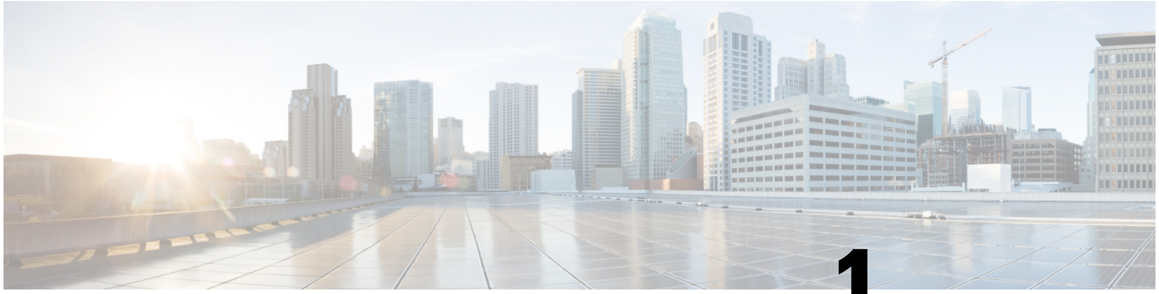
# Cisco cBR Series Converged Broadband Routers Layer 3 Configuration Guide for Cisco IOS XE Amsterdam 17.2.x

**First Published:** 2020-03-31

**Americas Headquarters**
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

# CONTENTS



CHAPTER 1

# DHCP, ToD, and TFTP Services for CMTS Routers

This document describes how to configure Cisco Cable Modem Termination System (CMTS) platforms so that they support onboard servers that provide Dynamic Host Configuration Protocol (DHCP), Time-of-Day (ToD), and Trivial File Transfer Protocol (TFTP) services for use in Data-over-Cable Service Interface Specification (DOCSIS) networks. In addition, this document provides information about optional configurations that can be used with external DHCP servers.

## Prerequisites for DHCP, ToD, and TFTP Services

To use the Cisco CMTS as the ToD server, either standalone or with other external ToD servers, you must configure the DHCP server to provide the IP address of the Cisco CMTS as one of the valid ToD servers (DHCP option 4) for cable modems.

## Restrictions for DHCP, ToD, and TFTP Services

- The ToD server must use the UDP protocol to conform to DOCSIS specifications.
- For proper operation of the DOCSIS network, especially a DOCSIS 1.1 network using BPI+ encryption and authentication, the system clock on the Cisco CMTS must be set accurately. You can achieve this by manually using the **set clock** command, or by configuring the CMTS to use either the Network Time Protocol (NTP) or the Simple Network Time Protocol (SNTP).
- Cisco cBR series routers do not support internal DHCP servers.

# Information About DHCP, ToD, and TFTP Services

This section provides the following information about the DHCP, ToD, and TFTP Services feature, and its individual components:

## Feature Overview

All Cisco CMTS platforms support onboard servers that provide DHCP, ToD, and TFTP proxy-services for use in DOCSIS cable networks. These servers provide the registration services needed by DOCSIS 1.0- and 1.1-compliant cable modems:

- External DHCP Servers—Provides DHCP services. External DHCP servers are usually part of an integrated provisioning system that is more suitable when managing large cable networks.
- Time-of-DayServer_—Provides an RFC 868 -compliant ToD service so that cable modems can obtain the current date and time during the registration process. The cable modem connects with the ToD server after it has obtained its IP address and other DHCP-provided IP parameters.

Although cable modems do not need to successfully complete the ToD request before coming online, this allows them to add accurate timestamps to their event logs so that these logs are coordinated to the clock used on the CMTS. In addition, having the accurate date and time is essential if the cable modem is trying to register with Baseline Privacy Interface Plus (BPI+) encryption and authentication.

- External TFTP_Server—Downloads the DOCSIS configuration file to the cable modem. The DOCSIS configuration file contains the operational parameters for the cable modem. The cable modem downloads its DOCSIS configuration file after connecting with the ToD server.

**Note** You can add additional servers in a number of ways. For example, most cable operators use Cisco Network Registrar (CNR) to provide the DHCP and TFTP servers. ToD servers are freely available for most workstations and PCs. You can install the additional servers on one workstation or PC or on different workstations and PCs.

## External DHCP Servers

The Cisco CMTS router provides the following optional configurations that can enhance the operation and security of external DHCP servers that you are using on the DOCSIS cable network:

### Cable Source Verify Feature

To combat theft-of-service attacks, you can enable the **cable source-verify** command on the cable interfaces on the Cisco CMTS router. This feature uses the router's internal database to verify the validity of the IP packets that the CMTS receives on the cable interfaces, and provides three levels of protection:

- At the most basic level of protection, the Cable Source Verify feature examines every IP upstream packet to prevent duplicate IP addresses from appearing on the cable network. If a conflict occurs, the Cisco CMTS recognizes only packets coming from the device that was assigned the IP address by the DHCP server. The devices with the duplicate addresses are not allowed network address. The CMTS also refuses to recognize traffic from devices with IP addresses that have network addresses that are unauthorized for that particular cable segment.

- Adding the **dhcp** option to the **cable source-verify** command provides a more comprehensive level of protection by preventing users from statically assigning currently-unused IP addresses to their devices. When the Cisco CMTS receives a packet with an unknown IP address on a cable interface, the CMTS drops the packet but also issues a DHCP LEASEQUERY message that queries the DHCP servers for any information about the IP and MAC addresses of that device. If the DHCP servers do not return any information about the device, the CMTS continues to block the network access for that device.
- When you use the **dhcp** option, you can also enable the **leasetimer** option, which instructs the Cisco CMTS to periodically check its internal CPE database for IP addresses whose lease times have expired. The CPE devices that are using expired IP addresses are denied further access to the network until they renew their IP addresses from a valid DHCP server. This can prevent users from taking DHCP-assigned IP addresses and assigning them as static addresses to their CPE devices.
- In addition to the dhcp option, you can also configure prefix-based source address verification (SAV) on the Cisco CMTS using the cable source-verify group command. A CM may have a static IPv4 or IPv6 prefix configured, which belongs to an SAV group. When the SAV prefix processing is enabled on the Cisco CMTS, the source IP address of the packets coming from the CM is matched against the configured prefix and SAV group (for that CM) for verification. If the verification fails, the packets are dropped, else the packets are forwarded for further processing. For more information on SAV prefix processing and SAV prefix configuration, see Prefix-based Source Address Verification , on page 3 and Configuring Prefix-based Source Address Verification, on page 14

### Prefix-based Source Address Verification

The Source Address Verification (SAV) feature verifies the source IP address of an upstream packet to ensure that the SID/MAC and IP are consistent. The DOCSIS 3.0 Security Specification introduces prefix-based SAV where every CM may have static IPv4 or IPv6 prefixes configured. These prefixes are either preconfigured on the CMTS, or are communicated to the CMTS during CM registration. The Cisco CMTS uses these configured prefixes to verify the source IP address of all the incoming packets from that CM.

An SAV group is a collection of prefixes. A prefix is an IPv4 or IPv6 subnet address. You can use the cable source-verify group command in global configuration mode to configure SAV groups. A total of 255 SAV groups are supported on a CMTS, with each SAV group having a maximum of four prefixes. Prefixes can be configured using the prefix command.

During registration, CMs communicate their configured static prefixes to the CMTS using two TLVs, 43.7.1 and 43.7.2. The TLV 43.7.1 specifies the SAV prefix group name that the CM belongs to, and TLV 43.7.2 specifies the actual IPv4 or IPv6 prefix. Each CM can have a maximum of four prefixes configured. When the Cisco CMTS receives these TLVs, it first identifies if the specified SAV group and the prefixes are already configured on the Cisco CMTS. If they are configured, the Cisco CMTS associates them to the registering CM. However if they are not configured, the Cisco CMTS automatically creates the specified SAV group and prefixes before associating them to the registering CM.

The SAV group name and the prefixes that are provided by these TLVs are considered valid by the Cisco CMTS. The packets received (from the CM) with the source IP address belonging to the prefix specified by the TLV are considered authorized. For example, if a given CM has been configured with an SAV prefix of 10.10.10.0/24, then any packet received from this CM (or CPE behind the CM) that is sourced with this address in the subnet 10.10.10.0/24 is considered to be authorized.

For more information on how to configure SAV groups and prefixes see Configuring Prefix-based Source Address Verification, on page 14.

## Smart Relay Feature

The Cisco CMTS supports a Smart Relay feature (the **ip dhcp smart-relay** command), which automatically switches a cable modem or CPE device to secondary DHCP servers or address pools if the primary server runs out of IP addresses or otherwise fails to respond with an IP address. The relay agent attempts to forward DHCP requests to the primary server three times. After three attempts with no successful response from the primary, the relay agent automatically switches to the secondary server.

When you are using the **cable dhcp-giaddr policy** command to specify that the CPE devices should use the secondary DHCP pools corresponding to the secondary addresses on a cable interface, the smart relay agent automatically rotates through the available secondary in a round robin fashion until an available pool of addresses is found. This ensures that clients are not locked out of the network because a particular pool has been exhausted.

## GIADDR Field

When using separate IP address pools for cable modems and CPE devices, you can use the **cable dhcp-giaddr policy** command to specify that cable modems should use an address from the primary pool and that CPE devices should use addresses from the secondary pool. The default is for the CMTS to send all DHCP requests to the primary DHCP server, while the secondary servers are used only if the primary server does not respond. The different DHCP servers are specified using the **cable helper** commands.

## DHCP Relay Agent Sub-option

The DHCP Relay Agent Information sub-option (DHCP Option 82, Suboption 9) enhancement simplifies provisioning of the CPE devices. Using this sub-option, the cable operators can relay the service class or QoS information of the CPE to the DHCP server to get an appropriate IP address.

To provision a CPE, the DHCP server should be made aware of the service class or QoS information of the CPE. The DHCP server obtains this information using the DHCP DISCOVER message, which includes the service class or QoS information of the CM behind which the CPE resides.

During the provisioning process, the Cisco CMTS uses the DHCPv4 Relay Agent Information sub-option to advertise information about the service class or QoS profile of the CMs to the DHCP server. Using the same technique, the CPE information is relayed to the DHCP server to get an appropriate IP address.

To enable the service classes option, the service class name specified in the CM configuration file must be configured on the Cisco CMTS. This is done by using the **cable dhcp-insert service-class** command.

**Note** To insert service class relay agent information option into the DHCP DISCOVER messages, the **ip dhcp relay information option-insert** command must be configured on the bundle interface.

# Time-of-Day Server

The Cisco CMTS can function as a ToD server that provides the current date and time to the cable modems and other customer premises equipment (CPE) devices connected to its cable interfaces. This allows the cable modems and CPE devices to accurately timestamp their Simple Network Management Protocol (SNMP) messages and error log entries, as well as ensure that all of the system clocks on the cable network are synchronized to the same system time.

The DOCSIS 1.0 and 1.1 specifications require that all DOCSIS cable modems request the following time-related fields in the DHCP request they send during their initial power-on provisioning:

- Time Offset (option 2)—Specifies the time zone for the cable modem or CPE device, in the form of the number of seconds that the device’s timestamp is offset from Greenwich Mean Time (GMT).

- Time Server Option (option 4)—Specifies one or more IP addresses for a ToD server.

After a cable modem successfully acquires a DHCP lease time, it then attempts to contact one of the ToD servers provided in the list provided by the DHCP server. If successful, the cable modem updates its system clock with the time offset and timestamp received from the ToD server.

If a ToD server cannot be reached or if it does not respond, the cable modem eventually times out, logs the failure with the CMTS, and continues on with the initialization process. The cable modem can come online without receiving a reply from a ToD server, but it must periodically continue to reach the ToD server at least once in every five-minute period until it successfully receives a ToD reply. Until it reaches a ToD server, the cable modem must initialize its system clock to midnight on January 1, 1970 GMT.

**Note** Initial versions of the DOCSIS 1.0 specification specified that the cable device must obtain a valid response from a ToD server before continuing with the initialization process. This requirement was removed in the released DOCSIS 1.0 specification and in the DOCSIS 1.1 specifications. Cable devices running older firmware that is compliant with the initial DOCSIS 1.0 specification, however, might require receiving a reply from a ToD server before being able to come online.

Because cable modems will repeatedly retry connecting with a ToD server until they receive a successful reply, you should consider activating the ToD server on the Cisco CMTS, even if you have one or more other ToD servers at the headend. This ensures that an online cable modem will always be able to connect with the ToD server on the Cisco CMTS, even if the other servers go down or are unreachable because of network congestion, and therefore will not send repeated ToD requests.

**Tip** To be able to use the Cisco CMTS as the ToD server, you must configure the DHCP server to provide the IP address Cisco CMTS as one of the valid ToD servers (DHCP option 4) for cable modems.

In addition, although the DOCSIS specifications do not require that a cable modem successfully obtain a response from a ToD server before coming online, not obtaining a timestamp could prevent the cable modem from coming online in the following situations:

- If DOCSIS configuration files are being timestamped, to prevent cable modems from caching the files and replaying them, the clocks on the cable modem and CMTS must be synchronized. Otherwise, the cable modem cannot determine whether a DOCSIS configuration file has the proper timestamp.
- If cable modems register using Baseline Privacy Interface Plus (BPI+) authentication and encryption, the clocks on the cable modem and CMTS must be synchronized. This is because BPI+ authorization requires that the CMTS and cable modem verify the timestamps on the digital certificates being used for authentication. If the timestamps on the CMTS and cable modem are not synchronized, the cable modem cannot come online using BPI+ encryption.

**Note** DOCSIS cable modems must use RFC 868 -compliant ToD server to obtain the current system time. They cannot use the Network Time Protocol (NTP) or Simple Network Time Protocol (SNTP) service for this purpose. However, the Cisco CMTS can use an NTP or SNTP server to set its own system clock, which can then be used by the ToD server. Otherwise, you must manually set the clock on the CMTS using the **clock set** command each time that the CMTS boots up.

**Tip** Additional servers can be provided by workstations or PCs installed at the cable headend. UNIX and Solaris systems typically include a ToD server as part of the operating system, which can be enabled by putting the appropriate line in the inetd.conf file. Windows systems can use shareware servers such as Greyware and Tardis. The DOCSIS specifications require that the ToD servers use the User Datagram Protocol (UDP) protocol instead of the TCP protocol for its packets.

# TFTP Server

All Cisco CMTS platforms can be configured to provide a TFTP server that can provide the following types of files to DOCSIS cable modems:

- DOCSIS Configuration File—After a DOCSIS cable modem has acquired a DHCP lease and attempted to contact a ToD server, the cable modem uses TFTP to download a DOCSIS configuration file from an authorized TFTP server. The DHCP server is responsible for providing the name of the DOCSIS configuration file and IP address of the TFTP server to the cable modem.
- Software Upgrade File—If the DOCSIS configuration file specifies that the cable modem must be running a specific version of software, and the cable modem is not already running that software, the cable modem must download that software file. For security, the cable operator can use different TFTP servers for downloading DOCSIS configuration files and for downloading new software files.
- Cisco IOS-XE Configuration File—The DOCSIS configuration file for Cisco cable devices can also specify that the cable modem should download a Cisco IOS-XE configuration file that contains command-line interface (CLI) configuration commands. Typically this is done to configure platform-specific features such as voice ports or IPSec encryption.

**Note** Do not confuse the DOCSIS configuration file with the Cisco IOS-XE configuration file. The DOCSIS configuration file is a binary file in the particular format that is specified by the DOCSIS specifications, and each DOCSIS cable modem must download a valid file before coming online. In contrast, the Cisco IOS-XE configuration file is an ASCII text file that contains one or more Cisco IOS-XE CLI configuration commands. Only Cisco cable devices can download a Cisco IOS-XE file.

All Cisco CMTS platforms can be configured as TFTP servers that can upload these files to the cable modem. The files can reside on any valid device but typically should be copied to the Flash memory device inserted into the Flash disk slot on the Cisco CMTS.

# Sniff out boot file name from DHCP process per CM

Starting from Cisco IOS XE Gibraltar 16.12.1, the cBR-8 can sniff, parse and save the configuration file related information per CM. This includes path and name of the configuration file, the IPv4 or IPv6 address of the TPFT server from which the CM requests the configuration file, and the timestamp when cBR-8 sniffed the CM’s TFTP RRQ packet that requests the configuration file. This feature is enabled by default and cannot be disabled.

The **show cable modem tftp** command displays a single CM’s configuration file related information.

The following example shows the sample output for this command.

```
Router#show cable modem 34bd.fa0f.4418 tftp
Host Interface : C1/0/0
```

```
MAC Address : 34bd.fa0f.4418
IP Address : 50.13.0.4
IPv6 Address : 2001:50:13:0:74E3:4197:E2F2:8162
Modem Status : w-online(pt)
TFTP Server Address : 2001:1:38::25:3
Modem Configuration File Name : cbr8/cm.bin
Timestamp : 02:16:02 CST Tue May 21 2019
```

# Benefits

- The Cisco CMTS can act as a primary or backup ToD server to ensure that all cable modems are synchronized with the proper date and time before coming online. This also enables cable modems to come online more quickly because they will not have to wait for the ToD timeout period before coming online.
- The ToD server on the Cisco CMTS ensures that all devices connected to the cable network are using the same system clock, making it easier for you to troubleshoot system problems when you analyze the debugging output and error logs generated by many cable modems, CPE devices, the Cisco CMTS, and other services.
- The Cisco CMTS can act as a TFTP server for DOCSIS configuration files, software upgrade files, and Cisco IOS configuration files.

# How to Configure ToD, and TFTP Services

See the following configuration tasks required to configure time-of-day service, and TFTP service on a Cisco CMTS:

## Configuring Time-of-Day Service

This section provides procedures for enabling and disabling the time-of-day (ToD) server on the Cisco CMTS routers.

### Prerequisites

To be able to use the Cisco CMTS as the ToD server you must configure the DHCP server to provide the IP address Cisco CMTS as one of the valid ToD servers (DHCP option 4) for cable modems.

### Enabling Time-of-Day Service

To enable the ToD server on a Cisco CMTS, use the following procedure, beginning in EXEC mode.

#### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable`<br>`Router#` | Enables privileged EXEC mode. Enter your password if prompted. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal`<br>`Router(config)#` | Enters global configuration mode. |
| **Step 3** | **service udp-small-servers max-servers no-limit**<br>**Example:**<br>`Router(config)# service udp-small-servers max-servers no-limit`<br>`Router(config)#` | Enables use of minor servers that use the UDP protocol (such as ToD, echo, chargen, and discard).<br>The max-servers no-limit option allows a large number of cable modems to obtain the ToD server at one time, in the event that a cable or power failure forces many cable modems offline. When the problem has been resolved, the cable modems can quickly reconnect. |
| **Step 4** | **cable time-server**<br>**Example:**<br>`Router(config)# cable time-server`<br>`Router(config)#` | Enables the ToD server on the Cisco CMTS. |
| **Step 5** | **exit**<br>**Example:**<br>`Router(config)# exit`<br>`Router#` | Exits global configuration mode. |

## Disabling Time-of-Day Service

To disable the ToD server, use the following procedure, beginning in EXEC mode.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable`<br>`Router#` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal`<br>` Router(config)#` | Enters global configuration mode. |
| **Step 3** | **no cable time-server**<br>**Example:** | Disables the ToD server on the Cisco CMTS. |

| | Command or Action | Purpose |
|---|---|---|
| | `Router(config)# cable time-server`<br>`Router(config)#` | |
| **Step 4** | **no service udp-small-servers**<br>**Example:**<br>`Router(config)# no service udp-small-servers`<br>`Router(config)#` | (Optional) Disables the use of all minor UDP servers.<br>**Note** Do not disable the minor UDP servers if you are also enabling the other DHCP or TFTP servers. |
| **Step 5** | **exit**<br>**Example:**<br>`Router(config)# exit`<br>`Router#` | Exits global configuration mode. |

# Configuring TFTP Service

To configure TFTP service on a Cisco CMTS where the CMTS can act as a TFTP server and download a DOCSIS configuration file to cable modems, perform the following steps:

- Create the DOCSIS configuration files using the DOCSIS configuration editor of your choice.
- Copy all desired files (DOCSIS configuration files, software upgrade files, and Cisco IOS configuration files) to the Flash memory device on the Cisco CMTS. Typically, this is done by placing the files first on an external TFTP server, and then using TFTP commands to transfer them to the router's Flash memory.
- Enable the TFTP server on the Cisco CMTS with the **tftp-server** command.

Each configuration task is required unless otherwise listed as optional.

**Step 1** Use the **show file systems** command to display the Flash memory cards that are available on your CMTS, along with the free space on each card and the appropriate device names to use to access each card.

Most configurations of the Cisco CMTS platforms support both linear Flash and Flash disk memory cards. Linear Flash memory is accessed using the **slot0** (or **flash**) and **slot1** device names. Flash disk memory is accessed using the **disk0** and **disk1** device names.

For example, the following command shows a Cisco uBR7200 series router that has two linear Flash memory cards installed. The cards can be accessed by the **slot0** (or **flash**) and **slot1** device names.

**Example:**

```
Router# show file systems

File Systems:
     Size(b)     Free(b)      Type  Flags  Prefixes
    48755200    48747008     flash     rw   slot0: flash:
    16384000    14284000     flash     rw   slot1:
    32768000    31232884     flash     rw   bootflash:
*          -           -      disk     rw   disk0:
           -           -      disk     rw   disk1:
```

```
           -           -    opaque     rw   system:
           -           -    opaque     rw   null:
           -           -   network     rw   tftp:
      522232      507263     nvram     rw   nvram:
           -           -   network     rw   rcp:
           -           -   network     rw   ftp:
           -           -   network     rw   scp:
Router#
```

The following example shows a Cisco uBR10012 router that has two Flash disk cards installed. These cards can be accessed by the **disk0** and **sec-disk0** device names.

**Example:**

```
Router# show file systems

File Systems:
     Size(b)     Free(b)      Type  Flags  Prefixes
           -           -     flash     rw   slot0: flash:
           -           -     flash     rw   slot1:
    32768000    29630876     flash     rw   bootflash:
*  128094208    95346688      disk     rw   disk0:
           -           -      disk     rw   disk1:
           -           -    opaque     rw   system:
           -           -     flash     rw   sec-slot0:
           -           -     flash     rw   sec-slot1:
*  128094208    95346688      disk     rw   sec-disk0:
           -           -      disk     rw   sec-disk1:
    32768000    29630876     flash     rw   sec-bootflash:
           -           -     nvram     rw   sec-nvram:
           -           -    opaque     rw   null:
           -           -   network     rw   tftp:
      522232      505523     nvram     rw   nvram:
           -           -   network     rw   rcp:
           -           -   network     rw   ftp:
           -           -   network     rw   scp:
Router#
```

**Step 2** Verify that the desired Flash memory card has sufficient free space for all of the files that you want to copy to the CMTS.

**Step 3** Use the **ping** command to verify that the remote TFTP server that contains the desired files is reachable. For example, the following shows a **ping** command being given to an external TFTP server with the IP address of 10.10.10.1:

**Example:**

```
Router# ping 10.10.10.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 5/6/6 ms
```

**Step 4** Use the **copy tftp** *devname* command to copy each file from the external TFTP server to the appropriate Flash memory card on the CMTS, where *devname* is the device name for the destination Flash memory card. You will then be prompted for the IP address for the external TFTP server and the filename for the file to be transferred.

The following example shows the file docsis.cm being transferred from the external TFTP server at IP address 10.10.10.1 to the first Flash memory disk (disk0):

**Example:**

```
Router# copy tftp disk0
Address or name of remote host []? 10.10.10.1
```

```
Source filename []? config-files/docsis.cm

Destination filename [docsis.cm]?
Accessing tftp://10.10.10.1/config-file/docsis.cm......
Loading docsis.cm from 10.10.10.1 (via Ethernet2/0): !!!
[OK - 276/4096 bytes]
276 bytes copied in 0.152 secs
Router#
```

**Step 5** Repeat Step 4, on page 10 as needed to copy all of the files from the external TFTP server to the Flash memory card on the Cisco CMTS.

