Purpose
show access-lists	Displays all access lists configured on the switch.
show ipv6 access-list [*access-list-name*]	Displays all configured IPv6 access lists or the access list specified by name.
show VLAN access-map [*map-name*]	Displays VLAN access map configuration.
show VLAN filter [**access-map** *access-map* | **VLAN** *VLAN-id*]	Displays the mapping between VACLs and VLANs.