**Step 6** Use the **dir** command to verify that the Flash memory card contains all of the transferred files.

**Example:**

```
Router# dir disk0:

Directory of disk0:/
    1  -rw-    10705784   May 30 2002 19:12:46  ubr10k-p6-mz.122-2.8.BC
    2  -rw-        4772   Jun 20 2002 18:12:56  running.cfg.save
    3  -rw-         241   Jul 31 2002 18:25:46  gold.cm
    4  -rw-         225   Jul 31 2002 18:25:46  silver.cm
    5  -rw-         231   Jul 31 2002 18:25:46  bronze.cm
    6  -rw-          74   Oct 11 2002 21:41:14  disable.cm
    7  -rw-     2934028   May 30 2002 11:22:12  ubr924-k8y5-mz.bin
    8  -rw-     3255196   Jun 28 2002 13:53:14  ubr925-k9v9y5-mz.bin
128094208 bytes total (114346688 bytes free)
Router#
```

**Step 7** Use the **configure terminal** command to enter global configuration mode:

**Example:**

```
Router# configure terminal

Router(config)#
```

**Step 8** Use the **tftp-server** command to specify which particular files can be transferred by the TFTP server that is onboard the Cisco CMTS. You can also use the **alias** option to specify a different filename that the DHCP server can use to refer to the file. For example, the following commands enable the TFTP transfer of the configuration files and software upgrade files:

**Example:**

```
Router(config)# tftp-server disk0:gold.cm alias gold.cm

Router(config)# tftp-server disk0:silver.cm alias silver.cm

Router(config)# tftp-server disk0:bronze.cm alias bronze.cm

Router(config)# tftp-server disk0:ubr924-k8y5-mz.bin alias ubr924-codefile

Router(config)# tftp-server disk0:ubr925-k9v9y5-mz.bin alias ubr925-codefile

Router(config)#
```

**Note** The **tftp-server** command also supports the option of specifying an access list that restricts access to the particular file to the IP addresses that match the access list.

**Step 9** (Optional) Use the following command to enable the use of the UDP small servers, and to allow an unlimited number of connections at one time. This will allow a large number of cable modems that have gone offline due to cable or power failure to rapidly come back online.

**Example:**

```
Router(config)# service udp-small-servers max-servers no-limit

Router(config)#
```

# Optimizing the Use of an External DHCP Server

The Cisco CMTS offers a number of options that can optimize the operation of external DHCP servers on a DOCSIS cable network. See the following sections for details. All procedures are optional, depending on the needs of your network and application servers.

## Configuring Cable Source Verify Option

To enhance security when using external DHCP servers, you can optionally configure the Cable Source Verify feature with the following procedure.

**Restriction**

- The Cable Source Verify feature supports only external DHCP servers. It cannot be used with the internal DHCP server.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable`<br>`Router#` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal`<br>`Router(config)#` | Enters global configuration mode. |
| **Step 3** | **interface cable** *x/y*<br>**Example:**<br>`Router(config)# interface cable 4/0`<br>`Router(config-if)#` | Enters cable interface configuration mode for the specified cable interface. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 4** | **cable source-verify** [**dhcp** \| **leasetimer** *value* ]<br>**Example:**<br>`Router(config-if)# cable source-verify dhcp`<br>**Example:**<br>`Router(config-if)# cable source-verify leasetimer 30`<br>`Router(config-if)#` | (Optional) Ensures that the CMTS allows network access only to those IP addresses that DCHP servers issued to devices on this cable interface. The CMTS examines DHCP packets that pass through the cable interfaces to build a database of which IP addresses are valid on which interface.<br>• **dhcp** = (Optional) Drops traffic from all devices with unknown IP addresses, but the CMTS also sends a query to the DHCP servers for any information about the device. If a DHCP server informs the CMTS that the device has a valid IP address, the CMTS then allows the device on the network.<br>• **leasetimer** *value* = (Optional) Specifies how often, in minutes, the router should check its internal CPE database for IP addresses whose lease times have expired. This can prevent users from taking DHCP-assigned IP addresses and assigning them as static addresses to their CPE devices. The valid range for value is 1 to 240 minutes, with no default.<br>**Note** The **leasetimer** option takes effect only when the **dhcp** option is also used on an interface. |
| **Step 5** | **no cable arp**<br>**Example:**<br>`Router(config-if)# no cable arp`<br>`Router(config-if)#` | (Optional) Blocks Address Resolution Protocol (ARP) requests originating from devices on the cable network. Use this command, together with the **cable source-verify dhcp** command, to block certain types of theft-of-service attacks that attempt to hijack or spoof IP addresses.<br>**Note** Repeat Step 3, on page 12 through Step 5, on page 13 for each desired cable interface. |
| **Step 6** | **exit**<br>**Example:**<br>`Router(config-if)# exit`<br>`Router(config)#` | Exits interface configuration mode. |
| **Step 7** | **ip dhcp relay information option**<br>**Example:**<br>`Router(config)# ip dhcp relay information option`<br>`Router(config)#` | (Optional) Enables the CMTS to insert DHCP relay information (DHCP option 82) in relayed DHCP packets. This allows the DHCP server to store accurate information about which CPE devices are using which cable modems. You should use this command if you are also using the **cable source-verify dhcp** command. |
| **Step 8** | **exit**<br>**Example:**<br>`Router(config)# exit`<br>`Router#` | Exits global configuration mode. |

## Configuring Prefix-based Source Address Verification

To enhance security when using external DHCP servers, you can configure a prefix-based SAV with the following procedure, beginning in global configuration (config) mode.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable`<br>`Router#` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal`<br>`Router(config)#` | Enters global configuration mode. |
| **Step 3** | cable source-verify enable-sav-static<br>**Example:**<br>`Router# cable source-verify enable-sav-static`<br>`Router(config)#` | Enables SAV prefix processing on the Cisco CMTS. |
| **Step 4** | **cable source-verify group** *groupname*<br>**Example:**<br>`Router(config)# cable source-verify group sav-1` | Configures the SAV group name.<br>*groupname*— Name of the SAV group with a maximum length of 16 characters. |
| **Step 5** | **prefix** [**ipv4_prefix/ipv4_prefix_length** \| **ipv6_prefix/ipv6_prefix_length** ]<br>**Example:**<br>`Router(config-sav)# prefix 10.10.10.0/24`<br>`Router(config-sav)#` | Configures the IPv4 or IPv6 prefix associated with the SAV group.<br>• *ipv4_prefix*— IPv4 prefix associated with the SAV group, specified in the X.X.X.X/X format.<br>• ipv4_prefix_length—Length of the IPv4 prefix. The valid range is from 0 to 32.<br>• ipv6_prefix—IPv6 prefix associated with a particular SAV group, specified in the X:X:X:X::/X format.<br>• ipv6_prefix_length—Length of the IPv6 prefix. The valid range is from 0 to 128.<br>A maximum of four prefixes can be configured in a single SAV group. These prefixes can be either IPv4s, IPv6s, or a combination of both. |
| **Step 6** | **exit**<br>**Example:**<br>`Router(config-sav)# exit` | Exits SAV configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 7** | **exit**<br>**Example:**<br>`Router(config)# exit` | Exits global configuration mode. |

## Configuring Optional DHCP Parameters

When using an external DHCP server, the Cisco CMTS supports a number of options that can enhance operation of the cable network in certain applications. To configure these options, use the following procedure, beginning in EXEC mode.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable`<br>`Router#` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal`<br>`Router(config)#` | Enters global configuration mode. |
| **Step 3** | **ip dhcp smart-relay**<br>**Example:**<br>`Router(config)# ip dhcp smart-relay`<br>`Router(config)#` | (Optional) Enables the DHCP relay agent on the CMTS to automatically switch a cable modem or CPE device to a secondary DHCP server or address pool if the primary DHCP server does not respond to three successive requests. If multiple secondary servers have been defined, the relay agent forwards DHCP requests to the secondary servers in a round robin fashion. |
| **Step 4** | **ip dhcp ping packet 0**<br>**Example:**<br>`Router(config)# ip dhcp ping packet 0`<br>`Router(config)#` | (Optional) Instructs the DHCP server to assign an IP address from its pool without first sending an ICMP ping to test whether a client is already currently using that IP address. Disabling the ping option can speed up address assignment when a large number of modems are trying to connect at the same time. However, disabling the ping option can also result in duplicate IP addresses being assigned if users assign unauthorized static IP addresses to their CPE devices. |

| | Command or Action | Purpose |
|---|---|---|
| | | **Note** By default, the DHCP server pings a pool address twice before assigning a particular address to a requesting client. If the ping is unanswered, the DHCP server assumes that the address is not in use and assigns the address to the requesting client. |
| **Step 5** | **ip dhcp relay information check**<br>**Example:**<br>`Router(config)# ip dhcp relay information check`<br>`Router(config)#` | (Optional) Configures the DHCP server to validate the relay agent information option in forwarded BOOTREPLY messages. Invalid messages are dropped.<br>**Note** The **ip dhcp relay information** command contains several other options that might be useful for special handling of DHCP packets. See its command reference page in the Cisco IOS-XE documentation for details. |
| **Step 6** | **interface cable** *x/y*<br>**Example:**<br>`Router(config)# interface cable 4/0`<br>`Router(config-if)#` | Enters cable interface configuration mode for the specified cable interface. |
| **Step 7** | **cable dhcp-giaddr policy** [**host** \| **stb** \| **mta** \| **ps**\| **profile** *name*] *giaddr*<br>**Example:**<br>`Router(config-if)# cable dhcp-giaddr policy mta 172.1.1.10`<br>`Router(config-if)#` | Sets the DHCP GIADDR field for DHCP request packets to the primary address for cable modems, and the secondary address for CPE devices. This enables the use of separate address pools for different clients.<br>• **host**—Specifies the GIADDR for hosts.<br>• **mta**—Specifies the GIADDR for MTAs.<br>• **ps**—Specifies the GIADDR for PSs.<br>• **stb**—Specifies the GIADDR for STBs.<br>• **profile** *name*Specifies DHCP profile as control policy.<br>• *giaddr*—IP addresses of the secondary interface of the bundle interface.<br>**Note** The **cable dhcp-giaddr** command also supports the **primary** option. The **primary** option forces all device types to use only the primary interface IP address as GIADDR and not rotate through the secondary address if the primary address fails. |
| **Step 8** | **cable helper-address** *address* [**cable-modem** \| **host** \| **mta** \| **stb**\| **profile** *name*]<br>**Example:**<br>`Router(config-if)# cable helper-address` | (Optional) Enables load-balancing of DHCP requests from cable modems and CPE devices by specifying different DHCP servers according to the cable interface or subinterface. You can also specify separate servers for cable modems and CPE devices. |

| | Command or Action | Purpose |
|---|---|---|
| | `10.10.10.13`<br>`Router(config-if)#` | • *address* = IP address of a DHCP server to which UDP broadcast packets will be sent via unicast packets.<br>• **cable-modem** = Specifies this server should only accept cable modem packets (optional).<br>• **host** = Specifies this server should only accept CPE device packets (optional).<br>• **mta**—(Optional) Specifies this server should only accept MTA packets .<br>• **stb** —(Optional) Specifies this server should only accept STB packets .<br>• **profile** *name*—(Optional) Specifies that only UDP broadcasts with specific DHCP profile are forwarded.<br>**Note** If you do not specify an option, the helper-address will support all cable devices, and the associated DHCP server will accept DHCP packets from all cable device classes.<br>**Note** If you specify only one option, the other types of devices (cable modem, host, mta, or stb) will not be able to connect with a DHCP server. You must specify each desired option in a separate command<br>.<br>**Tip** Repeat this command to specify more than one helper address on each cable interface. You can specify more than 16 helper addresses, but the Cisco IOS software uses only the first 16 valid addresses.<br>**Tip** If you configure different helper addresses on different sub-bundles within a bundle, the cable modem may not come online. We recommend that you use the same helper address on all sub-bundles within a bundle.<br>**Note** The **ip helper-address** command performs a similar function to **cable helper-address**, but it should be used on non-cable interfaces. The **cable helper-address** command should be used on cable interfaces because it is optimized for the operation of DHCP requests on DOCSIS networks. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 9** | **cable dhcp-giaddr policy**<br>**Example:**<br>`Router(config-if)# cable dhcp-giaddr policy` | Selects the control policy, so the primary address is used for cable modems and the secondary addresses are used for hosts and other customer premises equipment (CPE) devices. This setting is typically used when the CMs on the interface are configured for routing mode, so that the cabel modems and hosts can use IP addresses on different subnets. |
| **Step 10** | **exit**<br>**Example:**<br>`Router(config-if)# exit`<br>`Router(config)#` | Exits interface configuration mode. |
| **Step 11** | **exit**<br>**Example:**<br>`Router(config)# exit`<br>`Router#` | Exits global configuration mode. |

# How to Configure ToD, and TFTP Services

See the following configuration tasks required to configure time-of-day service, and TFTP service on a Cisco CMTS:

# Configuration Examples

This section provides examples for the following configurations:

## ToD Server Example

The following example shows a typical ToD server configuration:

```
service udp-small-servers max-servers no-limit
cable time-server
```

These are the only commands required to enable the ToD server.

## TFTP Server Example

The following lines are an excerpt from a configuration that includes a TFTP server. Change the files listed with the **tftp-server** command to match the specific files that are on your system.

```
! Enable the user of unlimited small servers
 service udp-small-servers max-servers no-limit
!
```

```
...
! Enable the TFTP server and specify the files that can be
!   downloaded along with their aliases
 tftp-server disk0:gold.cm alias gold.cm
 tftp-server disk0:silver.cm alias silver.cm
 tftp-server disk0:bronze.cm alias bronze.cm
 tftp-server disk0:ubr924-k8y5-mz.bin alias ubr924-codefile
 tftp-server disk0:ubr925-k9v9y5-mz.bin alias ubr925-codefile
```

# Additional References

### 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.<br>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.<br>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password. | http://www.cisco.com/support |

# Feature Information for the DHCP, ToD, and TFTP Services for the CMTS Routers

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 1: Feature Information for Downstream Interface Configuration*

| Feature Name | Releases | Feature Information |
|---|---|---|
| DHCP, ToD, and TFTP services | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on the Cisco cBR Series Converged Broadband Routers. |

| Feature Name | Releases | Feature Information |
|---|---|---|
| Sniff out boot file name from DHCP process per CM | Cisco IOS XE Gibraltar 16.12.1 | This feature was supported on the Cisco cBR Series Converged Broadband Routers. |

CHAPTER 2

# Virtual Interface Bundling

Virtual Interface Bundling allows supports combining multiple cable interfaces in a Cisco cBR series router into a single logical bundle, so as to conserve IP address space and simplify network management.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 2: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information About Virtual Interface Bundling

This section contains the following:

## Overview of Virtual Interface Bundling

**Note** All cable bundles are automatically converted and configured to virtual interface bundles. Any standalone cable interfaces must be manually configured to be in a virtual bundle to operate properly.

Virtual interface bundling supports the following:

- Virtual interface bundling uses *bundle interface* and *bundle members* instead of primary or secondary interfaces.
- A virtual bundle interface is virtually defined, as with IP loopback addresses.
- Virtual interface bundling supports bundle information in multiple **show** commands.
- The CISCO-DOCS-EXT-MIB is updated for cable helper-address and IPv6 DHCP relay configurations.

Virtual interface bundling prevents loss of connectivity on physical interfaces should there be a failure, problematic online insertion and removal (OIR) of one line card in the bundle, or erroneous removal of configuration on the primary interface.

Virtual interface bundling supports and governs the following Layer 3 settings for the bundle member interfaces:

- IP address
- IP helper-address
- source-verify and lease-timer functions
- cable dhcp-giaddr (The giaddr field is set to the IP address of the DHCP client.)
- Protocol Independent Multicast (PIM)
- Access control lists (ACLs)
- Sub-interfaces
- IPv6
- 1982 bytes layer 3 MTU.

**Note** In case customer wants to test 1982 bytes MTU by issuing a ping from CMTS to DOCSIS 3.1 modem, **cable mtu-override** command needs to be configured. After the test, please remove this configuration using **no cable mtu-override** command. By default, there is no cable mtu-override configured in bundle interface.

**Note** This virtual interface for the bundle should always remain on (enabled with **no shutdown**).

# Guidelines for Virtual Interface Bundling

The following guidelines describe virtual interface bundling:

- Initial configuration of the first virtual bundle *member* automatically creates a virtual bundle interface.
- All cable bundles are automatically converted and configured to be in a virtual bundle after loading the software image.

- Standalone cable interfaces must be manually configured to be in a virtual bundle to operate properly.
- The virtual bundle interface accumulates the counters from members; counters on member links are not cleared when they are added to the bundle. If a bundle-only counter is desired, clear the bundle counter on the members before adding them to the bundle, or before loading the image.
- This feature supports a maximum of 40 virtual interface bundles, with the numeric range from 1 to 255.
- The virtual bundle interface remains configured unless specifically deleted, even if all members in the bundle are deleted.
- This feature supports subinterfaces on the virtual bundle interface.
- *Bundle-aware* configurations are supported on the virtual bundle interface.
- *Bundle-unaware* configurations are supported on each bundle member.
- While creating the virtual bundle interface, if the bundle interface existed in earlier Cisco IOS releases, then the earlier cable configurations re-appear after upgrade.
- When using sub-bundle, all layer 3 configurations must be configured on sub-bundle, instead of main bundle.

# Virtual Interface Bundle-aware and Bundle-unaware Support

Virtual interface bundling uses two configurations: the virtual *bundle* itself, and the interfaces in that virtual bundle, known as *bundle members* . The virtual interface bundle and bundle members are either aware of the bundle, or unaware of the bundle, as follows.

- Bundle-aware features are maintained on the virtual *bundle* . These include:
  - IP Address
  - IP helper, cable helper
  - Dhcp-giaddr
  - Sub-interface
  - Source verify
  - Lease-query
  - Address Resolution Protocol (Cable ARP filtering, which also bundles cable interfaces, and Proxy ARP)
  - Cable match
  - Access Control Lists (ACLs)
  - Protocol Independent Multicast (PIM)
  - Cable Intercept
- Bundle-unaware features are maintained on the *bundle members* . These include:
  - DS/US configurations
  - HCCP redundancy
  - Load balancing
  - DMIC, tftp-enforce, shared-secret
  - Spectrum management
  - Admission control
  - Intercept

# Configuring Virtual Interface Bundling

To enable virtual interface bundling, and to reconfigure interface information on the Cisco CMTS as required, you first configure the virtual interface bundle, then add additional bundle members for the specified virtual bundle. Perform these steps on each interface, as needed for all virtual interface bundles.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface bundle** *n*<br>**Example:**<br>`Router(config-if)# interface bundle 1` | Adds the selected interface to the virtual bundle. If this is the first interface on which the virtual bundle is configured, this command enables the bundle on the specified interface.<br>As many as 40 virtual interface bundles can be configured on the Cisco CMTS. Numeric identifiers may range from 1 to 255. |
| **Step 4** | **ip address** *address mask*<br>**Example:**<br>`Router(config-if)# ip address 7.7.7.7 255.255.255.0` | Use as needed after Cisco IOS upgrade.<br>Configures the IP address for the specified interface and virtual bundle. |
| **Step 5** | **cable helper-address address** [**cable-modem** \| **host** \| **mta** \| **ps** \| **stb**]<br>**Example:**<br>`Router(config-if)# cable helper-address 10.10.10.13` | (Optional) Specifies the IPv4 DHCP server address. |
| **Step 6** | **cable dhcp-giaddr** {**primary** \| **policy** [**host** \| **stb** \| **mta** \| **ps** \| **strict**]}<br>**Example:**<br>`Router(config-if)# cable dhcp-giaddr policy host` | Sets the DHCP GIADDR field for DHCP request packets. |
| **Step 7** | **cable source-verify dhcp**<br>**Example:**<br>`Router(config-if)# cable source-verify dhcp` | (Optional) Ensures that the Cisco CMTS allows network access only to those IP addresses that DCHP servers issued to devices on this cable interface. The Cisco CMTS examines the DHCP packets that pass through the cable |

| | Command or Action | Purpose |
|---|---|---|
| | | interfaces to build a database of which IP addresses are valid on which interface. Drops traffic from all devices with unknown IP addresses, but the Cisco CMTS also sends a query to the DHCP servers for any information about the device. If a DHCP server informs the Cisco CMTS that the device has a valid IP address, the CMTS then allows the device on the network. |
| **Step 8** | **no cable arp**<br>**Example:**<br>`Router(config-if)# no cable arp` | (Optional) Blocks the static IPv4 CPE from coming online. Also blocks Address Resolution Protocol (ARP) process destined to devices on the cable network.<br>**Note** Use this command, together with the **cable source-verify dhcp** command, to block certain types of scanning attacks that attempt to cause denial of service (DoS) on the Cisco CMTS. |
| **Step 9** | **exit**<br>**Example:**<br>`Router(config-if)# exit` | Exits the interface configuration mode and enters global configuration mode. |
| **Step 10** | **interface cable** *slot /subslot/port*<br>**Example:**<br>`Router(config)# interface cable 3/0/0` | Enters interface configuration mode for the selected interface, on which virtual interface bundling is to be enabled. |
| **Step 11** | **cable bundle** *n*<br>**Example:**<br>`Router(config-if)# cable bundle 1` | Configures a cable interface to belong to an interface bundle, where *n* is the bundle number. |
| **Step 12** | **no cable upstream** *n* **shut**<br>**Example:**<br>`Router(config-if)# no cable upstream 4 shut` | Use as needed after Cisco IOS upgrade.<br>The cable interface must be enabled using the no shutdown command for the specified cable interface.<br>*n* —Specifies the cable interface to enable for the virtual bundle. |
| **Step 13** | **end**<br>**Example:**<br>`Router(config-if)# end` | Returns to privileged EXEC mode. |

### What to do next

To remove a virtual bundle from the interface, use the **no interface bundle** command in interface configuration mode, where *n* specifies the bundle identifier:

**no interface bundle** *n*

If you remove a member from a bundle, the bundle remains on the interface (even if empty) until the bundle itself is specifically removed.

For more information on configuring IPv6 parameters for bundle interface, see *IPv6 on Cable* feature guide.

# Verfiying the Virtual Interface Bundling Configuration

- **show ip interface brief**—Displays the summary of interfaces.

  Following is a sample output of this command:

  ```
  Router# show ip interface brief

  Interface              IP-Address      OK? Method Status                Protocol
  Cable3/0/0             Bundle1         YES unset  up                    up
  GigabitEthernet0       10.86.3.175     YES NVRAM  administratively down down
  Bundle1                100.1.2.1       YES manual up                    up
  Bundle2                100.1.3.1       YES NVRAM  up                    up
  Dti4/1/0               unassigned      YES unset  administratively down down
  Dti5/1/0               unassigned      YES unset  administratively down down
  Dti4/1/1               unassigned      YES unset  administratively down down
  Dti5/1/1               unassigned      YES unset  administratively down down
  Loopback1              1.2.3.4         YES NVRAM  up                    up
  Tunnel0                unassigned      YES unset  up                    up
  ```

- **show running-config interface bundle** *n*—Displays the information about the specified bundle.

  Following is a sample output of this command:

  ```
  Router# show running-config interface Bundle 1

  Current configuration : 696 bytes
  !
  interface Bundle2
   ip address 100.1.3.1 255.255.255.0
   no cable nd
   cable arp filter request-send 3 2
   cable arp filter reply-accept 3 2
   no cable arp
   cable ipv6 source-verify dhcp
   cable source-verify dhcp
   cable dhcp-giaddr primary
   cable helper-address 10.10.0.53
   ipv6 address 2001:420:3800:910::1/64
   ipv6 enable
   ipv6 nd reachable-time 3600000
   ipv6 nd cache expire 65536
   ipv6 nd managed-config-flag
   ipv6 nd other-config-flag
   ipv6 nd ra interval msec 2000
   no ipv6 redirects
   ipv6 dhcp relay destination 2001:420:3800:800:250:56FF:FEB2:F11D
   ipv6 dhcp relay destination vrf vrfa 2001:420:3800:800:250:56FF:FEB2:F11D
   ipv6 dhcp relay source-interface Bundle2
   arp timeout 2147483
  ```

- **show ip interface brief | include bundle**—Displays the bundle interface information.

  Following is a sample output of this command:

```
Router# show ip interface brief | include Bundle

Bundle1 unassigned YES unset up up
Bundle1.1 100.1.2.1 YES NVRAM up up
Bundle2 100.1.3.1 YES NVRAM up up
```

- **show running-config interface bundle** *n.n*—Displays the subinterface information for the specified bundle.

  Following is a sample output of this command:

```
Router# show running-config interface bundle 1.1

Current configuration : 1415 bytes
!
interface Bundle1.1
ip address 100.1.2.1 255.255.255.0
ip pim sparse-mode
ip rip send version 2
ip rip receive version 2
ip rip authentication mode md5
ip rip authentication key-chain ubr-rip
ip igmp static-group 239.1.4.1 source 115.255.0.100
ip igmp static-group 239.1.3.1 source 115.255.0.100
ip igmp static-group 239.1.2.1 source 115.255.0.100
ip igmp static-group 232.1.4.1 source 115.255.0.100
ip igmp static-group 232.1.3.1 source 115.255.0.100
ip igmp static-group 232.1.2.1 source 115.255.0.100
ip igmp static-group 232.1.1.1 source 115.255.0.100
ip igmp static-group 230.1.4.1
ip igmp static-group 230.1.3.1
ip igmp static-group 230.1.2.1
ip igmp static-group 224.1.4.1
ip igmp static-group 224.1.3.1
ip igmp static-group 224.1.2.1
ip igmp static-group 224.1.1.1
ip igmp version 3
ip igmp query-interval 20
no cable arp
cable ipv6 source-verify dhcp
cable source-verify dhcp
cable dhcp-giaddr primary
cable helper-address 10.10.0.53
ipv6 address 2001:420:3800:909::1/64
ipv6 enable
ipv6 nd reachable-time 3600000
ipv6 nd cache expire 65536
ipv6 nd prefix default no-advertise
ipv6 nd managed-config-flag
ipv6 nd other-config-flag
ipv6 nd ra interval msec 2000
no ipv6 redirects
ipv6 dhcp relay destination 2001:420:3800:800:250:56FF:FEB2:F11D link-address
2001:420:3800:909::1
ipv6 dhcp relay source-interface Bundle1
ipv6 rip CST enable
arp timeout 2147483
```

# Additional References

### Related Documents

| Related Topic | Document Title |
|---|---|
| CMTS Command Reference | Cisco IOS CMTS Cable Command Reference Guide |

### Standards and RFCs

| Standards | Title |
|---|---|
| SP-RFIv1.1-I09-020830 | Data-over-Cable Service Interface Specifications Radio Frequency Interface Specification, version 1.1 |
| SP-RFIv2.0-I03-021218 | Data-over-Cable Service Interface Specifications Radio Frequency Interface Specification, version 2.0 |
| SP-OSSIv2.0-I03-021218 | Data-over-Cable Service Interface Specifications Operations Support System Interface Specification, version 2.0 |
| SP-BPI+-I09-020830 | Data-over-Cable Service Interface Specifications Baseline Privacy Plus Interface Specification, version 2.0 |

### 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.<br>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.<br>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password. | http://www.cisco.com/support |

# Feature Information for Virtual Interface Bundling

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 3: Feature Information for Virtual Interface Bundling*

| Feature Name | Releases | Feature Information |
|---|---|---|
| Virtual interface bundling | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on the Cisco cBR Series Converged Broadband Routers. |

CHAPTER 3

# IPv6 on Cable

Cisco cBR series Converged Broadband Router supports full IPv6 functionality.

The IPv6 feature support available in the Cisco IOS software and for Cisco CMTS routers is extensive. This document provides a comprehensive overview of all of the IPv6 features supported on the Cisco CMTS routers, and their restrictions.

However, the details of every feature are not covered in this document. The areas of IPv6 protocol support for the Cisco CMTS routers discussed in this document are classified by platform-independence or by platform-specific feature support.

- Platform-independent IPv6 features—Describes IPv6 features that are supported in the Cisco IOS software for several other Cisco platforms, and which generally do not have any platform-specific behavior or configuration differences on the Cisco CMTS routers.
- Documentation about the restrictions for these platform-independent features can be found in the Restrictions for IPv6 on Cable.
- Detailed information about these features, including conceptual and task-based configuration information, is documented outside of this feature and in the Cisco IOS software documentation. Detailed information about the location of this related documentation in the Cisco IOS software documentation is described in the Feature Information for IPv6 on Cable.

Platform-specific IPv6 features—Describes IPv6 features that are specific to the cable technology area and that only apply to the supported Cisco CMTS routers. The cable-specific IPv6 feature support includes new or modified cable features supporting IPv6, and any transparent support of the IPv6 protocol in existing (legacy) cable features on the CMTS router platforms.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

# Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 4: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Restrictions for IPv6 on Cable

## Multicast Restrictions

IPv6 multicast has the following behavior restrictions on the Cisco CMTS routers:

- ICMP redirects are not sent to the originating host if the packet is destined for another CPE behind the same CM. All CPE-to-CPE traffic is processed by the Cisco CMTS router.
- IPv6 multicast forwarding is not supported in Parallel Express Forwarding (PXF), therefore, the IPv6 multicast forwarding performance is limited by the Router Processor (RP).

The following areas of IPv6 multicast are not supported by the Cisco CMTS routers:

- Address family support for Multiprotocol Border Gateway Protocol (MBGP)
- Bidirectional Protocol Independent Multicast (PIM)
- Bootstrap router (BSR)
- DOCSIS 3.0 encrypted multicast
- Explicit tracking of receivers
- IPv6 multicast echo
- Multicast Forwarding Information Base (MFIB) display enhancements
- Multicast use authentication and profile support
- PIM embedded rendezvous point
- Protocol Independent Multicast sparse mode (PIM-SM) accept register feature
- Reverse path forwarding (RPF) flooding of bootstrap router (BSR) packets
- Routable address hello option
- Source Specific Multicast (SSM) mapping for Multicast Listener Device (MLD) version 1 SSM

## QoS Restrictions

In Cisco IOS-XE Release 16.5.1, the following fields are supported for theIPv6 downstream classification:

- IPv6 dest addr
- ipv6 src addr
- IPv6 next header
- IPv6 traffic class

**Note** IPv6 flow label field is not supported.

The following areas of DOCSIS QoS are not supported by the Cisco CMTS routers:

- Upstream IPv6 Type of Service (ToS) overwrite
- Downstream IPv6 classification

**Note** ToS overwrite, DOCSIS classification, and Modular QoS CLI (MQC) on Gigabit Ethernet are supported.

# Information About IPv6 on Cable

This section includes the following topics:

## Features Supported

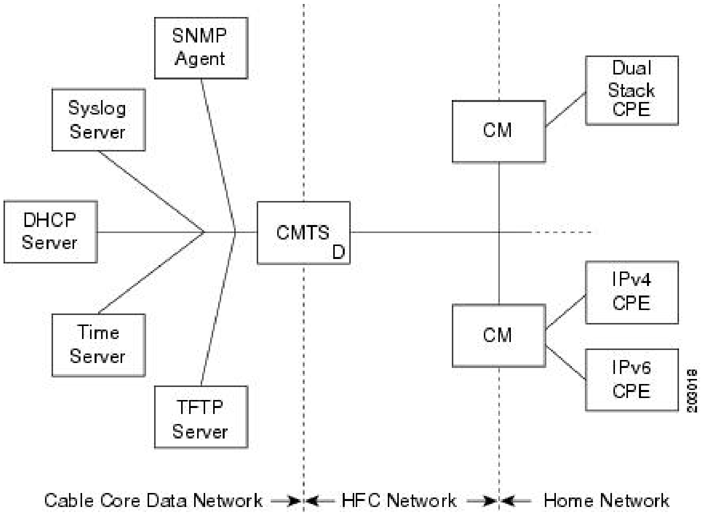
The following features are supported on the Cisco CMTS routers:

- Source verification of IPv6 packets in PXF
- ACL support for PXF
- ToS overwrite
- DOCSIS classification
- Modular QoS CLI (MQC) on Gigabit Ethernet
- IPv6 DOCSIS RP and LC HA and DCC
- MAC tapping of IPv6 packets
- Equal cost route load balancing of IPv6 packets destined to the backhaul
- IPv6 over IPv4 GRE tunnels
- Assignment of different prefixes to CM and CPE
- DHCPv6 over MPLS-VPN
- DHCPv6 relay prefix delegation VRF awareness
- Assignment of multiple IAPDs in a single advertise for each CPE.
- Assignment of multiple IA_NA and IAPD combinations to multiple CPEs behind a CM.
- The default maximum number of IA_NA and IAPD combinations for each cable modem is 16, including link-local addresses.
- IPv6 Downstream ToS overwrite.
- DHCPv6 Client Link-Layer Address Option (RFC 6939).
- Voice over IPv6. PacketCable Multimedia needs to be enabled before using this feature. For more information, see http://www.cisco.com/c/en/us/td/docs/cable/cbr/configuration/guide/b_pktcbl_pktcblmm/packetcable_and_packetcable_multimedia.html.

## Overview of the DOCSIS 3.0 Network Model Supporting IPv6

Figure below illustrates the network model described by the *DOCSIS 3.0 MAC and Upper Layer Protocols Interface Specification*.

*Figure 1: DOCSIS 3.0 Network Model*



In this model, the different devices support the following functions and services:

- Customer premises equipment (CPE)—Supports IPv4, IPv6, or dual stack operation.

**Note** Cisco cBR routers support CPE devices provisioned for dual stack operation.

- Cable modem (CM)—Functions as a bridging device and supports IPv4, IPv6, or dual stack operation.
- Cable modem termination system (CMTS) router—Works with the CM over the hybrid fiber coaxial cable (HFC) network to provide IPv4 and IPv6 network connectivity to the provisioning servers and the core data network behind the CMTS router.

The CMTS router supports IPv6 address assignment, routing, and forwarding of IPv6 multicast and unicast packets.

**Note** The Cisco cBR router supports only a single DHCPv6 IPv6 address per client cable modem or CPE. This restriction also applies to DHCPv6 Prefix Delegation prefixes. The reason for blocking more than one DHCPv6 address or prefix for a client is because the end-to-end network requires Source Address Selection (SAS) and all nodes in the end-to-end network may not support the correct SAS. Moreover, the SAS specification (RFC 3484) is being revised by the IETF to define the correct SAS behavior.

- Simple Network Management Protocol (SNMP) agent—Provides management tools to configure and query devices on the network.
- Syslog server—Collects messages from the CM to support its functions.
- Dynamic Host Control Protocol (DHCP) server—The DOCSIS 3.0 network model supports both DHCPv4 and DHCPv6 servers to control the assignment of IP addresses.
- Time server—Provides the current time to the CM.

- Trivial File Transport Protocol (TFTP) server—Provides the CM configuration file.

# Overview of Cable Modem IPv6 Address Provisioning

Prior to cable modem registration with a CMTS router, the CMTS router sends a MAC Domain Descriptor (MDD) message to provide information to the cable modem about its supported IP provisioning mode. You configure the CMTS router provisioning mode using the **cable ip-init** interface configuration command. For more information, see the Implementing IPv6 Addressing and Basic Connectivity for Cable Interfaces and Bundles, on page 45.

The MDD contains an IP initialization parameters type length value (TLV) that defines the IP version, management and alternate provisioning mode, and pre-registration downstream service ID (DSID) that is used by cable modems that are capable of downstream traffic filtering.

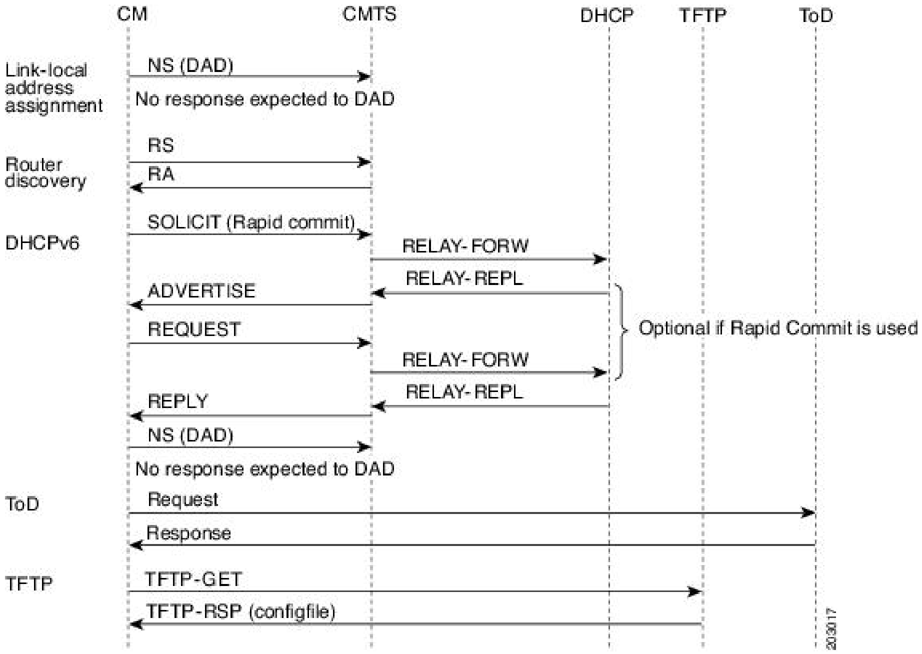
When IPv6 is configured and active, for transmitting multicast DSID carrying IPv6 ND messages within a MAC domain, the CMTS contains the pre-registration DSID TLV encoding (type 5.2) in the MDD message.

**Note** The Cisco CMTS routers do not support alternate provisioning mode or pre-registration DSID.

To support the MULPIv3.0 I04 or later version of the *DOCSIS 3.0 MAC and Upper Layer Protocols Interface Specification*, the cable modem must attempt IPv6 address acquisition first.

Figure below illustrates the message flow between a cable modem, the CMTS router, and the DHCP server when the cable modem is requesting an IPv6 address.

*Figure 2: Message Flow for CM Provisioning of DHCP IPv6 Address Assignment*



1. Link-local address assignment—The cable modem sends a Neighbor Solicit (NS) message with its link-local address (LLA) to the CMTS router, which starts the duplicate address detection (DAD) process for that LLA. The cable modem expects no response to the NS message.

2. Router discovery—The cable modem listens to the downstream to detect periodical Router Advertise (RA) messages. When an RA message is detected, the cable modem uses the data in the RA message to configure the default route. If an RA is not detected in a specified period, the cable modem sends a Router Solicit (RS) message to find the router on the link (all nodes multicast). The CMTS router responds with a Router Advertise (RA) message with theM and O bits set to 1 to instruct the CM to perform stateful address configuration.

**Note** Cisco CMTS routers do not support SLAAC address assignment.

- DHCPv6—The cable modem sends a DHCPv6 Solicit message to the CMTS router to request an IPv6 address. The CMTS router relays this message to the DHCPv6 servers. The DHCPv6 servers send an Advertise message indicating the server's availability.

If the Rapid-Commit option is not used by the cable modem, then the cable modem responds to the Advertise message of the server with a Request message to select the server that the CMTS router relays to the DHCPv6 server. If the Rapid-Commit option is used, then multiple DHCPv6 servers that could assign different addresses to the same CPE must not be used.

The cable modem starts the DAD process to verify the uniqueness of the IPv6 address that the DHCPv6 server assigns to it.

- TFTP and Time of Day (ToD)—Once the CM establishes IP connectivity, it sends a request to the TFTP server to download a configuration file and requests the current time from the ToD server to complete its boot process.

# Overview of IPv6 Dual Stack CPE Support on the CMTS

Most operating systems (OS) deployed at homes support dual stack operation. Cisco CMTS supports dual stack, which is both IPv4 and IPv6 addressing on the CPE.

# Overview of IPv6 over Subinterfaces

Cisco CMTS supports IPv6 over bundle subinterfaces. To configure IPv6 on bundle subinterfaces, see the Implementing IPv6 Addressing and Basic Connectivity for Cable Interfaces and Bundles, on page 45 section. For a CMTS bundle configuration example, see the Example: IPv6 over Subinterfaces , on page 60 section.

To enable IPv6 on subinterfaces, configure IPv6 on bundle subinterfaces and not the bundle. Reset the CMs after the subinterface is configured.

**Note** MPLS VPN over subinterfaces for IPv6 is not supported.

# Overview of High Availability on IPv6

Cisco cBR Series routers support IPv6 HA for the Supervisor card.

**Note** IPv6 DOCSIS HA and HCCP is supported on the Cisco CMTS routers.

The IPv6 HA feature support in Cisco CMTS routers covers the following capabilities:

- DOCSIS PRE HA
- DOCSIS line card HA
- Dynamic Channel Change (DCC)

## DOCSIS PRE HA

The DOCSIS PRE HA has the following behavior restrictions and prerequisites on the Cisco CMTS routers:

- The CMs and CPEs should not go offline after a PRE switchover.
- The data structures of the IPv6 CM and CPE should be synchronized to the standby PRE before the PRE switchover. Both dynamic and bulk synchronization is supported.
- Single stack, dual stack, and APM are supported for the CM.
- Single stack and dual stack provisioning modes are supported on the CPE.
- After a PRE switchover, the IPv6 neighbor entries are rebuilt by Neighbor Discovery (ND) messages on the standby PRE, and the IPv6 routes are rebuilt after converging the routing protocol.

## DOCSIS Line Card HA

The DOCSIS line card HA has the following behavior restrictions and prerequisites on the Cisco CMTS routers:

- The data structures of the IPv6 CM and CPE should be synchronized to the standby line card before the line card switchover. Both dynamic and bulk synchronization is supported.
- The CMs and CPEs should not fall offline after a line card switches over and reverts; the CMs and CPEs should behave the same as before the switchover.
- The DOCSIS line card HA supports both 4+1 and 7+1 redundancy.
- Traffic outages in IPv6 may be longer because traffic recovery occurs only after converging the routing protocol.

## Dynamic Channel Change

The Dynamic Channel Change (DCC) feature is supported on Cisco CMTS routers.

**Note** The behavior of the DCC for single stack IPv6 CM and CPE, or dual stack CM and CPE is the same as that of a single stack IPv4 CM and CPE.

The IPv6 and IPv4 DCC functionality has the following behavior restrictions and prerequisites on the Cisco CMTS routers:

### Narrowband Cable Modem

- If the source and destination MAC domains of the CM are on the same line card, DCC initialization techniques 0, 1, 2, 3, and 4 are used to move the CM and its associated CPE from one upstream or downstream to another; or move the CM and CPE from one upstream and downstream combination to another.

- If the source and destination MAC domains of the CM are on different line cards, you can use only the DCC initialization technique 0 to move the CM and its associated CPE across line cards.

### Wideband Cable Modem

- If the source and destination MAC domains of the CM are on the same line card, DCC initialization techniques 0, 1, 2, 3, and 4 are used to move the CM and its associated CPE from one upstream to another.
- If the primary downstream of a CM is changed after DCC, you can use only the DCC initialization technique 0 to move the CM and its associated CPE across line cards.

## Overview of IPv6 VPN over MPLS

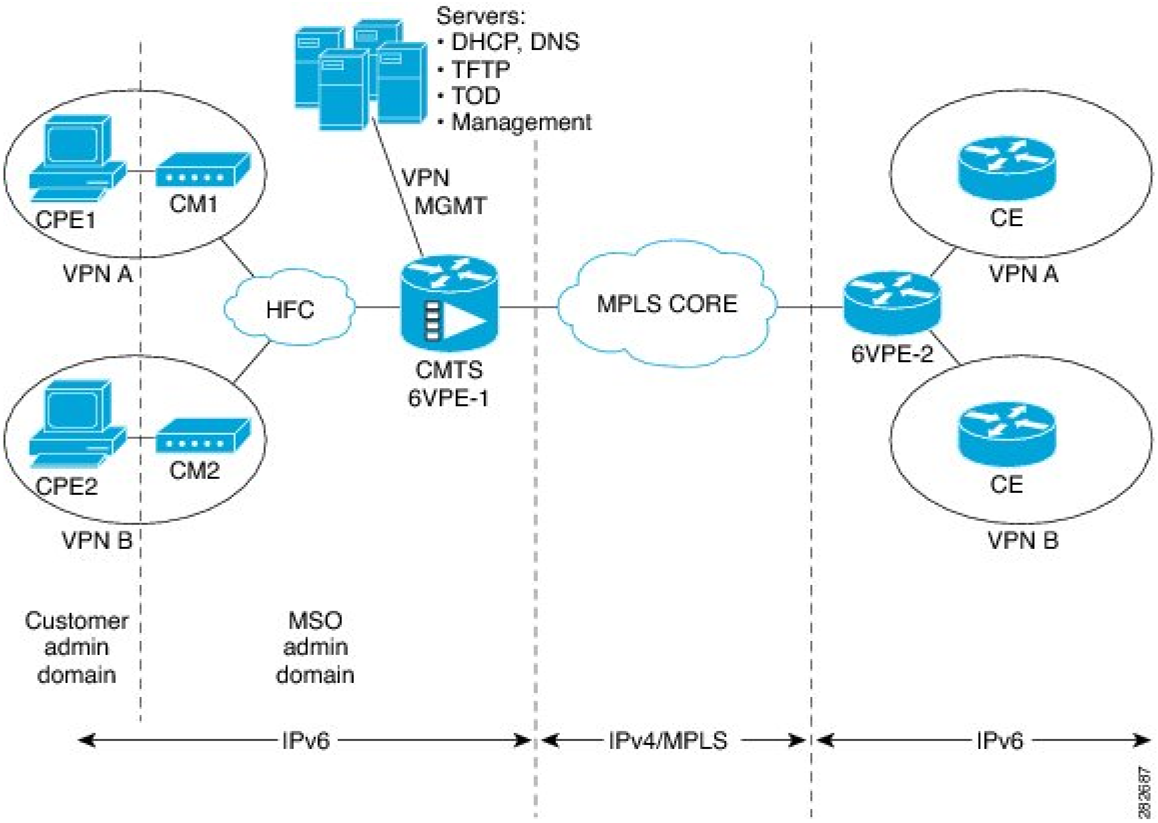
The Multiprotocol Label Switching (MPLS) VPN feature represents an implementation of the provider edge (PE) based VPN model. This document describes the IPv6 VPN over MPLS (6VPE) feature.

The 6VPE feature allows Service Providers to provide an IPv6 VPN service that does not require an upgrade or reconfiguration of the PE routers in the IPv4 MPLS Core. The resulting IPv6 VPN service has a configuration and operation which is virtually identical to the current IPv4 VPN service.

In principle, there is no difference between IPv4 and IPv6 VPNs. In both IPv4 and IPv6, the multiprotocol BGP is the core of the MPLS VPN for IPv6 (VPNv6) architecture. It is used to distribute IPv6 routes over the service provider backbone using the same procedures to work with overlapping addresses, redistribution policies, and scalability issues.

Figure below illustrates the 6PE/6VPE reference architecture diagram.

*Figure 3: 6PE/6VPE Reference Architecture*



# Cable Monitor

The Cable Monitor and Intercept features for Cisco CMTS routers provide a software solution for monitoring and intercepting traffic coming from a cable network. These features give service providers Lawful Intercept capabilities.

For more information, see the Cable Monitor and Intercept Features for the Cisco CMTS Routers guide.

# Overview of IPv6 CPE Router Support on the Cisco CMTS

The IPv6 CPE router support is provided on the Cisco CMTS. The IPv6 CPE router is a node primarily for home or small office use that connects the end-user network to a service provider network. It is also referred to as the home router.

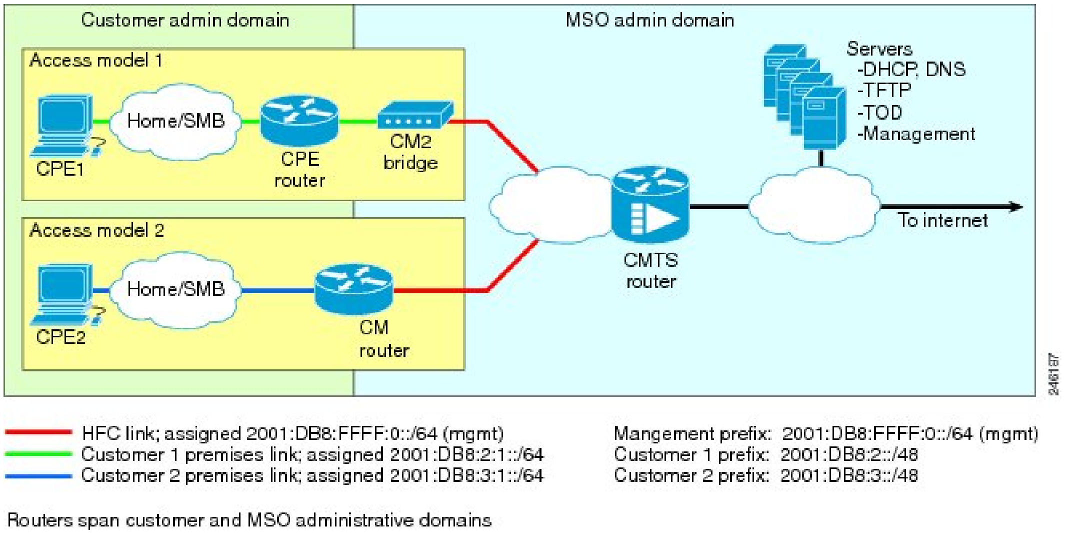
The IPv6 CPE router is responsible for implementing IPv6 routing; that is, the IPv6 CPE router looks up the IPv6 destination address in its routing table and decides to which interface the packet should be sent.

The IPv6 CPE router performs the following functions:

- Provisions its WAN interface automatically.
- Acquires IP address space for provisioning of its LAN interfaces.
- Fetches other configuration information from the service provider network.

Figure below illustrates the CPE router reference architecture diagram between the CPE router, the CMTS, and the DHCPv6 server (CNR) when the CM is requesting an IPv6 address.

*Figure 4: IPv6 CPE Router Reference Architecture*


Customer admin domain
MSO admin domain
Access model 1
Home/SMB
CPE1
CPE
router
CM2
bridge
Access model 2
Home/SMB
CPE2
CM
router
CMTS
router
Servers
-DHCP, DNS
-TFTP
-TOD
-Management
To internet
246187
HFC link; assigned 2001:DB8:FFFF:0::/64 (mgmt)
Customer 1 premises link; assigned 2001:DB8:2:1::/64
Customer 2 premises link; assigned 2001:DB8:3:1::/64
Mangement prefix: 2001:DB8:FFFF:0::/64 (mgmt)
Customer 1 prefix: 2001:DB8:2::/48
Customer 2 prefix: 2001:DB8:3::/48
Routers span customer and MSO administrative domains

As part of the IPv6 CPE Router Support feature, the following enhancements are introduced:

- Support to IPv6 router devices.
- IPv6 Prefix Delegation (PD) High Availability.
- Prefix awareness support in IPv6 cable source-verify, Cable DOCSIS filters code, and packet intercepts.

## Support for IPv6 Prefix Stability on the CMTS

IPv6 prefix stability is supported on the Cisco CMTS as specified in DOCSIS 3.0 MULPI CM-SP-MULPIv3.0-I15-110210 standard. The IPv6 prefix stability allows an IPv6 home router to move from one Cisco CMTS to another while retaining the same prefix.

The multiple service operators (MSOs) can use this feature to allow their business customers (with IPv6 routers) to retain the same IPv6 prefix during a node split.

## Configurable DHCPv6 Relay Address

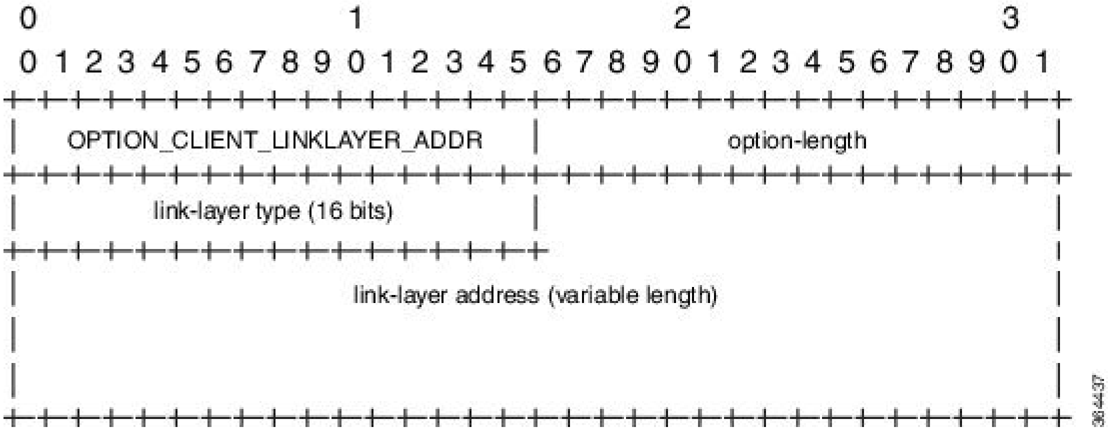
The DHCPv6 Cisco IOS relay agent on the Cisco CMTS router sends relay-forward messages from a source address to all configured relay destinations. The source address is either an IPv6 address provisioned on the network interface or a Cisco CMTS WAN IPv6 address. The relay destination can be a unicast address of a server, another relay agent, or a multicast address. The relay-forward messages contain specific DHCPv6 link-addresses.

A DHCP relay agent is used to relay messages between the client and server. A client locates a DHCP server using a reserved, link-scoped multicast address.

### DHCPv6 Client Link-Layer Address Option (RFC 6939)

Cisco IOS-XE Releases support DHCPv6 Client Link-Layer Address Option (RFC 6939). It defines an optional mechanism and the related DHCPv6 option to allow first-hop DHCPv6 relay agents (relay agents that are connected to the same link as the client) to provide the client's link-layer address in the DHCPv6 messages being sent towards the server.

The format of the DHCPv6 Client Link-Layer Address option is shown below.



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  OPTION_CLIENT_LINKLAYER_ADDR |         option-length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    link-layer type (16 bits)  |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|              link-layer address (variable length)             |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

| **Name** | **Description** |
|---|---|
| option-code | OPTION_CLIENT_LINKLAYER_ADDR (79) |
| option-length | 2 + length of MAC address |
| link-layer type | CPE or CM MAC address type. The link-layer type MUST be a valid hardware type assigned by the IANA, as described in RFC0826. |
| link-layer address | MAC address of the CPE or CM. |

**Note** RFC6939 is enabled by default. It can not be enabled/disabled by any CLI command.

To configure DHCPv6 Relay Address on the Cisco CMTS bundle subinterfaces, see the Configuring DHCPv6 Relay Agent, on page 56 section.

For more information about the DHCPv6 client, server, and relay functions, see the “Implementing DHCP for IPv6” chapter in the IPv6 Implementation Guide, Cisco IOS XE Release 3S.

## Support for Multiple IAPDs in a Single Advertise

Assignment of multiple IA_NA and IAPD to CPEs behind a CM is supported on Cisco CMTS routers. This feature includes support for link-local addresses and IA_NA and IAPD. However, a CM can be assigned only one IA_NA. This IA_NA can be either static or DHCP-assigned.

The CPEs behind the CM can request for multiple DHCPv6 IA_NAs and IAPDs. Each CPE is assigned multiple IA_NAs and IAPDs in a single Advertise/Reply message. Each CPE request for IA_NA and IAPD is treated as a separate Advertise/Reply message.

## IPv6 Neighbor Discovery Gleaning

The IPv6 Neighbor Discovery (ND) Gleaning feature enables Cisco CMTS routers to automatically recover lost IPv6 CPE addresses and update the CPE records in the Cisco CMTS subscriber database. The Cisco CMTS router gleans only the solicited neighbor advertise (NA) messages transmitted in the upstream direction. IPv6 ND gleaning is similar to Address Resolution Protocol (ARP) gleaning for IPv4 CPE recovery.

The IPv6 ND Gleaning feature is configured by default on Cisco CMTS routers. To disable this feature, use the **no** form of the **cable nd** command in bundle interface configuration mode. The **cable nd** command adds a CPE (host behind a cable modem) to the Cisco CMTS subscriber database. This command does not impact the IPv6 ND protocol operation on the router.

**Note** The IPv6 ND Gleaning feature does not support gleaning of NA messages transmitted in the downstream direction.

# How to Configure IPv6 on Cable

This section includes the following tasks:

## Configuring IPv6 Switching Services

The CMTS routers support forwarding of unicast and multicast IPv6 traffic using either Cisco Express Forwarding for IPv6 (CEFv6) or distributed CEFv6 (dCEFv6):

- CEFv6—All CMTS platforms
- dCEFv6—Cisco uBR10012 universal broadband router only

The CMTS routers also support Unicast Reverse Path Forwarding (RPF), as long as you enable Cisco Express Forwarding switching or distributed Cisco Express Forwarding switching globally on the router. There is no need to configure the input interface for Cisco Express Forwarding switching. As long as Cisco Express Forwarding is running on the router, individual interfaces can be configured with other switching modes.

To configure forwarding of IPv6 traffic using Cisco Express Forwarding or distributed Cisco Express Forwarding (supported on the Cisco uBR10012 universal broadband router only) on the CMTS routers, you must configure forwarding of IPv6 unicast datagrams using the **ipv6 unicast-routing** global configuration command, and you must configure an IPv6 address on the bundle interface using the **ipv6 address** command.

The **show ipv6 cef platform** command is supported on the Cisco CMTS platform. You can use the **show ipv6 cef platform** command for debugging purposes.

### Before you begin

- You must enable Cisco Express Forwarding for IPv4 globally on the router by using the **ip cef** or **ip cef distributed** command before configuring Cisco Express Forwarding v6 or distributed Cisco Express Forwarding v6.

**Note** The **ip cef** command is enabled by default on all Cisco CMTS routers. Therefore, you only must configure the command if it has been disabled. However, you must explicitly configure the **ip cef distributed** command on a Cisco uBR10012 universal broadband router if you want to run distributed CEF switching services for IPv4 or IPv6.

- You must configure forwarding of IPv6 unicast datagrams using the **ipv6 unicast-routing** global configuration command.
- You must configure IPv6 addressing on the cable bundle interface.
- CEF switching is required for Unicast RPF to work.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | Do one of the following:<br>• **ip cef**<br>• **ip cef distributed**<br>**Example:**<br>`Router(config)# ip cef`<br>`or`<br>`Router(config)# ip cef distributed` | Enables Cisco Express Forwarding.<br>or<br>Enables distributed Cisco Express Forwarding for IPv4 datagrams.<br>**Note** For CMTS routers, distributed Cisco Express Forwarding is supported only on a Cisco uBR10012 universal broadband router. |
| **Step 4** | Do one of the following:<br>• **ipv6 cef**<br>• **ipv6 cef distributed**<br>**Example:**<br>`Router(config)# ipv6 cef`<br>`or`<br>`Router(config)# ipv6 cef distributed` | Enables Cisco Express Forwarding v6.<br>or<br>Enables distributed Cisco Express Forwarding v6 for IPv6 datagrams.<br>**Note** For CMTS routers, distributed Cisco Express Forwarding v6 is supported only on a Cisco uBR10012 universal broadband router. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 5** | **ipv6 unicast-routing**<br>**Example:**<br>`Router(config)# ipv6 unicast-routing` | Enables the forwarding of IPv6 unicast datagrams. |

### What to do next

- (Optional) Enable IPv6 multicast routing using the **ipv6 multicast-routing** command in global configuration mode and configure other multicast features.

# Implementing IPv6 Addressing and Basic Connectivity for Cable Interfaces and Bundles

## Configuring the Cable Virtual Bundle Interface

The only required IPv6 configuration on a cable line card interface is the IP provisioning mode. The remainder of the IPv6 features are configured at the virtual bundle interface, which is then associated with a particular cable line card interface to establish its configuration.

Most of the IPv6 features that are supported in interface configuration mode (both cable-specific as well as platform-independent IPv6 features) are configured at a cable bundle interface.

The Cisco CMTS routers support IPv6 routing on the bundle interface and map both IPv6 unicast and multicast addresses into the cable bundle forwarding table, for packet forwarding.

Each bundle interface has a unique link-local address (LLA) to support link-local traffic when IPv6 is enabled. Cisco CMTS routers can support a maximum of 40 active bundle interfaces, which also translates to a maximum of 40 active IPv6-enabled bundle interfaces.

IPv6 commands can be configured on multiple bundle subinterfaces.

### Before you begin

The **cable ipv6 source-verify** and **cable nd** commands are not compatible with each other in Cisco IOS release 12.2(33)SCE and later. You must disable IPv6 ND gleaning using the **no** form of the **cable nd** command before using the **cable ipv6 source-verify** command to ensure that only DHCPv6 and SAV-based CPEs can send traffic on the router.

**Restriction** All multicast traffic is flooded onto bundle member interfaces.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:** | Enables privileged EXEC mode. Enter your password if prompted. |

| | Command or Action | Purpose |
|---|---|---|
| | `Router> enable` | |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface bundle** *n*<br>**Example:**<br>`Router(config)# interface bundle 1` | Specifies the cable bundle interface and enters interface configuration mode, where *n* specifies the number of the bundle interface. |
| **Step 4** | **ipv6 address***ipv6-prefix / prefix-length* [**eui-64** ]<br>**Example:**<br>`Router(config-if)# ipv6 address 2001:DB8::/32 eui-64` | Specifies an IPv6 network assigned to the interface and enables IPv6 processing on the interface. The ipv6 address eui-64 command configures site-local and global IPv6 addresses with an interface identifier (ID) in the low-order 64 bits of the IPv6 address. You need to specify only the 64-bit network prefix for the address; the last 64 bits are automatically computed from the interface ID. |
| **Step 5** | **ipv6 address***ipv6-prefix / prefix-length* **link-local**<br>**Example:**<br>`Router(config-if)# ipv6 address 2001:DB8::/32 link-local` | (Optional) Specifies an IPv6 address assigned to the interface and enables IPv6 processing on the interface. The **ipv6 address link-local** command configures a link-local address on the interface that is used instead of the link-local address that is automatically configured, when IPv6 is enabled on the interface (using the **ipv6 enable** command). |
| **Step 6** | **ipv6 enable**<br>**Example:**<br>`Router(config-if)# ipv6 enable` | Automatically configures an IPv6 link-local address on the interface while also enabling the interface for IPv6 processing. The link-local address can be used only to communicate with nodes on the same link. |
| **Step 7** | cable ipv6 source-verify<br>**Example:**<br>`Router(config-if)# cable ipv6 source-verify` | (Optional) Enables source verification of MAC address-MD-SID-IPv6 address binding packets received by a cable interface upstream on Cisco CMTS routers. |

### What to do next

- Configure the desired platform-independent IPv6 features on the bundle interface, such as Neighbor Discovery and DHCPv6 features.
- Configure the IP provisioning mode and bundle on the cable interface.

## Configuring the IP Provisioning Mode and Bundle on the Cable Interface

The CMTS routers allow you to configure cable interfaces to support cable modems provisioned for both IPv4 and IPv6 addressing support (known as “dual stack”), only IPv4 addressing, or only IPv6 addressing.

Prior to cable modem registration, the CMTS router sends its supported provisioning mode to the cable modem in the MDD message.

In addition to configuring the provisioning mode on the cable interface, you must also associate the cable interface with a cable bundle. You perform most of the other IPv6 feature configuration at the bundle interface.

**Note** This section describes only the commands associated with establishing IPv6 support on a CMTS router. Other cable interface commands that apply but are optional are not shown, such as to configure upstream and downstream features.

### Before you begin

Configuration of a bundle interface is required.

**Step 1** **enable**

**Example:**

```
Router> enable
```

Enables privileged EXEC mode. Enter your password if prompted.

**Step 2** **configure terminal**

**Example:**

```
Router# configure terminal
```

Enters global configuration mode.

**Step 3** **interface cable** {*slot* / *port* | *slot* / *subslot* /*port* }

**Example:**

```
Router(config)# interface cable 5/0/1
```

Specifies the cable interface line card, where:

The valid values for these arguments are dependent on your CMTS router and cable interface line card. Refer to the hardware documentation for your router chassis and cable interface line card for supported slot and port numbering.

**Step 4** **cable ip-init** {**apm** | **dual-stack** | **ipv4** | **ipv6**}

**Example:**

```
Router(config-if)# cable ip-init ipv6
```

Specifies the IP provisioning mode supported by the cable interface, where:

**Step 5** **cable bundle***n*

**Example:**

```
Router(config)# cable bundle 1
```

Associates the cable interface with a configured virtual bundle interface, where *n* specifies the number of the bundle interface.

**What to do next**

- Proceed to configuring any other cable interface features that you want to support, such as upstream and downstream features. For more information about the other cable interface features, refer to the *Cisco IOS CMTS Cable Software Configuration Guide*.
- Proceed to configure other optional IPv6 cable features.

## Enabling MDD with Pre-Registration DSID

When only IPv4 is configured and active, where the IP provisioning mode is IPv4, for Cable Modems to come online, the pre-registration DSID (TLV 5.2) is required in the MAC Domain Descriptor (MDD) message. To enable this pre-registration DSID, use the following command.

By default, this command is disabled.

```
Router#configure terminal
Router(config)#cable ipv4-prereg-dsid
Router(config)#end
```

# Configuring IPv6 Cable Filter Groups

The Cisco CMTS router supports IPv6 cable filter group capability with IPv6 filter options.

# Configuring IPv6 Cable Filter Groups

The Cisco CMTS router supports IPv6 cable filter group capability with IPv6 filter options.

## Cable Filter Groups and the DOCSIS Subscriber Management MIB

Cable subscriber management is a DOCSIS 1.1 specification, which can be established using the following configuration methods:

- CMTS router configuration (via CLI)
- SNMP configuration
- DOCSIS 1.1 configuration file (TLVs 35, 36, and 37)

This section describes the IPv6 cable filter group feature support of the packet filtering portion of the DOCSIS Subscriber Management MIB (DOCS-SUBMGMT-MIB) using configuration commands on the CMTS routers. This IPv6 cable filter group support extends filter classifiers with IPv6 addressing options for CM and CPE traffic, but is independent of DOCSIS IPv6 classifiers, which are used to match packets to service flows.

Configuration of IPv6 cable filter groups on the CMTS routers is supported according to the following guidelines:

- A cable filter group consists of a set of **cable filter group** commands that share the same group ID.
- Separate indexes can be used to define different sets of filters for the same group ID. This can be used to define both IPv4 and IPv6 filters to the same filter group.
- CMs can be associated with one upstream and one downstream filter group.

  - Upstream traffic—All traffic coming from CMs is evaluated against the assigned upstream filter group that is configured by the **cable submgmt default filter-group cm upstream** command.
  - Downstream traffic—All traffic going to CMs is evaluated against the assigned downstream filter group that is configured by the **cable submgmt default filter-group cm downstream** command.

- CPEs can be associated with one upstream and one downstream filter group.
  - Upstream traffic—All traffic coming from CPEs is evaluated against the assigned upstream filter group that is configured by the **cable submgmt default filter-group cpe upstream** command.
  - Downstream traffic—All traffic going to CPEs is evaluated against the assigned downstream filter group that is configured by the **cable submgmt default filter-group cpe downstream** command.

**Note** Because TLVs 35, 36, and 37 do not apply to DOCSIS 1.0 CM configuration files, the only way to enable cable subscriber management for a DOCSIS 1.0 CM is to configure it explicitly on the Cisco CMTS router and activate it by using the **cable submgmt default active** global configuration command.

### Before you begin

You must create the cable filter group before you assign it to a CM or CPE upstream or downstream.

**Restriction**

- Chained IPv6 headers are not supported.
- An individual filter group index cannot be configured to support both IPv4 and IPv6 versions at the same time. If you need to support IPv4 and IPv6 filters for the same filter group, then you must use a separate index number with the same filter group ID, and configure one index as **ip-version ipv4**, and the other index as **ip-version ipv6**.
- Only a single upstream and a single downstream filter group can be assigned for CM traffic.
- Only a single upstream and a single downstream filter group can be assigned to CPEs attached to a CM such that all CPEs behind a CM share a common filter group.
- For the filter group to work for CMs, a CM must re-register after the CMTS router is configured for the filter group.
- If parallel eXpress forwarding (PXF) is configured on the Cisco uBR10012 router, either the **cable filter group** commands or the interface ACL (**ip access-list**) command can be configured.
- If you do not provision TLVs 35, 36, and 37 in the DOCSIS CM configuration file, then you must activate the functionality by specifying the **cable submgmt default active** global configuration command on the CMTS router.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode. Enter your password if prompted. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **cable filter group***group-id* **index***index-num***dest-port***port-num*<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 dest-port 69` | (Optional) Specifies the TCP/UDP destination port number that should be matched. The valid range is from 0 to 65535. The default value matches all TCP/UDP port numbers (IPv4 and IPv6 filters). |
| **Step 4** | **cable filter group** *group-id* **index** *index-num* **ip-proto** *proto-type*<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 ip-proto 17` | (Optional) Specifies the IP protocol type number that should be matched. The valid range is from 0 to 256, with a default value of 256 that matches all protocols (IPv4 and IPv6 filters).<br>Some commonly used values are: |
| **Step 5** | **cable filter group** *group-id* **index** *index-num* **ip-tos** *tos-mask tos-value*<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 ip-tos 0xff 0x80` | (Optional) Specifies a ToS mask and value to be matched (IPv4 and IPv6 filters):<br>The *tos-mask* is logically ANDed with the *tos-value* and compared to the result of ANDing the *tos-mask* with the actual ToS value of the packet. The filter considers it a match if the two values are the same.<br>The default values for both parameters matches all ToS values. |
| **Step 6** | **cable filter group** *group-id* **index** *index-num* **ip-version ipv6**<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 ip-version ipv6` | Specifies that this filter group is an IPv6 filter group. |
| **Step 7** | **cable filter group** *group-id* **index** *index-num* **match-action** {**accept** \| **drop**}<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 match-action drop` | (Optional) Specifies the action that should be taken for packets that match this filter (IPv4 and IPv6 filters): |
| **Step 8** | **cable filter group** *group-id* **index** *index-num* **src-port** *port-num*<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 src-port 50` | (Optional) Specifies the TCP/UDP source port number that should be matched. The valid range is from 0 to 65535. The default value matches all TCP/UDP port numbers (IPv4 and IPv6 filters). |

| | Command or Action | Purpose |
|---|---|---|
| **Step 9** | **cable filter group** *group-id* **index** *index-num* **status** {**active** \| **inactive**}<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 status inactive` | (Optional) Enables or disables the filter (IPv4 and IPv6 filters):<br>**Note** You must create a filter group using at least one of the other options before you can use this command to enable or disable the filter. |
| **Step 10** | **cable filter group** *group-id* **index** *index-num* **tcp-flags** *flags-mask flags-value*<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 tcp-flags 0 0` | (Optional) Specifies the TCP flag mask and value to be matched (IPv4 and IPv6 filters): |
| **Step 11** | **cable filter group** *group-id* **index** *index-num* **v6-dest-address** *ipv6-address*<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 v6-dest-address 2001:DB8::/32` | (Optional) Specifies the IPv6 destination address that should be matched using the format X:X:X:X::X (IPv6 filters only). |
| **Step 12** | **cable filter group** *group-id* **index** *index-num* **v6-dest-pfxlen** *prefix-length*<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 v6-dest-pfxlen 64` | (Optional) Specifies the length of the network portion of the IPv6 destination address. The valid range is from 0 to 128. |
| **Step 13** | **cable filter group** *group-id* **index** *index-num* **v6-src-address** *ipv6-address*<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 v6-src-address 2001:DB8::/32` | (Optional) Specifies the IPv6 source address that should be matched using the format X:X:X:X::X (IPv6 filters only). |
| **Step 14** | **cable filter group** *group-id* **index** *index-num* **v6-src-pfxlen** *prefix-length*<br>**Example:**<br>`Router(config)# cable filter group 1 index 1 v6-src-pfxlen 48` | (Optional) Specifies the length of the network portion of the IPv6 source address. The valid range is from 0 to 128 (IPv6 filters only). |
| **Step 15** | **cable submgmt default filter-group** {**cm** \| **cpe**} {**downstream** \| **upstream**} *group-id*<br>**Example:**<br>`Router(config)# cable submgmt default filter-group cm upstream 1` | Applies a defined filter group (by specifying its *group-id)* to either a CM or its CPE devices, for downstream or upstream traffic. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 16** | **cable submgmt default active**<br>**Example:**<br>`Router(config)# cable submgmt default active` | (Required if you do not provision TLVs 35, 36, and 37 in the DOCSIS 1.1 CM configuration file)<br>Enables filters and allows the CMTS to manage the CPE devices for a particular CM (sets the docsSubMgtCpeActiveDefault attribute to TRUE). |

### Example

The following example shows how to create an IPv6 filter group with ID 254 and an index number of 128. The **ip-version ipv6** keywords must be configured to create the IPv6 filter group; otherwise, the default is an IPv4 filter group:

```
configure terminal
cable filter group 254
 index 128 v6-src-address 2001:DB8::/32
cable filter group 254
 index 128 v6-src-pfxlen 48
cable filter group 254
 index 128 v6-dest-address 2001:DB8::/32
cable filter group 254
 index 128 v6-dest-pfxlen 64
cable filter group 254
 index 128 ip-version ipv6
cable filter group 254
 index 128 match-action drop
cable submgmt default filter-group cm upstream 254
```

This group filters CM upstream traffic and drops any packets with an IPv6 source address of 2001:33::20B:BFFF:FEA9:741F (with network prefix of 128) destined for an IPv6 address of 2001:DB8::/32 (with network prefix of 128).

All of the **cable filter group** commands are associated by their group ID of 254 (and index of 128), and the **cable submgmt default filter-group** command applies the corresponding filter group ID of 254 to CM upstream traffic.

To monitor your cable filter group configuration, use forms of the **show cable filter** command as shown in the following examples. In these output examples, the output from the **show cable filter**, **show cable filter group 254**, and **show cable filter group 254 index 128** commands all display the same information because there is currently only a single filter group and index defined.

**Note** The "Use Verbose" string appears in the output area of the SrcAddr/mask and DestAddr/Mask fields suggesting use of the **show cable filter group verbose** form of the command to display the complete IPv6 address.

```
Router# show cable filter
Filter    SrcAddr/Mask       DestAddr/Mask      Prot ToS  SPort DPort TCP   Action Status
Grp Id v6                                                             Flags
254 128Y  Use Verbose
        Use Verbose
                                    drop   active
```

```
Router# show cable filter group 254
Filter    SrcAddr/Mask       DestAddr/Mask      Prot ToS  SPort DPort TCP   Action Status
Grp Id v6                                                             Flags
254 128Y  Use Verbose        Use Verbose                                    drop   active
Router# show cable filter group 254 index 128
Filter    SrcAddr/Mask       DestAddr/Mask      Prot ToS  SPort DPort TCP   Action Status
Grp Id v6                                                             Flags
254 128Y  Use Verbose        Use Verbose                                    drop   active
Router# show cable filter group 254 index 128 verbose
Filter Group                         : 254
Filter Index                         : 128
Filter Version                       : IPv6
Matches                              : 0
    Source IPv6 address              : 2001:DB8::/32
    Destination IPv6 address         : 2001:DB8::/32
    Match action                     : drop
    Status                           : active
```

## Troubleshooting Tips

You should configure the **cable filter group** commands prior to applying a filter group using the **cable submgmt default filter-group** command. Failure to do so results in the following message, and an association to a filter group that is undefined:

```
Router(config)# cable submgmt default filter-group cm upstream 100
Default value set to a nonexistent filter-group 100.
```

# Configuring IPv6 Domain Name Service

Cisco IOS releases support the domain name service (DNS) capability for devices using IPv6 addressing on the Cisco CMTS routers.

DNS simplifies the identification of cable devices by associating a hostname with what can often be a complex 128-bit IPv6 address. The hostname can then be used in place of the IPv6 address within the CMTS router CLI that supports use of hostnames.

There are two separate DNS caches supported on a CMTS router—an IOS DNS cache and a cable-specific DNS cache that stores IPv6 addresses learned by the CMTS router for CMs and CPEs.

In this phase of the IPv6 DNS service on cable, the DNS server is queried for domain name information as needed when you use the **show cable modem domain-name** command. When you use this command, the following actions take place:

1. The CMTS router checks whether CMs are online. If a CM is online, the CMTS router uses the corresponding IPv6 address assigned to the CM and looks up its domain name from the IOS DNS cache.
2. If no match is found, the CMTS router sends a DNS-QUERY message with the IPv6 address of the CM to the DNS server, which tries to resolve the domain name.
3. When the DNS reply is received, the CMTS router stores the domain name in the IOS DNS cache for each IPv6 address.
4. The CMTS router also stores the fully-qualified domain name (FQDN) that is replied by the DNS server in the cable-specific DNS cache.

**Note** Running the **no ip domain lookup** command turns off the DNS resolution.

The following platform-independent Cisco IOS-xe software commands are supported using host names by the CMTS router for IPv6 DNS on cable:

- **connect**
- **ping ipv6**
- **show hosts**
- **telnet**
- **traceroute**

### Before you begin

- A DNS server must be configured.
- You must identify and assign the host names to the IPv6 addresses. If you are using the Cisco DNS server, use the **ip host** global configuration command to map hostnames to IP addresses.
- You must configure the DNS server using the **ip name-server** global configuration command before use of DNS host names (or domains) are available in the supported commands.
- The **show cable modem domain-name** command must be run first on the Route Processor (RP) of the CMTS router before any domain name can be used as part of a cable command.

**Restriction**

- DNS for cable devices using IPv4 addressing is not supported.
- Due to column size limitations within the command-line interface (CLI), the domain name display is limited to 32 characters. Therefore, the entire domain name cannot always be seen in CMTS router command output.
- Only those cable devices where IPv6 address learning takes place are supported, such as acquiring an IPv6 address through DHCPv6 or the IPv6 (ND) process.
- The cable-specific DNS cache is only updated when you use the **show cable modem domain-name** command on the Route Processor (RP). A DNS-QUERY can only be sent on the RP using this command, therefore the DNS cache cannot update if you use the **show cable modem domain-name** command on a line card console. The output is displayed on the RP only.
- The cable-specific DNS cache does not store partially qualified domain names, only FQDNs are stored.
- The cable-specific DNS cache is not associated with the timeouts that apply to the IOS DNS cache. Therefore, a cable-specific DNS cache entry is not removed when an IOS DNS cache timeout occurs for that device. The cable-specific DNS cache is only updated when you use the **show cable modem domain-name** command.
- The CMTS router supports storage of only one domain name per IPv6 address in the cable-specific DNS cache.
- Domain names for the link local address are not supported.
- The **no ip domain-name** command disables DNS lookup.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip name-server** [**vrf** *vrf-name*] *server-address1* [*server-address2...server-address6*]<br>**Example:**<br>`Router(config)# ip name-server 2001:DB8::/32` | Specifies the address of one or more name servers to use for name and address resolution. |
| **Step 4** | **exit**<br>**Example:**<br>`Router(config)# exit` | Leaves global configuration mode and enters privileged EXEC mode. |
| **Step 5** | **show cable modem domain-name**<br>**Example:**<br>`Router# show cable modem domain-name` | Updates the cable-specific DNS cache and displays the domain name for all CMs and the CPE devices behind a CM. |

# Configuring IPv6 Source Verification

Typically, the IPv6 source verification feature is enabled on a cable bundle interface. From there, the cable interface is associated with the virtual bundle interface to acquire its configuration.

When you enable IPv6 source verification on a cable line card interface, the source verification routine verifies the MAC address-MD-SID-IP binding of the packet. If the source verification succeeds, the packet is forwarded. If the verification fails, the packet is dropped.

When a CM is operating as a bridge modem device, then the CMTS router verifies all the IPv6 addresses related to that CM and the CPEs behind that CM.

The **cable ipv6 source-verify** command controls only the source verification of IPv6 packets. For IPv4-based source verification, use the **cable source-verify** command, which also supports different options.

For more information about how to configure IPv6 source verification on a bundle interface, see the Configuring the Cable Virtual Bundle Interface, on page 45.

### Restrictions

Source verification of IPv6 packets occurs only on packets in the process-switched path of the Route Processor (RP).

# Configuring IPv6 VPN over MPLS

The Cisco CMTS routers support the IPv6 VPN over MPLS (6VPE) feature. Implementing this feature includes the following configuration tasks.

- Configuring a VRF instance for IPv6
- Binding a VRF to an interface
- Creating a subinterface
- Configuring a static route for PE-to-CE-routing
- Configuring eBGP PE-to-CE routing sessions
- Configuring the IPv6 VPN address family for iBGP
- Configuring route reflectors for improved scalability
- Configuring Internet access

For detailed information about the configuration examples, see Configuration Examples for IPv6 on Cable, on page 60.

**Note** The IPv6 address of the sub-bundle interface (to which the CM is connected) is used in the DHCPv6 relay packet of the CPE DHCPv6 request. If the DHCPv6 packet has to go from one VRF interface to another, the IPv6 address of each VRF interface should be configured on the Cisco CMTS to establish connectivity.

# Configuring DHCPv6 Relay Agent

The Cisco CMTS router supports DHCPv6 relay agent to forward relay-forward messages from a specific source address to client relay destinations.

Perform the steps given below to enable the DHCPv6 relay agent function and specify relay destination addresses on an interface.

### Before you begin

The relay-forward messages should contain specific source IPv6 address. This is required because the firewall deployed between the Cisco CMTS DHCPv6 relay agent and the DHCPv6 server expects only one source address for one Cisco CMTS bundle interface.

**Restriction** If you change one or more parameters of the **ipv6 dhcp relay destination** command, you have to disable the command using the **no** form, and execute the command again with changed parameters.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| Step 1 | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode. Enter your password if prompted. |
| Step 2 | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| Step 3 | **interface type** *number*<br>**Example:**<br>`Router(config)# interface ethernet 4/2` | Specifies an interface type and number, and places the router in interface configuration mode. |
| Step 4 | **ipv6 dhcp relay destination** *ipv6-address*[ *interface*] [**link-address** *link-address* ] [ **source-address** *source-address*]<br>**Example:**<br>`Router(config-if) ipv6 dhcp relay destination 2001:db8:1234::1 ethernet 4/2 link-address 2001:db8::1 source-address 2001:db8::2` | Specifies a destination address to which client packets are forwarded and enables DHCPv6 relay service on the interface. |
| Step 5 | **end**<br>**Example:**<br>`Router(config-if) end` | Exits interface configuration mode and enters privileged EXEC mode. |

# Configuring IPv6 Source Address and Link Address

In some network deployments, there is a firewall between CNR servers and CMTS router. The firewall only allows packets from certain addresses to transmit to CMTS router.

To allow the DHCPv6 message to be sent to CNR server with specific source address, user needs to configure the source address of the relayed forwarded DHCPv6 message in the following 3 ways:

1. In the global configuration mode, use the command **ipv6 dhcp-relay source-interface** *interface-type interface-number*
2. In the interface configuration mode, use the command **ipv6 dhcp relay source-interface** *interface-type interface-number*
3. In the interface configuration mode, use the command **ipv6 dhcp relay destination** *ipv6-address[interface] [***link-address** *link-address ] [***source-address** *source-address]*

If the user does not have any of the above configuration, the DHCPv6 message will be relay forwarded with default source-address, which will be calculated based on relay destination address.

If any one of above is configured, the source-address of the DHCPv6 message will be based on that configuration.

If more than one configuration above is configured, the overriding rule is that more specific configuration wins, i.e., 3>2>1.

# Configurable DOCSIS CMTS Capabilities DHCPv6 Field

According to DOCSIS 3.1 specification, CMTS must support DOCSIS 3.0 cable modems and features. To support the backward compatibility of DOCSIS versions, the DHCPv6 vendor option must change from 30 to 31.

You can now upgrade to any DOCSIS version using the **cable docsis-ver [major version | minor version]** command.

The default value of the command is **cable docsis-ver 3 1**.

# Disabling IPv6 ND Gleaning

You must disable IPv6 ND gleaning before configuring IPv6 source verification using DHCPv6 leasequery.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interfacebundle** *bundle-no*<br>**Example:**<br>`Router(config)# interface bundle 1` | Specifies a bundle interface number and enters bundle interface configuration mode.<br>• *bundle-no* —Bundle interface number. The valid range is from 1 to 255. |
| **Step 4** | **no cable nd**<br>**Example:**<br>`Router(config-if) no cable nd` | Disables IPv6 ND gleaning on the Cisco CMTS router. |
| **Step 5** | **end**<br>**Example:** | Returns to privileged EXEC mode. |

| | Command or Action | Purpose |
|---|---|---|
| | `Router(config-if) end` | |

# How to Verify IPv6 Dual Stack CPE Support

This section describes how to use **show** commands to verify the configuration of the IPv6 Dual Stack CPE Support on the CMTS feature.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **show cable modem** [*ip-address* \| *mac-address* ] **ipv6**[ **cpe** \| **prefix** \| **registered** \| **unregistered]**<br>**Example:**<br>`Router# show cable modem ipv6 registered`<br>**Example:**<br>`Router# show cable modem 0019.474a.c14a ipv6 cpe` | Displays IPv6 information for specified CMs and CPEs behind a CM on a Cisco CMTS router. You can specify the following options: |
| **Step 3** | **show cable modem** [ip-address \| mac-address] **registered**<br>**Example:**<br>`Router# show cable modem 0019.474e.e4DF registered` | Displays a list of the CMs that have registered with the Cisco CMTS. You can specify the following options: |
| **Step 4** | **show cable modem** {ip-address \| mac-address} **cpe**<br>**Example:**<br>`Router# show cable modem 0019.474a.c14a cpe` | Displays the CPE devices accessing the cable interface through a particular CM. You can specify the following options: |

# Examples

Use the **show cable modem ipv6** command to display the IPv6 portion of a dual stack CPE and use the **show cable modem cpe** command to display the IPv4 mode of a dual stack CPE. Both **show cable modem ipv6 registered** and **show cable modem registered** commands display CPE count as one for a dual stack CPE.

The following example shows the output of the **show cable modem ipv6** command:

```
Router# show cable modem ipv6 registered
Interface   Prim Online      CPE IP Address                              MAC Address
```

```
            Sid  State
C4/0/U2     1    online       0   ---                                     0019.474a.c18c
C4/0/U2     3    online(pt)   1   2001:420:3800:809:EDA4:350C:2F75:4779   0019.474a.c14a
Router# show cable modem 0019.474a.c14a ipv6 cpe
MAC Address    IP Address                             Domain Name
0005.0052.2c1d 2001:420:3800:809:48F7:3C33:B774:9185
```

The following example shows the output of the **show cable modem ipv6** command:

```
Router# show cable modem
 0023.bed9.4c8e ipv6 cpe
Load for five secs: 0%/0%; one minute: 1%; five minutes: 1%
Time source is hardware calendar, *06:37:20.439 UTC Thu Aug 2 2012
MAC Address    IP Address
0023.bed9.4c91 2001:40:3:4:200:5EB7:BB6:C759
2001:40:3:4:210:D73B:7A50:2D05
```

The following example shows the output of the **show cable modem registered** command:

```
Router# show cable modem registered
Interface   Prim Online      Timing Rec   QoS CPE IP address      MAC address
            Sid  State       Offset Power
C4/0/U2     3    online      1022   0.00  2   1   50.3.37.12      0019.474a.c14a
```

The following example shows the output of the **show cable modem cpe** command:

```
Router# show cable modem 0019.474a.c14a cpe

IP address        MAC address      Dual IP
 50.3.37.3 0005.0052.2c1d Y
```

# Configuration Examples for IPv6 on Cable

This section includes the following examples:

## Example: IPv6 over Subinterfaces

The following example shows the CMTS bundle configuration that can be used with subinterfaces:

```
Router# show cable modem ipv6
Device Type: B - CM Bridge, R - CM Router
IP Assignment Method: D - DHCP
MAC Address    Type Interface    Mac State    D/IP IP Address
0019.474a.c18c B/D  C4/0/U2      online        Y  2001:420:3800:809:4C7A:D518:91
C6:8A18
Router# show run interface bundle2
Building configuration...
Current configuration : 138 bytes
!
interface Bundle2
 no ip address
 cable arp filter request-send 3 2
 cable arp filter reply-accept 3 2
 no cable ip-multicast-echo
end
Router#

show run interface bundle2.1
```

```
Building configuration...
Current configuration : 382 bytes
!
interface Bundle2.1
 ip address 50.3.37.1 255.255.255.0
 no cable ip-multicast-echo
 cable helper-address 10.10.0.12
 ipv6 address 2001:DB8::/32
 ipv6 enable
 ipv6 nd prefix default no-advertise
 ipv6 nd managed-config-flag
 ipv6 nd other-config-flag
 ipv6 nd ra interval msec 2000
 ipv6 dhcp relay destination 2001:420:3800:800:203:BAFF:FE11:B644
 arp timeout 240
end
```

## Example: Basic IPv6 Cable Filter Groups

The following example shows the configuration of an IPv6 filter group that drops traffic from a specific IPv6 host (with source address 2001:DB8::1/48) behind a cable router to an IPv6 host on the network (with destination address 2001:DB8::5/64):

```
configure terminal
!
! Specify the filter group criteria using a common group ID
!
cable filter group 254 index 128 v6-src-address 2001:DB8::1
cable filter group 254 index 128 v6-src-pfxlen 128
cable filter group 254 index 128 v6-dest-address 2001:DB8::5
cable filter group 254 index 128 v6-dest-pfxlen 128
!
! Specify that the filter group is IP version 6
!
cable filter group 254 index 128 ip-version ipv6
!
! Specify the drop action for matching packets
!
cable filter group 254 index 128 match-action drop
!
! Apply the filter group with ID 254 to all CM upstream traffic
!
cable submgmt default filter-group cm upstream 254
```

## Example: Complete Cable Configuration with IPv6

The following example shows a complete cable configuration example; it also displays the configuration of multiple cable filter groups using both IPv4 and IPv6 and separate indexes to associate the filter definitions with the same group ID.

```
Router# show running-config
Building configuration...
Current configuration : 15010 bytes
!
! Last configuration change at 08:32:14 PST Thu Nov 8 2007
!
version 12.2
no service pad
```

```
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service internal
service compress-config
!
hostname router
!
boot-start-marker
boot-end-marker
!
enable password password1
!
no aaa new-model
clock timezone PST -9
clock summer-time PDT recurring
clock calendar-valid
facility-alarm core-temperature major 53
facility-alarm core-temperature minor 45
facility-alarm core-temperature critical 85
facility-alarm intake-temperature major 49
facility-alarm intake-temperature minor 40
facility-alarm intake-temperature critical 67
!
!
card 1/0 2jacket-1
card 1/0/0 24rfchannel-spa-1
card 5/0 5cable-mc520h-d
cable admission-control preempt priority-voice
cable modem vendor 00.18.68 SA-DPC2203
cable modem vendor 00.19.47 SA-DPC2505
no cable qos permission create
no cable qos permission update
cable qos permission modems
!
cable filter group 1 index 1 src-ip 0.0.0.0
cable filter group 1 index 1 src-mask 0.0.0.0
cable filter group 1 index 1 dest-ip 0.0.0.0
cable filter group 1 index 1 dest-mask 0.0.0.0
cable filter group 2 index 1 src-ip 0.0.0.0
cable filter group 2 index 1 src-mask 0.0.0.0
cable filter group 2 index 1 dest-ip 0.0.0.0
cable filter group 2 index 1 dest-mask 0.0.0.0
cable filter group 3 index 1 src-ip 0.0.0.0
cable filter group 3 index 1 src-mask 0.0.0.0
cable filter group 3 index 1 dest-ip 0.0.0.0
cable filter group 3 index 1 dest-mask 0.0.0.0
cable filter group 4 index 1 src-ip 0.0.0.0
cable filter group 4 index 1 src-mask 0.0.0.0
cable filter group 4 index 1 dest-ip 0.0.0.0
cable filter group 4 index 1 dest-mask 0.0.0.0
cable filter group 5 index 1 src-ip 0.0.0.0
cable filter group 5 index 1 src-mask 0.0.0.0
cable filter group 5 index 1 dest-ip 0.0.0.0
cable filter group 5 index 1 dest-mask 0.0.0.0
cable filter group 6 index 1 src-ip 0.0.0.0
cable filter group 6 index 1 src-mask 0.0.0.0
cable filter group 6 index 1 dest-ip 0.0.0.0
cable filter group 6 index 1 dest-mask 0.0.0.0
cable filter group 7 index 1 src-ip 0.0.0.0
cable filter group 7 index 1 src-mask 0.0.0.0
cable filter group 7 index 1 dest-ip 0.0.0.0
cable filter group 7 index 1 dest-mask 0.0.0.0
cable filter group 8 index 1 src-ip 0.0.0.0
cable filter group 8 index 1 src-mask 0.0.0.0
cable filter group 8 index 1 dest-ip 0.0.0.0
cable filter group 8 index 1 dest-mask 0.0.0.0
cable filter group 9 index 1 src-ip 0.0.0.0
cable filter group 9 index 1 src-mask 0.0.0.0
cable filter group 9 index 1 dest-ip 0.0.0.0
cable filter group 9 index 1 dest-mask 0.0.0.0
cable filter group 10 index 1 src-ip 0.0.0.0
cable filter group 10 index 1 src-mask 0.0.0.0
cable filter group 10 index 1 dest-ip 0.0.0.0
cable filter group 10 index 1 dest-mask 0.0.0.0
cable filter group 12 index 1 src-ip 0.0.0.0
cable filter group 12 index 1 src-mask 0.0.0.0
cable filter group 12 index 1 dest-ip 0.0.0.0
cable filter group 12 index 1 dest-mask 0.0.0.0
cable filter group 16 index 1 src-ip 0.0.0.0
cable filter group 16 index 1 src-mask 0.0.0.0
cable filter group 16 index 1 dest-ip 0.0.0.0
cable filter group 16 index 1 dest-mask 0.0.0.0
ip subnet-zero
ip domain name cisco.com
ip host host1 239.192.254.254
ip host host2 239.192.254.253
ip name-server 10.39.26.7
ip name-server 2001:0DB8:4321:FFFF:0:800:20CA:D8BA
!
!
!
!
ipv6 unicast-routing
ipv6 cef
packetcable multimedia
packetcable
!
!
!
redundancy
 mode sso
!
!
controller Modular-Cable 1/0/0
 annex B modulation 64qam 0 23
 ip-address 10.30.4.175
 modular-host subslot 5/0
 rf-channel 0 cable downstream channel-id 24
 rf-channel 1 cable downstream channel-id 25
 rf-channel 2 cable downstream channel-id 26
 rf-channel 3 cable downstream channel-id 27
 rf-channel 4 cable downstream channel-id 28
 rf-channel 5 cable downstream channel-id 29
 rf-channel 6 cable downstream channel-id 30
 rf-channel 7 cable downstream channel-id 31
 rf-channel 8 cable downstream channel-id 32
 rf-channel 9 cable downstream channel-id 33
 rf-channel 10 cable downstream channel-id 34
 rf-channel 11 cable downstream channel-id 35
 rf-channel 12 cable downstream channel-id 36
 rf-channel 13 cable downstream channel-id 37
 rf-channel 14 cable downstream channel-id 38
 rf-channel 15 cable downstream channel-id 39
 rf-channel 16 cable downstream channel-id 40
 rf-channel 17 cable downstream channel-id 41
 rf-channel 18 cable downstream channel-id 42
 rf-channel 19 cable downstream channel-id 43
 rf-channel 20 cable downstream channel-id 44
 rf-channel 21 cable downstream channel-id 45
 rf-channel 22 cable downstream channel-id 46
 rf-channel 23 cable downstream channel-id 47
!
!
policy-map foo
policy-map 1
policy-map cos
policy-map qpolicy
policy-map shape
policy-map dscp
!
!
!
!
!
!
interface Loopback0
 ip address 127.0.0.1 255.255.255.255
!
interface FastEthernet0/0/0
 ip address 10.39.21.10 255.255.0.0
 speed 100
 half-duplex
 ipv6 address 2001:DB8::/32
 ipv6 enable
!
interface Wideband-Cable1/0/0:0
 no cable packet-cache
 cable bonding-group-id 1
!
interface Wideband-Cable1/0/0:1
 no cable packet-cache
 cable bonding-group-id 2
!
interface Wideband-Cable1/0/0:2
 no cable packet-cache
 cable bonding-group-id 3
!
interface Wideband-Cable1/0/0:3
 no cable packet-cache
 cable bonding-group-id 4
!
interface Wideband-Cable1/0/0:4
 no cable packet-cache
 cable bundle 1
 cable bonding-group-id 5
 cable rf-channel 1 bandwidth-percent 60
!
interface Wideband-Cable1/0/0:5
 no cable packet-cache
 cable bundle 1
 cable bonding-group-id 6
 cable rf-channel 0 bandwidth-percent 40
 cable rf-channel 2
 cable rf-channel 3
!
interface Wideband-Cable1/0/0:6
 no cable packet-cache
 cable bonding-group-id 7
!
interface Wideband-Cable1/0/0:7
 no cable packet-cache
 cable bonding-group-id 8
!
interface Wideband-Cable1/0/0:8
 no cable packet-cache
 cable bonding-group-id 9
!
interface Wideband-Cable1/0/0:9
 no cable packet-cache
 cable bonding-group-id 33
!
interface Wideband-Cable1/0/0:10
 no cable packet-cache
 cable bonding-group-id 34
!
interface Wideband-Cable1/0/0:11
 no cable packet-cache
 cable bonding-group-id 35
!
interface Cable5/0/0
 no cable packet-cache
 cable bundle 1
 cable downstream channel-id 119
 cable downstream annex B
 cable downstream modulation 256qam
 cable downstream interleave-depth 32
 cable downstream frequency 99000000
 no cable downstream rf-shutdown
 cable upstream max-ports 4
 cable upstream 0 connector 0
 cable upstream 0 frequency 6000000
 cable upstream 0 ingress-noise-cancellation 200
 cable upstream 0 docsis-mode tdma
 cable upstream 0 channel-width 1600000 1600000
 cable upstream 0 minislot-size 4
 cable upstream 0 range-backoff 3 6
 cable upstream 0 modulation-profile 21
 no cable upstream 0 shutdown
 cable upstream 1 connector 1
 cable upstream 1 ingress-noise-cancellation 200
 cable upstream 1 docsis-mode tdma
 cable upstream 1 channel-width 1600000 1600000
 cable upstream 1 minislot-size 4
 cable upstream 1 range-backoff 3 6
 cable upstream 1 modulation-profile 21
 cable upstream 1 shutdown
 cable upstream 2 connector 2
 cable upstream 2 ingress-noise-cancellation 200
 cable upstream 2 docsis-mode tdma
 cable upstream 2 channel-width 1600000 1600000
 cable upstream 2 minislot-size 4
 cable upstream 2 range-backoff 3 6
 cable upstream 2 modulation-profile 21
 cable upstream 2 shutdown
 cable upstream 3 connector 3
 cable upstream 3 ingress-noise-cancellation 200
 cable upstream 3 docsis-mode tdma
 cable upstream 3 channel-width 1600000 1600000
 cable upstream 3 minislot-size 4
 cable upstream 3 range-backoff 3 6
 cable upstream 3 modulation-profile 21
 cable upstream 3 shutdown
!
interface Cable5/0/1
 cable ip-init ipv6
 no cable packet-cache
 cable bundle 1
 cable downstream channel-id 120
 cable downstream annex B
 cable downstream modulation 64qam
 cable downstream interleave-depth 32
 cable downstream frequency 705000000
 no cable downstream rf-shutdown
 cable upstream max-ports 4
 cable upstream 0 connector 4
 cable upstream 0 frequency 6000000
 cable upstream 0 ingress-noise-cancellation 200
 cable upstream 0 docsis-mode tdma
 cable upstream 0 channel-width 1600000 1600000
 cable upstream 0 minislot-size 4
 cable upstream 0 range-backoff 3 6
 cable upstream 0 modulation-profile 21
 no cable upstream 0 shutdown
 cable upstream 1 connector 5
 cable upstream 1 ingress-noise-cancellation 200
 cable upstream 1 docsis-mode tdma
 cable upstream 1 channel-width 1600000 1600000
 cable upstream 1 minislot-size 4
 cable upstream 1 range-backoff 3 6
 cable upstream 1 modulation-profile 21
 cable upstream 1 shutdown
 cable upstream 2 connector 6
 cable upstream 2 ingress-noise-cancellation 200
 cable upstream 2 docsis-mode tdma
 cable upstream 2 channel-width 1600000 1600000
 cable upstream 2 minislot-size 4
 cable upstream 2 range-backoff 3 6
 cable upstream 2 modulation-profile 21
 cable upstream 2 shutdown
 cable upstream 3 connector 7
 cable upstream 3 ingress-noise-cancellation 200
 cable upstream 3 docsis-mode tdma
 cable upstream 3 channel-width 1600000 1600000
 cable upstream 3 minislot-size 4
 cable upstream 3 range-backoff 3 6
 cable upstream 3 modulation-profile 21
 cable upstream 3 shutdown
!
interface Cable5/0/2
 no cable packet-cache
 cable downstream channel-id 121
 cable downstream annex B
 cable downstream modulation 64qam
 cable downstream interleave-depth 32
 cable downstream rf-shutdown
 cable upstream max-ports 4
 cable upstream 0 connector 8
 cable upstream 0 ingress-noise-cancellation 200
 cable upstream 0 docsis-mode tdma
 cable upstream 0 channel-width 1600000 1600000
 cable upstream 0 minislot-size 4
 cable upstream 0 range-backoff 3 6
 cable upstream 0 modulation-profile 21
 cable upstream 0 shutdown
 cable upstream 1 connector 9
 cable upstream 1 ingress-noise-cancellation 200
 cable upstream 1 docsis-mode tdma
 cable upstream 1 channel-width 1600000 1600000
 cable upstream 1 minislot-size 4
 cable upstream 1 range-backoff 3 6
 cable upstream 1 modulation-profile 21
 cable upstream 1 shutdown
 cable upstream 2 connector 10
 cable upstream 2 ingress-noise-cancellation 200
 cable upstream 2 docsis-mode tdma
 cable upstream 2 channel-width 1600000 1600000
 cable upstream 2 minislot-size 4
 cable upstream 2 range-backoff 3 6
 cable upstream 2 modulation-profile 21
 cable upstream 2 shutdown
 cable upstream 3 connector 11
 cable upstream 3 ingress-noise-cancellation 200
 cable upstream 3 docsis-mode tdma
 cable upstream 3 channel-width 1600000 1600000
 cable upstream 3 minislot-size 4
 cable upstream 3 range-backoff 3 6
 cable upstream 3 modulation-profile 21
 cable upstream 3 shutdown
!
interface Cable5/0/3
 no cable packet-cache
 cable downstream channel-id 122
 cable downstream annex B
 cable downstream modulation 64qam
 cable downstream interleave-depth 32
 cable downstream rf-shutdown
 cable upstream max-ports 4
 cable upstream 0 connector 12
 cable upstream 0 ingress-noise-cancellation 200
 cable upstream 0 docsis-mode tdma
 cable upstream 0 channel-width 1600000 1600000
 cable upstream 0 minislot-size 4
 cable upstream 0 range-backoff 3 6
 cable upstream 0 modulation-profile 21
 cable upstream 0 shutdown
 cable upstream 1 connector 13
 cable upstream 1 ingress-noise-cancellation 200
 cable upstream 1 docsis-mode tdma
 cable upstream 1 channel-width 1600000 1600000
 cable upstream 1 minislot-size 4
 cable upstream 1 range-backoff 3 6
 cable upstream 1 modulation-profile 21
 cable upstream 1 shutdown
 cable upstream 2 connector 14
 cable upstream 2 ingress-noise-cancellation 200
 cable upstream 2 docsis-mode tdma
 cable upstream 2 channel-width 1600000 1600000
 cable upstream 2 minislot-size 4
 cable upstream 2 range-backoff 3 6
 cable upstream 2 modulation-profile 21
 cable upstream 2 shutdown
 cable upstream 3 connector 15
 cable upstream 3 ingress-noise-cancellation 200
 cable upstream 3 docsis-mode tdma
 cable upstream 3 channel-width 1600000 1600000
 cable upstream 3 minislot-size 4
 cable upstream 3 range-backoff 3 6
 cable upstream 3 modulation-profile 21
 cable upstream 3 shutdown
!
interface Cable5/0/4
 no cable packet-cache
 cable downstream channel-id 123
 cable downstream annex B
 cable downstream modulation 64qam
 cable downstream interleave-depth 32
 cable downstream rf-shutdown
 cable upstream max-ports 4
 cable upstream 0 connector 16
 cable upstream 0 ingress-noise-cancellation 200
 cable upstream 0 docsis-mode tdma
 cable upstream 0 channel-width 1600000 1600000
 cable upstream 0 minislot-size 4
 cable upstream 0 range-backoff 3 6
 cable upstream 0 modulation-profile 21
 cable upstream 0 shutdown
 cable upstream 1 connector 17
 cable upstream 1 ingress-noise-cancellation 200
 cable upstream 1 docsis-mode tdma
 cable upstream 1 channel-width 1600000 1600000
 cable upstream 1 minislot-size 4
 cable upstream 1 range-backoff 3 6
 cable upstream 1 modulation-profile 21
 cable upstream 1 shutdown
 cable upstream 2 connector 18
 cable upstream 2 ingress-noise-cancellation 200
 cable upstream 2 docsis-mode tdma
 cable upstream 2 channel-width 1600000 1600000
 cable upstream 2 minislot-size 4
 cable upstream 2 range-backoff 3 6
 cable upstream 2 modulation-profile 21
 cable upstream 2 shutdown
 cable upstream 3 connector 19
 cable upstream 3 ingress-noise-cancellation 200
 cable upstream 3 docsis-mode tdma
 cable upstream 3 channel-width 1600000 1600000
 cable upstream 3 minislot-size 4
 cable upstream 3 range-backoff 3 6
 cable upstream 3 modulation-profile 21
 cable upstream 3 shutdown
!
interface Bundle1
 ip address 10.46.2.1 255.255.0.0 secondary
 ip address 10.46.1.1 255.255.0.0
 cable arp filter request-send 3 2
 cable arp filter reply-accept 3 2
 cable dhcp-giaddr policy strict
 cable helper-address 10.39.26.8
 ipv6 address 2001:DB8::/32
 ipv6 enable
 ipv6 nd managed-config-flag
 ipv6 nd other-config-flag
 ipv6 nd ra interval 5
 ipv6 dhcp relay destination 2001:0DB8:4321:FFFF:0:800:20CA:D8BA
!
ip default-gateway 10.39.0.1
ip classless
ip route 0.0.0.0 0.0.0.0 10.39.26.12
ip route 192.168.254.253 255.255.255.255 10.39.0.1
ip route 192.168.254.254 255.255.255.255 10.39.0.1
!
!
no ip http server
no ip http secure-server
!
logging cmts cr10k log-level errors
cpd cr-id 1
```

```
nls resp-timeout 1
cdp run
!
tftp-server bootflash:docs10.cm alias docs10.cm
tftp-server bootflash:rfsw_x373.bin alias rfsw_x373.bin
snmp-server community private RW
snmp-server enable traps cable
snmp-server manager
!
!
control-plane
!
!
line con 0
 logging synchronous
 stopbits 1
line aux 0
line vty 0 4
 password lab
 login
!
!
cable fiber-node 1
  downstream Modular-Cable 1/0/0 rf-channel 1
  upstream Cable 5/0 connector 0
!
cable fiber-node 2
  downstream Modular-Cable 1/0/0 rf-channel 0 2-3
  upstream Cable 5/0 connector 4
!
end
```

# Example: BGP Configuration for 6VPE

The following example shows a sample BGP configuration on CMTS 6VPE.

```
Router# router bgp 1
 no synchronization
 bgp log-neighbor-changes
 neighbor 11.1.1.5 remote-as 1
 neighbor 11.1.1.5 update-source Loopback1
 no auto-summary
 !
 address-family vpnv6              --- Enable vpnv6 AF
  neighbor 11.1.1.5 activate      --- Activate neighbor 6VPE-2
  neighbor 11.1.1.5 send-community extended
 exit-address-family
 !
 address-family ipv6 vrf vrf_mgmt
  redistribute connected            ---- Publish directly connected route
  redistribute static
  no synchronization
 exit-address-family
!
 address-family ipv6 vrf vrfa   --- Enable IPv6 vrf AF for each VRF
  redistribute connected
  no synchronization
 exit-address-family
 !
 address-family ipv6 vrf vrfb --- Enable IPv6 vrf AF for each VRF
  redistribute connected
  no synchronization
 exit-address-family
!
```

## Example: Subinterface Configuration for 6VPE

The following example shows how to define a subinterface on virtual bundle interface 1.

When configuring IPv6 VPNs, you must configure the first subinterface created as a part of the management VRF. In the following example, Bundle 1.10 is the first sub-interface, which is configured into management VRF. Make sure the CNR server is reachable in management VRF.

```
interface Bundle1.10                --- Management VRF
 vrf forwarding vrf_mgmt
 cable dhcp-giaddr primary
 ipv6 address 2001:40:3:110::1/64
 ipv6 enable
 ipv6 nd managed-config-flag
 ipv6 nd other-config-flag
 ipv6 dhcp relay destination 2001:10:74:129::2
interface Bundle1.11          --- VRF A
 vrf forwarding vrfa
 cable dhcp-giaddr primary
 ipv6 address 2001:40:3:111::1/64
 ipv6 enable
 ipv6 dhcp relay destination 2001:10:74:129::2
interface Bundle1.12          --- VRFB
 vrf forwarding vrfb
 cable dhcp-giaddr primary
 ipv6 address 2001:40:3:112::1/64
 ipv6 enable
 ipv6 dhcp relay destination 2001:10:74:129::2
```

## Example: Cable Interface Bundling

The following example shows how to bundle a group of physical interfaces.

```
int C5/0/4 and int c5/0/3 are bundled.
int c5/0/4
cable bundle 1
int c5/0/3
cable bundle 1
```

## Example: VRF Configuration for 6VPE

The following example shows how to create VRFs for each VPN.

```
vrf definition vrf_mgmt
 rd 1:1
 !
 address-family ipv4
 route-target export 1:1
 route-target import 1:1
 route-target import 2:2
 route-target import 2:1
 exit-address-family
 !
 address-family ipv6
 route-target export 1:1
 route-target import 1:1
 route-target import 2:1  -- import route of vrfa
 route-target import 2:2  -- import route of vrfb
 exit-address-family
```

# Verifying IPv6 on Cable

This section explains how to verify IPv6 on cable configuration and it contains the following topics:

## Verifying IPv6 VRF Configuration

To verify the IPv6 VRF configuration, use the show vrf ipv6 command in privileged EXEC mode.

```
Router# show vrf ipv6 vrfa
  Name                             Default RD          Protocols   Interfaces
  vrfa                             2:1                 ipv4,ipv6   Bu1.11
Router# show vrf ipv6 interfaces
Interface              VRF                              Protocol   Address

Bu1.10                 vrf_mgmt                         up         2001:40:3:110::1

Fa0/0/0                vrf_mgmt                         up         2001:20:4:1::38

Bu1.11                 vrfa                             up         2001:40:3:111::1

Bu1.12                 vrfb                             up         2001:40:3:112::1

CMTS#
```

## Verifying IPv6 BGP Status

To verify the IPv6 BGP status, use the show ip bgp command in privileged EXEC mode.

```
Router# show ip bgp vpnv6 unicast all neighbors

BGP neighbor is 11.1.1.5,  remote AS 1, internal link
  BGP version 4, remote router ID 11.1.1.5
  Session state = Established, up for 00:35:52
  Last read 00:00:37, last write 00:00:14, hold time is 180, keepalive interval is 60 seconds

  BGP multisession with 2 sessions (2 established), first up for 00:40:07
  Neighbor sessions:
    2 active, is multisession capable
  Neighbor capabilities:
    Route refresh: advertised and received(new) on session 1, 2
    Address family IPv4 Unicast: advertised and received
    Address family VPNv6 Unicast: advertised and received
......
```

## Verifying MPLS Forwarding Table

To verify the output of the MPLS forwarding table, use the show mpls forwarding-table command in the privileged EXEC mode.

```
Router# show mpls forwarding-table

Local  Outgoing      Prefix            Bytes Label   Outgoing   Next Hop
Label  Label or VC   or Tunnel Id      Switched      interface
......
19     No Label      2001:40:3:110::/64[V]   \                             ---Route in
vrf_mgmt
                                       0             aggregate/vrf_mgmt
21     No Label      2001:40:3:111::/64[V]   \                             ---Route in
vrfa
                                       0             aggregate/vrfa
22     No Label      2001:40:3:112::/64[V]   \                             ---Route in
vrfb
                                       0             aggregate/vrfb
......
```

# Verifying IPv6 Cable Modem and its Host State

To verify IPv6 addresses and connected host states of cable modems and CPEs, use the **show interface cable modem** command in the privileged EXEC mode:

```
Router# show interface cable 7/0/0 modem ipv6
SID  Type State         IPv6 Address                            M MAC address
11   CM  online         2001:420:3800:809:3519:5F9C:B96A:D31    D 0025.2e2d.743a
11   CPE unknown        2001:420:3800:809:3DB2:8A6C:115F:41D8   D 0011.2544.f33b
```

# Verifying Multiple IAPDs in a Single Advertise

To verify the multiple IPv6 prefixes assigned to devices on a network, use the show cable modem ipv6 prefix command in privileged EXEC mode:

```
Router# show cable modem ipv6 prefix
Load for five secs: 1%/0%; one minute: 1%; five minutes: 1%
Time source is hardware calendar, *06:36:53.075 UTC Thu Aug 2 2012
Device Type: B - CM Bridge, R - CM Router
IP Assignment Method: D - DHCP
MAC Address    Type IPv6 prefix
0023.bed9.4c91 R/D  2001:40:1012::/64
               R/D  2001:40:2012:1::/64
0000.002e.074c R/D  2001:40:1012:8::/64
               R/D  2001:40:2012:1D::/64
0000.002e.074b R/D  2001:40:1012:23::/64
               R/D  2001:40:2012:1C::/64
0000.002e.074a R/D  2001:40:1012:22::/64
               R/D  2001:40:2012:1B::/64
```

To verify the multiple IPv6 prefixes assigned to CPEs behind a CM with a specific MAC address, use the **show cable modem** *mac-address* **ipv6 prefix** command in privileged EXEC mode:

```
Router# show cable modem 0023.bed9.4c8e ipv6 prefix
Load for five secs: 0%/0%; one minute: 1%; five minutes: 1%
Time source is hardware calendar, *06:37:22.335 UTC Thu Aug 2 2012
Device Type: B - CM Bridge, R - CM Router
IP Assignment Method: D - DHCP
MAC Address    Type IPv6 prefix
0023.bed9.4c91 R/D  2001:40:1012::/64
               R/D  2001:40:2012:1::/64
```

To verify the IPv6 information of CPEs behind a CM with a specific MAC address, use the show cable modem *mac-address* ipv6 **cpe** command in privileged EXEC mode:

```
Router# show cable modem 0023.bed9.4c8e ipv6 cpe
Load for five secs: 0%/0%; one minute: 1%; five minutes: 1%
Time source is hardware calendar, *06:37:20.439 UTC Thu Aug 2 2012
MAC Address    IP Address
0023.bed9.4c91 2001:40:3:4:200:5EB7:BB6:C759
               2001:40:3:4:210:D73B:7A50:2D05
```

# Supported MIBs

### CISCO-DOCS-EXT-MIB

The CISCO-DOCS-EXT-MIB contains objects that support extensions to the Data-over-Cable Service Interface Specifications (DOCSIS) interface MIB, DOCS-IF-MIB.

- CdxBundleIpHelperEntry—Provides a list of cable helper entries on the bundle and sub-bundle interfaces.
- CdxBundleIPv6DHCPRelayEntry—Contains the IPv6 DHCP relay option, IPv6 DHCP relay source-interface details, and IPv6 DHCP relay trust configuration on a bundle and sub-bundle interface.
- CdxBundleIPv6DHCPRelayDestEntry—Contains a list of IPv6 DHCP relay destination entries on the cable bundle and sub-bundle interfaces.

# Additional References

### 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.<br>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.<br>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password. | http://www.cisco.com/support |

# Feature Information for IPv6 on Cable

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 5: Feature Information for Downstream Interface Configuration*

| Feature Name | Releases | Feature Information |
|---|---|---|
| Enabling MDD with Pre-Registration DSID | Cisco IOS XE Gibraltar 16.12.1x | This feature was introduced on the Cisco cBR Series Converged Broadband Routers. |
| Configurable DOCSIS CMTS Capabilities DHCPv6 Field | Cisco IOS XE Fuji 16.7.1 | This feature was introduced on the Cisco cBR Series Converged Broadband Routers. |
| IPv6 on cable | Cisco IOS XE Everest 16.6.1 | This feature was integrated into the Cisco cBR Series Converged Broadband Routers. |

CHAPTER 4

# Cable DHCP Leasequery

This document describes the Dynamic Host Configuration Protocol (DHCP) Leasequery feature on the Cisco cable modem termination system (CMTS) router.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 6: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Prerequisites for Cable DHCP Leasequery

- You must configure a cable interface with the **cable source-verify dhcp** command and the **no cable arp** command before the Cisco CMTS router can enable DHCP Leasequery. Lease queries are sent to the DHCP server or to a configured alternate server.

To divert DHCP Leasequeries to a specific server, you must use the cable **source-verify dhcp server** ipaddress command and the **no cable arp** command before the Cisco CMTS router is enabled for DHCP Leasequery. Only one alternate server may be configured.

- You must configure the **ipv6 route** command when IPv6 Customer Premise Equipment (CPE) routers are deployed on the Cisco CMTS router.

# Restrictions for Cable DHCP Leasequery

- Leasequeries are sent to the DHCP server unless an alternate server is configured.
- Only one alternate server can be configured.
- Users are responsible for the synchronization of the DHCP server and the configured alternate server.
- If the configured alternate server fails, leasequery requests are *not* returned to the DHCP server.
- Only one IA_IADDR is supported per client. If the leasequery returns multiple results, only the IA_ADDR matching the query is added to the Cisco CMTS subscriber database.
- The Cisco CMTS will not verify the source of the IPv6 link-local address of a CPE.

# Information About Cable DHCP Leasequery

Problems can occur when viruses, denial of service (DoS) attacks, and theft-of-service attacks begin scanning a range of IP addresses, in an attempt to find unused addresses. When the Cisco CMTS router is verifying unknown IP addresses, this type of scanning generates a large volume of DHCP leasequeries, which can result in the following problems:

- High CPU utilization on the Cisco CMTS router PRE card.
- High utilization on the DHCP servers, resulting in a slow response time or no response at all.
- Packets can be dropped by the Cisco CMTS router or DHCP server (or configured alternate server).
- Lack of available bandwidth for other customers on the cable interface.

To prevent such a large volume of leasequery requests on cable interfaces, you can enable filtering of these requests on upstream interfaces, downstream interfaces, or both. When the Cable DHCP Leasequery feature is enabled, the Cisco CMTS allows only a certain number of DHCP leasequery requests for each service ID (SID) on an interface within the configured interval time period. If an SID generates more Leasequeries than the maximum, the router drops the excess number of requests until the next interval period begins.

You can configure both the number of allowable DHCP leasequery requests and the interval time period, so as to match the capabilities of your DHCP server (or configured alternate server) and cable network.

To configure the Cisco CMTS router to send DHCP leasequery requests to the DHCP server, use the **cable source-verify dhcp** and **no cable arp** commands. Unknown IP addresses that are found in packets for customer premises equipment (CPE) devices that use the cable modems on the cable interface are verified. The DHCP server returns a DHCP ACK message with the DHCP relay information and lease information of the CPE device that has been assigned this IP address, if any.

When **cable source-verify dhcp** and **no cable arp** commands are configured, DHCP leasequery is sent for downstream packets to verify unknown IP addresses within the IP address range configured on the cable bundle interface.

For DHCP leasequery to work in the downstream direction, the Cisco Network Registrar (CNR) should be made aware of the DHCP Option 82. This is required to make the CMTS map the CPE IP address to the correct CM. To do this, configure the **ip dhcp relay information option** command on the bundle interface to insert service class relay agent option into the DHCP DISCOVER messages. When the configuration is in place, during DHCP DISCOVER the values of DHCP Option 82 is cached by the CNR and is returned to the CMTS on any subsequent DHCP leasequery for that IP address.

To configure the Cisco CMTS router to divert DHCP leasequery requests to a server other than the DHCP server, use the **cable source-verify dhcp server ipaddress** and **no cable arp** commands.

The Cisco CMTS supports two types of DHCP leasequery implementation, Cisco standard compliant DHCP leasequery and RFC 4388 standard compliant DHCP leasequery. These two standards differ mostly in the identifiers used to query or respond to the DHCP Server. You can choose between these two implementations depending on which standard is supported on your DHCP Server.

Use the **ip dhcp compatibility lease-query client** {**cisco** | **standard**} command to configure the Cisco CMTS in either Cisco mode or RFC 4388 standard mode.

# DHCP MAC Address Exclusion List

This feature enables the ability to exclude trusted MAC addresses from the standard DHCP source verification checks for the Cisco CMTS. The DHCP MAC Address Exclusion List feature enables packets from trusted MAC addresses to pass when otherwise packets would be rejected with standard DHCP source verification. This feature overrides the cable source-verify command on the Cisco CMTS for the specified MAC address, yet maintains overall support for standard and enabled DHCP source verification processes. This feature is supported on the Performance Routing Engine 1 (PRE1), PRE2, and PRE4 modules on the Cisco cBR router chassis.

To enable packets from trusted source MAC addresses in DHCP to pass without source verification checks, use the cable trust command in global configuration mode. To remove a trusted MAC address from the MAC exclusion list, use the no form of this command. Removing a MAC address from the exclusion list subjects all packets from that source to standard DHCP source verification.

For more information on the cable trust command, see the Cisco IOS CMTS Cable Command Reference Guide .

# Unitary DHCPv6 Leasequery

This feature supports unitary DHCPv6 leasequery protocol (RFC 5007) on the Cisco CMTS routers for upstream IPv6 source verification. This protocol verifies the authenticity of the IPv6 CPE behind a home or small office cable deployment.

If the IPv6 source verification fails on the router and the **cable ipv6 source-verify dhcp** and **no cable nd** commands are configured on the bundle interface or subinterface, the Cisco CMTS triggers a unitary DHCPv6 leasequery to the Cisco Network Registrar (CNR). If a valid leasequery response is received from the CNR, the Cisco CMTS adds the CPE to its subscriber database and allows future traffic for the CPE.

The primary use of the unitary DHCPv6 leasequery protocol on the Cisco CMTS router is to recover lost CPE data including the Prefix Delegation (PD) route. The IPv6 CPE data can be lost from the Cisco CMTS in several ways. For example, PD route loss can occur during a Cisco CMTS reload.

The unitary DHCPv6 leasequery protocol also supports the following:

- DHCPv6 leasequery protocol.
- Rogue client database for failed source-verify clients.
- DHCPv6 leasequery filters.
- DHCPv6 leasequeries to a specific DHCPv6 server.

# How to Configure Filtering of Cable DHCP Leasequery Requests

Use the following procedures to configure the filtering of DHCP Leasequery requests on the Cisco CMTS downstreams and upstreams:

## Enabling DHCP Leasequery Filtering on Downstreams

Use the following procedure to start filtering DHCP leasequeries on all downstreams of a cable interface.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **cable source-verify leasequery-filter downstream** *threshold interval*<br>**Example:**<br>`Router(config)# cable source-verify leasequery-filter downstream 5 10` | Enables leasequery filtering on all downstreams on the specified bundle interface, using the specified *threshold* and *interval* values. |
| **Step 4** | **end**<br>**Example:**<br>`Router(config)# end` | Exits configuration mode and returns to privileged EXEC mode. |

## Enabling DHCP Leasequery Filtering on Upstreams

Use the following procedure to start filtering DHCP Leasequeries on all upstreams on a bundle interface.

### Procedure

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

| | Command or Action | Purpose |
|---|---|---|
| | `Router> enable` | |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface bundle** *bundle-no*<br>**Example:**<br>`Router(config)# interface bundle 1` | Enters interface configuration mode for the specified bundle interface. |
| **Step 4** | **cable source-verify leasequery-filter upstream** *threshold interval*<br>**Example:**<br>`Router(config-if)# cable source-verify leasequery-filter upstream 2 5` | Enables leasequery filtering on all upstreams on the specified bundle interface, using the specified *threshold* and *interval* values.<br>**Note** The **cable source-verify leasequery-filter upstream** command can only be configured under bundle interface.<br>**Note** Repeat step 3 and step 4 to enable the filtering of DHCP Leasequeries on the upstreams for other bundle interfaces. Primary and secondary interfaces in a cable bundle must be configured separately. |
| **Step 5** | **end**<br>**Example:**<br>`Router(config-if)# end` | Exits interface configuration mode and returns to privileged EXEC mode. |

## Configuring Unitary DHCPv6 Leasequery Filtering

Use the following procedure to configure the Cisco CMTS router to send Leasequeries to a DHCP server to verify the authenticity of the IPv6 CPE. You can also enable filtering of these requests to prevent large volumes of Leasequery requests on the bundle interfaces. Similarly, the number of allowable Leasequery requests and the interval time period can also be configured.

**Note** When the leasequery timer expires, only the IPv4 static CPE is automatically removed from the host database.

### Before you begin

- Disable the IPv6 Neighbor Discovery (ND) Gleaning feature using the **no** form of the **cable nd** command in bundle interface configuration mode before configuring the unitary DHCPv6 leasequery protocol. For details on IPv6 ND gleaning, see IPv6 on Cable feature guide.
- Configure the **cable ipv6 source-verify dhcp** command under the Cisco CMTS bundle or bundle subinterface to enable the unitary DHCPv6 leasequery protocol.
- Use the **cable ipv6 source-verify dhcp** [**server** *ipv6-address]* command for a single DHCP server.
- *Use the* **cable ipv6 source-verify dhcp command without any keywords** *for multiple DHCP servers.*

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface bundle** *bundle-no*<br>**Example:**<br>`Router(config)# interface bundle 1` | Enters interface configuration mode for the specified bundle interface. |
| **Step 4** | **cable ipv6 source-verify** or**cable ipv6 source-verify dhcp** [server ipv6-address]<br>**Example:**<br>`Router(config-if)# cable ipv6 source-verify`<br>`or`<br>`Router(config-if)# cable ipv6 source-verify dhcp server 2001:DB8:1::1` | Enables leasequery filtering on the specified bundle interface and verifies the IP address with multiple DHCPv6 servers. or Enables leasequery filtering on the specified bundle interface and verifies the IP address with a specified DHCPv6 server. |
| **Step 5** | **cable ipv6 source-verify leasetimer** *value*<br>**Example:**<br>`Router(config-if)# cable ipv6 source-verify leasetimer 200` | Enables leasequery timer on the specified bundle interface, for the Cisco CMTS to check its internal CPE database for IPv6 addresses whose lease time has expired. |
| **Step 6** | **cable ipv6 source-verify leasequery-filter** *threshold interval*<br>**Example:** | Enables filtering of the IPv6 leasequery requests. |

| | Command or Action | Purpose |
|---|---|---|
| | `Router(config-if)# cable ipv6 source-verify leasetimer 5 10` | |
| Step 7 | **end**<br>**Example:**<br>`Router(config-if)# end` | Exits interface configuration mode and returns to privileged EXEC mode. |

# Enabling DHCPv6 Leasequery Filtering on Downstreams

Use the following procedure to start filtering DHCP Leasequeries on all downstreams of a cable interface.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| Step 1 | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| Step 2 | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| Step 3 | **cable ipv6 source-verify leasequery-filter downstream** *threshold interval*<br>**Example:**<br>`Router(config-if)# cable ipv6 source-verify leasetimer 5 10` | Enables leasequery filtering on all downstreams on the specified bundle interface, using the specified threshold and interval values: |
| Step 4 | **end**<br>**Example:**<br>`Router(config-if)# end` | Exits interface configuration mode and returns to privileged EXEC mode. |

# Configuration Examples for Filtering of DHCP Leasequery

This section provides the following examples on how to configure the DHCP leasequery filtering feature:

## Example: DHCP Leasequery Filtering

The following example shows an excerpt from a typical configuration of a bundle interface that is configured for filtering DHCP leasequery requests on both its upstream and downstream interfaces:

**Note** If an alternate server has been configured to receive leasequery requests, the **cable source-verify dhcp server** *ipaddress command* would display in place of the **cable source-verify dhcp** command below.

```
.
.
.
cable source-verify leasequery-filter downstream 5 20
.
.
.
interface bundle 1
.
.
.
 cable source-verify dhcp
 cable source-verify leasequery-filter upstream 1 5
 no cable arp
.
.
```

## Example: Unitary DHCPv6 Leasequery Filtering

The following example shows how to display the total number of DHCPv6 leasequery requests that have been filtered on the router in Cisco IOS Release 12.2(33)SCF1:

```
Router# show cable leasequery-filter
IPv4 Lease Query Filter statistics for Unknown Sid
  Requests Sent : 0 total. 0 unfiltered, 0 filtered
IPv6 Lease Query Filter statistics for Unknown Sid
  Requests Sent : 0 total. 0 unfiltered, 0 filtered
```

# Additional References

### 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.<br>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.<br>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password. | http://www.cisco.com/support |

# Feature Information for Cable DHCP Leasequery

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 7: Feature Information for Cable DHCP Leasequery*

| Feature Name | Releases | Feature Information |
|---|---|---|
| Cable DHCP leasequery | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on the Cisco cBR Series Converged Broadband Routers. |

CHAPTER 5

# DHCPv6 Bulk-Lease query

This document describes the Dynamic Host Configuration Protocol (DHCP) v6 Bulk-Lease query feature on the Cisco cable modem termination system (CMTS) router.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 8: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information About DHCPv6 Bulk-Lease Query

This document describes the Dynamic Host Configuration Protocol (DHCP) v6 bulk-lease query feature on the Cisco cable modem termination system (CMTS) router. Cisco cBR-8 supports DHCPv6 bulk lease query in accordance with RFC 5460.

DHCPv6 bulk lease query is used to recover the IPv6 bindings(GUA/PD/LLA) for CPE ONLY after a chassis reload. The CPE’s IPv6 binding information is retrieved from the DHCPv6 server. On cBR8,this feature is designed to be started some time later after a chassis reload. Since before recover CPE’s information, we must wait its CM online firstly.

cBR8 will check all active CPE IPv6 bindings retrieved from DHCPv6 server one by one:

- If the IPv6 binding is for CM, cBR8 will drop it.
- If the IPv6 binding is for a CPE. If this binding does not present on cBR8 side and the CM of the CPE is online, cBR8 will recover the IPv6 binding, otherwise, cBR8 will drop it.

The DHCPv6 bulk-lease query feature is disabled by default.

To make this feature works, the DHCPv6 server side must also support DHCPv6 bulk lease query.

# How to Configure DHCPv6 Bulk-Lease Query

The following steps help you to setup the DHCPv6 Bulk-Lease query feature with Cisco cBR-8 Converged Broadband Router.

To enable the DHCPv6 Bulk-Lease query feature, run the following commands:

1. Run the **[no] ipv6 dhcp-relay bulk-lease disable** command.
2. Run the **cable ipv6 source-verify bulk-lease [start <start-seconds> timeout <timeout-seconds>]** command.

**Note** The start-seconds means the time DHCPv6 Bulk-lease will be started after common reload. The start-seconds default value is 2400 seconds. The timeout-seconds means the max time that DHCPv6 Bulk-lease allowed to run. The timeout-seconds default value is 600 seconds.

Ensure that you enable both the Cisco common and Cable specific parts.

To disable the DHCPv6 Bulk-Lease query feature, run the following commands:

1. Run the **ipv6 dhcp-relay bulk-lease disable** command.
2. Run the **[no] cable ipv6 source-verify bulk-lease** command.

**Note** The DHCPv6 server needs to listen on the TCP port 547.

# Debugging DHCPv6 Bulk-Lease Query

The following debugging commands are supported for the DHCPv6 Bulk-Lease Query:

- **debug cable ipv6 bulk-lq**
- **debug ipv6 dhcp relay bulk-lease**

To check the results of DHCPv6 Bulk-Lease query, you can use the **debug cable ipv6 bulk-lq** command. See the following example:

```
Router# show cable ipv6 bulk-lq
CMTS DHCPv6 Bulk Lease Query Statistics:
```

```
Start time 1200 seconds after system up
End time 1500 seconds after system up
DHCPv6 Bulk Lease Query glean ready: 0
DHCPv6 Bulk Lease Query process created: 0
Time out happened: No
Total number of CM option received: 31
Total number of LLA received: 10
Total number of LLA recovered: 7
Total number of GUA received: 10
Total number of GUA recovered: 8
Total number of PD received: 0
Total number of PD recovered: 0
```

# Feature Information for DHCPv6 Bulk-Lease query

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 9: Feature Information for DHCPv6 Bulk-Lease query*

| Feature Name | Releases | Feature Information |
|---|---|---|
| DHCPv6 Bulk-Lease query | Cisco IOS XE Gibraltar 16.12.1 | This feature was integrated into Cisco IOS XE Gibraltar 16.12.1 on the Cisco cBR Series Converged Broadband Routers. |

CHAPTER 6

# Layer 3 CPE Mobility

Layer 3 CPE Mobility feature is introduced to allows the mobility CPE devices to move between cable modems with as less disruption of traffic as possible.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 10: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Prerequisites for Layer 3 CPE Mobility

No special equipment or software is needed to use the Layer 3 CPE Mobility feature.

# Restrictions for Layer 3 CPE Mobility

- Layer 3 CPE Mobility feature allows CPE devices to move only in the same bundle or sub-bundle interface.

- The IPv4 or IPv6 subnets that are configured with mobility must match with the IPv4 or IPv6 subnets already configured on bundle or sub-bundle interface. Otherwise, configuration will not be accepted and the following message will be displayed:

```
Please remove the previous online CPEs or reset CMs,
```

- If you remove the IPv4 or IPv6 address on bundle or sub-bundle interface, it also removes the relative mobility subnets at the same time.
- Multicast packets will not trigger the Layer 3 CPE Mobility feature.
- VRF configured under bundle or sub-bundle interface is not supported for CPE mobility feature.
- In Layer 3 CPE Mobility feature, the packet lost time period during mobility will be unpredictable, depending on how many CPE devices move at the same time and system loading conditions.
- For CPE devices, which have multiple IPv4 or IPv6 addresses, all of IPv4 or IPv6 addresses will be rebuilt with new source information.
- Layer 3 CPE Mobility may be failed during line card or SUP HA and the trigger upstream packet will be dropped.
- If CPE mobility is turned on, mobility behavior will become effective before cable Ipv4 or IPv6 source verify.
- If Layer 3 CPE Mobility is enabled, some of the security checks will be skipped for the mobility subnets to achieve faster movement of the CPE devices.

# Information About Layer 3 CPE Mobility

The Layer 3 CPE Mobility feature allows CPE devices to move from cable modem to other by trigger of any unicast upstream packets of IPv4 or IPV6.

Each cable modem would be situated at a business hotspot location and the CPE devices move from one business location to another, where the service provider is the same and the head end CMTS is the same. This mobility is allowed for selected IP subnets.

The IPv4 or IPv6 subnets that are configured with mobility must match with the IPv4 or IPv6 subnets already configured on bundle or sub-bundle interface. Otherwise, configuration will not be accepted and the following message will be displayed:

```
Please remove the previous online CPEs or reset CMs,
```

When you remove mobility subnets under bundle or sub-bundle interface. The following warning message will be displayed after mobility subnets is configured or removed.

```
Warning: Please remove the previous online CPEs or reset CMs, to make the mobility scope
change works for every device !!!
```

**Note** If you have enabled mobility configuration for a subnet, the existing online CPE devices will be updated to aware of the mobility subnets, and the CPU usage will rise up during that time. So it's better to configure the mobility subnets before CM and CPE come online.

Enabling the Layer 3 CPE Mobility feature may, in certain situations, cause excessive punted packets. By default, the Source-Based Rate-Limiting (SBRL) feature rate-limits these punted packets to avoid CPU overload.

## Benefits of Layer 3 CPE Mobility

The feature provides the movement of CPE devices from one cable modem to another without change in the IP address and the TCP or UDP sessions established are maintained.

# How to Configure Layer 3 Mobility

## Configuring CPE Mobility

This section describes how to enable mobility on a particular IP subnet on a interface or subinterface bundle.

### Before you begin

Mobility subnets should match the IPv4 or IPv6 address configured on the bundle or sub-bundle interface.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface bundle bundle number\| bundle-subif-number**<br>**Example:**<br>`Router(config)# interface bundle 1`<br>`or`<br>`Router(config)# interface Bundle 1.1` | Enters interface configuration or subinterface mode. |
| **Step 4** | **cable l3-mobility** *IP-address mask \| IPv6 prefix*<br>**Example:**<br>`Router(config-if)# cable l3-mobility 2001:DB:22:1::1/64`<br>**Example:**<br>`Router(config-subif)# cable l3-mobility 192.0.3.1 255.255.255.0`<br>**Example:** | Enables mobility for a particular IPv4 or IPv6 subnet.<br>**Note** This command can be configured on a interface or a subinterface bundle. |

| | Command or Action | Purpose |
|---|---|---|
| | `Router(config-subif)#cable l3-mobility 2001:DB:22:1::1/64` | |
| **Step 5** | **exit**<br>**Example:**<br>`Router(config-if)# exit` | Exits interface configuration mode. |

### What to do next

**Troubleshooting Tips**

If the mobility IP address does not match with the mobility subnet, the following warning message is displayed:

```
Mobility IP should match the IDB subnet!
```

If you remove the IPv4 or IPv6 address from the interface, the mobility scope is removed for the IP address and the following warning message is displayed.

```
IPv6 2001:DBB:3:111::1 removed from Mobility subnets on Bundle1
```

.

## Configure Source-Based Rate Limit (SBRL) for L3-mobility

This section describes how to configure Source-Based Rate Limit (SBRL) for the L3-mobility feature. This procedure is optional and if not configured, the default SBRL configuration will apply.

**Note** SBRL for L3-mobility is enabled by default, so this configuration is optional.

Subscriber-side SBRL has a global and per-punt-cause configuration. L3-mobility punts are only subject to the per-punt-cause configuration. Traffic streams are identified by hashing the punt-cause and the source-MAC-address. This value is used as the index for rate-limiting. There is no special processing for hash-collisions, so hash-colliding streams are treated as if they are the same stream.

The default rate for L3-mobility punts is 4 packets per second.

### Before you begin

**Note** All punted packets are subject to CoPP and the punt-policer.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **platform punt-sbrl subscriber punt-cause** *punt-cause* **rate** *rate*<br>**Example:**<br>`Router(config)# platform punt-sbrl subscriber punt-cause 99 rate 8` | Configures Subscriber-MAC-address SBRL. |
| **Step 4** | **exit**<br>**Example:**<br>`Router(config-if)# exit` | Exits global configuration mode. |

# Disabling CPE Mobility

This section describes how to disable mobility on a particular IP subnet.

### Before you begin

The CPE mobility should be enabled on a particular IP subnet before you complete this procedure.

### Procedure

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

| | Command or Action | Purpose |
|---|---|---|
| **Step 3** | **interface bundle** *bundle number* \| *bundle-subif-number*<br>**Example:**<br>`Router(config)# interface bundle 1`<br>`or`<br>`Router(config)# interface Bundle 1.1` | Enters interface configuration or subinterface mode. |
| **Step 4** | **no cable l3-mobility** *IP-address mask* \| *IPv6 prefix*<br>**Example:**<br>`Router(config-if)# cable l3-mobility 192.0.3.1 255.255.255.0`<br>`Router(config-if)# cable l3-mobility 2001:DB:22:1::1/64` | Disbles mobility for a particular IPv4 or IPv6 subnet.<br>**Note** This command can be configured on a interface or a subinterface bundle |
| **Step 5** | **exit**<br>**Example:**<br>`Router(config-if)# exit` | Exits interface configuration mode. |

## Verifying Layer 3 Mobility Configuration

To verify the layer 3 mobility configuration, use the **show cable bundle** command.

# Configuration Examples for Layer 3 Mobility

This section provides the following configuration examples:

## Example: Configuring CPE Layer 3 Mobility

The following example shows how to configure the layer 3 CPE mobility on a interface bundle:

```
Router#show running interface bundle 10
Building configuration...
Current configuration : 1247 bytes
!
interface Bundle10
ip address 192.0.3.1 255.255.255.0 secondary
ip address 192.2.21.1 255.255.255.0 secondary
ip address 192.3.23.1 255.255.255.0
ip pim sparse-dense-mode
ip igmp static-group 231.1.1.1
no cable arp filter request-send
no cable arp filter reply-accept
cable l3-mobility 192.0.3.1 255.255.255.0
cable l3-mobility 192.2.21.1 255.255.255.0
cable l3-mobility 192.3.23.1 255.255.255.0
cable l3-mobility 2001:DB:26:1::1/64
cable l3-mobility 2001:DB:27:1::1/96
cable dhcp-giaddr primary
cable helper-address 20.1.0.3
ipv6 address 2001:DB:26:1::1/64
ipv6 address 2001:DB:27:1::1/96
ipv6 enable
ipv6 nd reachable-time 3600000
ipv6 nd managed-config-flag
ipv6 nd other-config-flag
ipv6 dhcp relay destination 2001:DB:1:1:214:4FFF:FEA9:5863
end
```

## Example: Configuring SBRL for L3-mobility

The following example shows how SBRL is configured for L3-mobility:

```
Router# show run | i punt-sbrl
platform punt-sbrl subscriber punt-cause 99 rate 8
```

# Additional References

The following sections provide references related to Layer 3 CPE Mobility feature for the Cisco CMTS routers.

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for Layer 3 CPE Mobility

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 11: Feature Information for Layer 3 CPE Mobility*

| Feature Name | Releases | Feature Information |
|---|---|---|
| Layer 3 Mobility | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |

CHAPTER 7

# DOCSIS 3.0 Multicast Support

The Cisco cBR Series Routers support multicast improvements based on Data-over-Cable Service Interface Specifications (DOCSIS) 3.0. DOCSIS 3.0 multicast support improves bandwidth efficiency and allows service providers to offer differentiated quality of service for different types of traffic.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 12: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Prerequisites for the DOCSIS 3.0 Multicast Support

- DOCSIS 3.0-compliant Cisco CMTS and DOCSIS 3.0-enabled cable modems are required.
- Cisco CMTS must be MDF-enabled by default.
- Quality of service (QoS) parameters must be configured for various multicast sessions.

# Restrictions for the DOCSIS 3.0 Multicast Support

- You cannot disable explicit tracking.

- For multicast QoS, you must define three objects and templates, Service-Class, Group-QoS-Config (GQC), and Group-Config, and associate them to a particular bundle or forwarding interface.
- You must define a default service class and GQC before defining objects and templates.
- Static multicast feature is always enabled and you cannot disable it.
- The service flow attribute-based selection will be ignored if the group configuration is configured on the default forwarding interface.
- The multicast DSID feature is supported only on DOCSIS 3.0-compliant cable modems.
- The cable multicast mdf-disable wb-incapable-cm command disables multicast downstream service identifier (DSID) forwarding capability on the cable modem, which impacts the DSID capability between the Cisco CMTS and the cable modem.
- The multicast traffic to CPE increases two-fold after changing the multicast QoS configuration or the service-flow attribute during an active session. The traffic replication will continue till the default session timeout period (180 seconds). After the session timeout, the multicast DSID is removed from both Cisco CMTS and CM, and normal multicast traffic flow is resumed.
- For the DOCSIS 3.0 Multicast support feature to function properly, the CPE and the CM must be in the same virtual routing and forwarding (VRF) interface.

# Information About the DOCSIS 3.0 Multicast Support

IP multicast, an integral technology in networked applications, is the transmission of the same information to multiple recipients. Any network application, including cable networks, can benefit from the bandwidth efficiency of multicast technology. Two new technologies—Channel Bonding and Single Source Multicast (SSM)—are expected to dramatically accelerate multicast deployment.

The channel bonding and SSM technologies dramatically increase the operational efficiency of the existing hybrid fiber-coaxial (HFC) network. Using the multicast improvements, the cable operators can seamlessly deliver advanced services like video on demand (VoD), internet protocol television (IPTV), and facilitate interactive video and audio, and data services.

The following sections explain the benefits of DOCSIS 3.0 Multicast Support:

## Multicast DSID Forwarding

DOCSIS 3.0 multicast support introduces centralized control at the Cisco CMTS to provide flexibility and scalability to support a large array of multicast protocols. It replaces the Internet Group Management Protocol (IGMP), version 2 snooping infrastructure, which was part of the DOCSIS 1.1 and 2.0 models. Now, the Cisco CMTS allocates an unique Downstream Service Identifier (DSID) to identify every multicast stream. These DSIDs are sent to the CMs that use these DSIDs to filter and forward Multicast traffic to the CPEs.

The multicast DSID forwarding (MDF) provides the following benefits:

- Unique identification of packet stream across bonding group within a MAC domain.
- Designation of packet stream as either Any Source Multicast (ASM) or Source Specific Multicast (SSM) per multicast channel.
- Implementation of multicast DSID management on the Route Processor (RP) makes it operate on a standalone basis.

- Snooping of all upstream signal control packets by the Cisco CMTS to find the customer premises equipment (CPE) on the Multicast DSID-based Forwarding (MDF) enabled CM and allocates DSID from the pool.
- Transmission of allocated DSIDs to the CM through Dynamic Bonding Change (DBC) message.
- Reuse of DSIDs on other MDF-enabled CMs in the same bonding group, joining the multicast session.
- Removal of DSIDs from the CM through a DBC message by the Cisco CMTS after a multicast session leave event.
- Release of DSID to the pool by the Cisco CMTS when the last member leaves the bonding group.
- The following DSIDs are preallocated for each primary downstream (modular and integrated cable interfaces) to forward general query messages. These DSIDs form part of the multicast group signaling protocol. Other multicast groups, do no use these DSIDs.
  - IGMPv2 general query (IPv4)
  - IGMPv3 general query (IPv4)
  - MLDv1 general query (IPv6)
  - MLDv2 general query (IPv6)
  - Preregistration of DSID (IPv6)
- Allocation of DSID ensures traffic segregation between virtual private networks (VPNs) for DOCSIS 3.0 MDF-enabled CMs. For example, two clients from two VPNs joining the same multicast will get two distinct DSIDs.

## Multicast Forwarding on Bonded CM

Multicast packets to the DOCSIS 3.0-enabled CMs are transmitted as bonded packets with DSID extension header on the primary bonding group if the Secondary Multicast Bonding Group is disabled. Multicast packets for MDF-disabled or pre-DOCSIS 3.0 CMs are transmitted as non-bonded without DSID extension header. For more information on this feature, refer to Multicast Secondary Bonding Group, on page 103.

In a network, where only MDF-enabled or MDF-disabled CMs exist, the traffic is segregated using field types. The MDF-enabled CM forwards the frame with the field type and the MDF-disabled CM drops it. The DSID labeling ensures that MDF-enabled CM gets a copy of the multicast session to prevent “cross talk”.

For hybrid CMs (MDF-enabled and MDF-disabled CMs) that do not support field type forwarding, you should configure per session encryption or security association identifier (SAID) isolation to ensure traffic segregation. DOCSIS 3.0 mandates that if the hybrid CM fails to forward field type frames, the Cisco CMTS should employ multicast security association identifier (MSAID) isolation. This isolation is achieved by assigning different MSAID to each replication, one to bonded CM and another to the non-bonded or hybrid CM. This helps to prevent CMs from receiving duplicate traffic.

## Static TLV Forwarding

As per DOCSIS 3.0 specifications, the Cisco CMTS must support Static Multicast. When the CM tries to register with the Cisco CMTS, the Cisco CMTS checks whether Static Multicast Encoding is present in the CM configuration file. If the Static Multicast Encoding is present, the Cisco CMTS sends a DSID corresponding to each Static Multicast channel in the Registration-Response (REG-RSP) message.

The Multicast DSID management is located at Supervisor and the interface card has to contact the Supervisor for proper DSID assignment. The interface card also caches the response from Supervisor to eliminate the need to communicate to the Supervisor for subsequent Static Multicast encoding.

## Explicit Tracking

The Cisco CMTS can perform explicit tracking with IGMPv3 support. The IGMPv3 removes the report suppression feature associated with the IGMPv2 specification enabling the Cisco CMTS to get the complete information on session and host information. This benefits the IGMP Fast Leave processing and DSID management for each CM.

A host or session database is used to track hosts (IP/MAC) joining a particular multicast session. From the host, you can track the CM based on the SID and cable downstream interface. This database also helps to determine whether the Cisco CMTS should remove the DSID from a particular CM when the multicast session is over.

## Multicast Quality of Service Enhancement

DOCSIS 3.0 mandates that the CMTS should not admit any flow exceeding the session limit. Though the current Multicast QoS (MQoS) session limit admits the session, it fails to provide any QoS for sessions exceeding the session limit.

**Note** Multicast packets are sent using the default Group Service Flows (GSF) when the Multicast QoS feature is disabled.

As part of DOCSIS 3.0 requirements for Multicast QoS, Group Classifier Rules (GCR) is supported. The Cisco CMTS determines the set of Group Configurations (GCs) whose session range matches the multicast group address. For SSM, the source address is also used to identify the matching GCs. A GCR is created for each matching GC and linked to the multicast session. The GCR is assigned also with an unique identifier, SAID, and Group Service Flow (GSF).

The following conditions are used to select the GC entries:

- The GC entry with the highest rule priority is selected, if more than one GC entry matches.
- All matching GC entries are selected, when multiple GCs have the same highest rule priority.

The GCR classification is done based on type of service (TOS) fields. The TOS specifier in the GCR is used to choose the correct GCR when multiple GCRs match a single multicast session.

**Note** When two multicast group configurations (GCs) have the same session range and configuration (under global or bundle configuration), then the same forwarding interface selection is not guaranteed.

Non-IP multicasts and broadcast packets use GSF. They are similar to individual service flows and are shared by all the CMs on a particular Digital Command Signal (DCS) matching the same GCR. A single GSF is used for multicast sessions matching different GCs using the same aggregate GQC.

## Multicast Secondary Bonding Group

The DOCSIS 3.0-compliant CM can receive multicast packets from non-primary (or bonded) channels using the MDF support at the CMTS.

The multicast secondary bonding group is defined as a shared bonding group or RF channel that feeds more than one fiber node through an optical split. This allows CMs from different primary bonding groups and channels to listen to one or more shared sets. The multicast packets are replicated only to the shared downstream channel set, which helps conserve the downstream bandwidth.

DOCSIS 3.0 defines attribute-based service flow creation, which allows the Cisco CMTS to make more “intelligent” decisions on the selection of bonding group or individual channel for unicast and multicast forwarding.

The Multicast Secondary Bonding Group provides the following benefits:

- New MQoS and attribute-based forwarding for Multicast Secondary Bonding Group.
- The primary downstream interface acts as a forwarding interface for narrowband CMs.
- The following algorithm is used to select a forwarding interface for wideband CMs:
    - A primary bonding group is selected if a group-config matching the session is present in it. MQoS parameters are taken from the group-config.
    - A primary bonding group is selected if a group-config is not present at the bundle level or at the global level.
    - A group-config found at the bundle level or global level is used to find the Group-QoS-Config (GQC) and eventually the attribute and forbidden bit-masks, which are then used to find the interface.
    - All Wideband Cable Modems (WCMs) in a bundle use the same secondary bonding group if a bundle-level group-config or global-level group-config is configured.
- The IGMP report ignores a source if the given source address fails to find a matching interface.
    - If a matching interface is found, that interface is used for forwarding and the MQoS parameters are taken from the matching group-config from the forwarding interface or bundle interface or global level.
    - If a matching interface is not found, then the IGMP report is ignored.
- For a static join, attribute-based forwarding is not supported, and only the primary downstream is used.

# Load Balancing

The Load Balancing feature does not load balance a CM while a multicast stream is going on for that particular CM. It utilizes the Explicit Tracking Database, which holds complete information on the CM subscription to achieve this.

# Multicast DSID Forwarding Disabled Mode

For any application that needs the cable modem to perform IGMP snooping, the MDF on the cable modem must be disabled. Cable modems registered in MDF-enabled mode by the Cisco CMTS do not perform IGMP snooping because MDF forwarding is based on DSID filtering. The **cable multicast mdf-disable** command disables the MDF capability on the cable modem.

This command is configured on the route processor and is downloaded to the cable line card via the configuration update. The configuration does not change the Cisco CMTS forwarding mechanism or DSID allocation. The Cisco CMTS allocates the DSID and the multicast packet is encapsulated with the DSID header. This does not affect traffic forwarding on the MDF-disabled cable modem. According to DOCSIS3.0 specification, pre-DOCSIS2.0 or MDF-disabled cable modems ignore the DSID header and continue multicast forwarding based on the Group Media Access Control (GMAC) from IGMP snooping. When the cable modem runs in MDF-disabled mode, only IGMPv2 is supported and the Cisco CMTS drops IGMPv3 and MLD messages.

Multicast encryption based on BPI+ is not supported on non-MDF cable modems, if IGMP SSM mapping is used. A non-MDF cable modem is either a pre-DOCSIS 3.0 cable modem or a DOCSIS 3.0 cable modem running in MDF-disabled mode.

## MDF1 Support for DOCSIS 2.0 Hybrid Cable Modems

The Cisco CMTS router enables MDF capability for DOCSIS 2.0 hybrid cable modems, IPv6, and other cable modems that advertise MDF capability to allow IPv6 packet forwarding. The **wb-incapable-cm** keyword in the **cable multicast mdf-disable** command disables MDF on all DOCSIS 2.0 hybrid cable modems including DOCSIS Set-Top Gateway (DSG) hybrid embedded cable modems to support IGMP snooping.

## DSG Disablement for Hybrid STBs

The **cable multicast mdf-disable** command with the **wb-incapable-cm** keyword prevents all DOCSIS 2.0 DSG embedded cable modems from receiving DSG multicast traffic besides disabling MDF support.

The **wb-incapable-cm** keyword disables MDF capability only on non-DSG DOCSIS 2.0 hybrid cable modems. To disable MDF capability on all DSG embedded cable modems (DOCSIS 3.0 DSG and DOCSIS 2.0 DSG hybrid), a new keyword, DSG, is introduced.

**Note** After disabling MDF capability, you must run **clear cable modem reset** command to bring all DSG embedded cable modems online.

## Benefits of MDF1 Support

- Supports IPv6 on different known cable modem firmware types.
- Disables the MDF capability on the Cisco CMTS.
- Supports In-Service Software Upgrade (ISSU) and line card high availability.

# Dynamic Multicast Replication Sessions

When users enable IPTV service on the Cisco cBR routers, to enhance the performance, the following features are supported on Cisco cBR.

- Supports 8000 SIDs per bundle interface:

  The Cisco cBR supports 8000 SIDs per bundle, because each MQoS need one SID for each multicast session.

- Provides faster and efficient IP Communicator messages.

- Provides faster multicast forwarding.

- Enables caching of dynamic multicast sessions.

## Cache Multicast Replication Sessions

Creating a new multicast replication session takes most of the CPU cycles when compared to joining an existing multicast replication session. Most resources associated with a multicast replication session can be cached after the session ends.

Hence, when a new IGMP join request is received later, these resources can be reused.

The multicast session replication cache is available only on an active SUP. When SUPSO happens, all cached sessions are lost, and are then recreated on the new active SUP when an IGMP/MLD join request is received.

When LCSO happens, all cache sessions of this LC are cleared and are recreated on the new active LC when an IGMP/MLD join request is received.

# How to Configure the DOCSIS 3.0 Multicast Support

This section describes the following tasks that are required to implement DOCSIS 3.0 Multicast Support on Cisco CMTS Routers:

## Configuring Basic Multicast Forwarding

To configure a basic multicast forwarding profile that can be applied to a DOCSIS 3.0 multicast configuration, use the **ip multicast-routing** command. You must configure a multicast routing profile before you can proceed with a multicast group.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **IP multicast-routing** [**vrf**]<br>**Example:**<br>`Router(config)# IP multicast-routing vrf` | Enables multicast routing globally or on a particular virtual routing and forwarding (VRF) interface. |
| **Step 4** | **interface bundle** *number*<br>**Example:**<br>`Router(config)# interface bundle 1` | Configures the interface bundle and enters interface configuration mode. |
| **Step 5** | **IP pim sparse-mode**<br>**Example:** | Configures sparse mode of operation. |

| | Command or Action | Purpose |
|---|---|---|
| | `Router(config-if)# IP pim sparse-mode` | **Note** The Cisco CMTS router must have a Protocol Independent Multicast (PIM) rendezvous point (RP) configured for the PIM sparse mode. The Supervisor is configured using the **ip pim rp-address** command or Auto-Supervisor configuration protocol. |
| **Step 6** | **IP pim sparse-dense-mode**<br>**Example:**<br>`Router(config-if)# IP pim sparse-dense-mode` | Configures the interface for either sparse mode or dense mode of operation, depending on the mode in which the multicast group is operating. |
| **Step 7** | **IP igmp version version-number**<br>**Example:**<br>`Router(config-if)# IP igmp version 3` | Configures the interface to use IGMP version 3. |
| **Step 8** | **IP igmp v3-query-max-response-time** *response_time*<br>**Example:**<br>`Router(config-if)# IP igmp v3-query-max-response-time 500` | Configures the maximum query response time for igmp version 3. |

# Configuring Multicast DSID Forwarding

The multicast DSID forwarding is enabled by default. You cannot configure this feature.

# Configuring Explicit Tracking

The Explicit Tracking feature is enabled by default. You cannot configure it.

# Configuring Multicast QoS

To configure a Multicast QoS profile that can be applied to a DOCSIS 3.0 configuration, use the **cable multicast group-qos** command. You must configure a Multicast QoS profile before you can add a Multicast QoS profile to a QoS multicast group.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **configureterminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **cable service class** *class-index* **name** *service-class-name*<br>**Example:**<br>`Router(config)# cable service class 1 name MQOS_DEFAULT` | Configures the name of the cable service class. |
| **Step 4** | **cable service class** *class-index* **downstream**<br>**Example:**<br>`Router(config)# cable service class 1 downstream` | Configures the downstream for the cable service class. |
| **Step 5** | **cable service class** *class-index* **max-rate** *maximum-bandwidth-allowed*<br>**Example:**<br>`Router(config)# cable service class 1 max-rate 10000000` | Configures the maximum allowed bandwidth for the cable service class. |
| **Step 6** | **cable service class** *class-index* **min-rate** *cir*<br>**Example:**<br>`Router(config)# cable service class 1 min-rate 1000000` | Configures the minimum committed information rate for the cable service class. |
| **Step 7** | **cable multicast group-qos default scn** *service-class-name* **aggregate**<br>**Example:**<br>`Router(config)# cable multicast group-qos default scn MQOS_DEFAULT aggregate` | Specifies the default service class name for the QoS profile. |
| **Step 8** | **cable multicast qos group** *number* **priority** *value*<br>**Example:**<br>`Router(config)# cable multicast qos group 20 priority 1` | Configures a multicast QoS group and enters multicast QoS configuration mode, and specifies the priority of the cable multicast QoS group. |
| **Step 9** | **application-id** *app-id*<br>**Example:**<br>`Router(config-mqos)# application-id 10` | Specifies the application identification number of the multicast QoS group. This value is configured to enable admission control to the multicast QoS group. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 10** | **session-range ip-address ip-mask**<br>**Example:**<br>`Router(config-mqos)# session-range 230.0.0.0 255.0.0.0` | Specifies the session range IP address and IP mask of the multicast QoS group. You can configure multiple session ranges. |
| **Step 11** | **cable multicast qos group** *number priority value* [**global**]<br>**Example:**<br>`Router(config)#cable multicast qos group 20 priority 63 global` | Specifies the multicast QoS group identifier. |

# Selecting a Forwarding Interface Based on Service Flow Attribute

The Service Flow Attribute feature allows a bonded CM to listen to multiple bonding groups, and using the interface-specific bit-masks, the CM can select the best route to receive multicast traffic.

The Service Flow Attribute feature allows selection of a forwarding interface based on the DOCSIS 3.0 construct named “service flow attribute mask.” Every interface has an attribute bit-mask depicting attributes of that interface. The multicast service class specified in the group QoS configuration contains required and forbidden attribute bit-masks. If a bonded CM can listen to multiple bonding groups (wideband interfaces), using specific bit-masks in the service class as well as on the bonding group, then one of these bonding groups can be selected for forwarding of multicast traffic.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **cable service class** *class-index* **name** *name*<br>**Example:**<br>`Router(config)# cable service class 10 name mcast10` | Configures the service class name. |
| **Step 4** | **cable service class** *class-index* **downstream**<br>**Example:** | Configures the downstream for the selected service class. |

| | Command or Action | Purpose |
|---|---|---|
| | `Router(config)# cable service class 10 downstream` | |
| **Step 5** | **cable service class** *class-index* **max-rate** *maximum-rate*<br>**Example:**<br>`Router(config)# cable service class 10 max-rate 1000000` | Configures the maximum rate for the selected service class. |
| **Step 6** | **cable service class** *class-index* **min-rate** *minimum-rate*<br>**Example:**<br>`Router(config)#  cable service class 10 min-rate 100000` | Configures the minimum rate for the selected service class. |
| **Step 7** | **cable service class** *class-index* **req-attr-mask** *required-attribute-mask*<br>**Example:**<br>`Router(config)# cable service class 10 req-attr-mask 8000000F` | Configures the required attribute mask for the selected service class. |
| **Step 8** | **cable service class** *class-index* **forb-attr-mask** *forbidden-attribute-mask*<br>**Example:**<br>`Router(config)# cable service class 10 forb-attr-mask 7FFFFFF0` | Configures the forbidden attribute mask for the selected service class name. |
| **Step 9** | **cable multicast group-qos** *number* **scn** *service-class-name* **aggregate**<br>**Example:**<br>`Router(config)# cable multicast group-qos 1 scn 10 mcast10 aggregate` | Configures the cable multicast group QoS identifier, service class name, and multicast value. |
| **Step 10** | **cable multicast qos group** *group* **priority** *priority*<br>**Example:**<br>`Router(config)# cable multicast qos group 1 priority 1` | Configures the cable MQoS group and enters MQoS configuration mode. |
| **Step 11** | **session-range** *session-range mask*<br>**Example:**<br>`Router(config-mqos)# session-range 230.1.1.1 255.255.255.255` | Specifies session range. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 12** | **group-qos** *qos*<br>**Example:**<br>`Router(config-mqos)# group-qos 1` | Specifies the group QoS. |
| **Step 13** | **exit**<br>**Example:**<br>`Router(config-mqos)# exit` | Returns to global configuration mode. |
| **Step 14** | **interface bundle** *number*<br>• **ip address** *ip mask*<br>• **ip pim sparse-mode**<br>• **ip helper-address** *helper-address*<br>• **cable multicast-qos group** *group*<br>**Example:**<br>`Router(config)# interface Bundle1`<br>`Router(config-if)#ip address 40.1.1.1 255.255.255.0`<br>`Router(config-if)#ip pim sparse-mode`<br>`Router(config-if)#ip helper-address 2.39.16.1`<br>`Router(config-if)#cable multicast-qos group 1` | Configures the interface bundle with the IP address, helper address, and MQoS group. |
| **Step 15** | **exit**<br>**Example:**<br>`Router(config-if)# exit` | Returns to global configuration mode. |
| **Step 16** | **interface wideband-cable** *slot*/*subslot*/*port*:*wideband-channel*<br>• **description** *description*<br>• **cable bundle** *number*<br>• **cable rf-channel channel-list** *grouplist* **bandwidth-percent** *bw-percent*<br>• **cable downstream attribute-mask attribute-***mask*<br>**Example:**<br>`Router(config)# interface Wideband-Cable1/0/0:0`<br>`Router(config-if)# description cable rf-channels channel-list 0-7 bandwidth-percent 20`<br>`Router(config-if)# cable bundle 1`<br>`Router(config-if)# cable rf-channels channel-list 0-7 bandwidth-percent 20`<br>`Router(config-if)# cable downstream attribute-mask 8000000F` | Selects the interface for forwarding based on the bit-masks specified in the service class and on the wideband interface. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 17** | **end**<br>**Example:**<br>`Router(config-if)# end` | Returns to privileged EXEC mode. |

# Configuring Multicast DSID Forwarding Disabled Mode

To disable MDF on the cable modem, use the **cable multicast mdf-disable** command in global configuration mode.

**Note** Multicast encryption based on BPI+ is not supported on non-MDF cable modems, if IGMP SSM mapping is used.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>- Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **cable multicast mdf-disable [wb-incapable-cm]**<br>**Example:**<br>`Router(config)# cable multicast mdf-disable` | Disables MDF capability on the cable modem. |
| **Step 4** | exit<br>**Example:**<br>`Router(config)# exit`<br>`Router#` | Exits the global configuration mode. |

# Configuring Multicast Replication Session Globally

Use the following command to configure the maximum number of multicast replication sessions globally and the value is configured per L2 forwarding interface.

If the operator does not configure a value for the maximum number, by default, is set to 0 for all L2 forwarding interfaces, and the cache function is not valid. Cisco cBR does not cache the multicast replication sessions.

If the value is changed from a number such as 10 to 0, all current caches is cleared. The value range is from 0 to 500.

The following example shows how to set the maximum number of cache to 0:

```
enable
configure terminal
cable multicast ses-cache 0
```

The following example shows how to change the current value:

```
enable
configure terminal
 [no|default] cable multicast ses-cache <0-500>
```

# Configuring Multicast Replication Sessions on Forwarding Interface

Use the following command to enable the multicast replication session on each L2 forwarding interface.

The value range for the maximum number is 0 to 500. If the value is changed from a number such as 10 to 0, all current caches is cleared.

The configured value for the interface has higher priority than the system value. The following example shows how to configure session cache on forwarding interface and make Cisco cBR use the system values:

```
enable
configure terminal
interface wideband-Cable {slot /subslot /controller :wideband-channel}
[no|default] cable multicast ses-cache
```

The following example shows how to set the maximum number of cache for the interface:

```
enable
configure terminal
interface integrated-Cable {slot/subslot/port:rf-channel}
cable multicast ses-cache 500
```

The following example shows how to configure a value 0 for an interface:

```
enable
configure terminal
interface integrated-Cable {slot/subslot/port:rf-channel}
no cable multicast ses-cache
```

# Clearing Multicast Replication Cache

Use the following command to clear the multicast replication session for all or for a specific L2 forwarding interface. The system deletes all current cache entries for all L2 forwarding interfaces or for a specific L2 interface.

```
enable
clear cable multicast ses-cache [interface xxx | all | counter]
```

# How to Monitor the DOCSIS 3.0 Multicast Support

To monitor the DOCSIS 3.0 Multicast Support feature, use the following procedures:

## Verifying the Basic Multicast Forwarding

To verify the configuration parameters for basic multicast forwarding, use the **show ip mroute** command as shown in the following example:

```
Router# show ip mroute

IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       V - RD & Vector, v - Vector
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 230.1.1.1), 00:00:03/00:02:55, RP 30.1.1.1, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Bundle1, Forward/Sparse, 00:00:03/00:02:55, H
(*, 224.0.1.40), 00:12:02/00:02:19, RP 30.1.1.1, flags: SJCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Bundle1, Forward/Sparse, 00:12:02/00:02:19
```

To verify the multicast information for the specified virtual interface bundle, based on IGMPv3, use the **show cable bundle multicast** command as shown in the following example:

```
Router# show cable bundle 1 multicast

CableBundle Interface Source IP  Multicast IP    MAC Address
1    Bundle1.1 *       230.1.1.1    0100.5e00.0001
```

To verify the MAC forwarding table for the specified virtual interface bundle, based on IGMPv3, use the **show cable bundle forwarding** command as shown in the following example:

```
Router# show cable bundle 1 forwarding

MAC address Interface Flags Location link sublink
00c0.5e01.0203 Cable8/0/0 3 64E5BF60 0 64E5BE00
00c0.5e01.0203 Cable7/0/0 3 64E5BE00 0 0
00c0.5e01.0101 Cable8/0/0 3 64E5BEE0 0 64E5BE40
```

# Verifying the Multicast DSID Forwarding

To verify the entire DSID database content, use the **show cable multicast dsid** command as shown in the following example:

```
Router# show cable multicast dsid
Multicast Group    : 230.1.2.3
         Source    : *
         IDB       : Bu2         Interface: Mo1/1/0:0   Dsid: 0x1F078
         StatIndex : 2           SAID: DEFAULT
Multicast Group    : 230.1.2.3
         Source    : *
         IDB       : Bu2         Interface: Mo1/1/0:0   Dsid: 0x1F078
         StatIndex : 3           SAID: 8196
Multicast Group    : 230.1.2.3
         Source    : *
         IDB       : Bu2         Interface: Mo1/1/0:0   Dsid: 0x1F078
```

**StatIndex : 4 SAID: 8197**

To verify the entire database content, use the **show cable multicast db** command as shown in the following example:

Router# **show cable multicast db**

```
interface : Bundle1
Session (S,G) : (*,230.1.1.1)
Fwd Intfc Sub Intfc Host Intfc CM Mac Hosts
Wi1/1/0:0 Bundle1 Ca5/0/0 0018.6852.8056 1
```

To verify the information for the registered and unregistered CMs, use the **show cable modem verbose** command as shown in the following example:

Router# **show cable modem 0010.7bb3.fcd1 verbose**

```
MAC Address : 00C0.7bb3.fcd1
IP Address : 10.20.113.2
Prim Sid : 1
QoS Profile Index : 6
Interface : C5/0/U5
sysDescr : Vendor ABC DOCSIS 2.0 Cable Modem
Upstream Power : 0 dBmV (SNR = 33.25 dBmV)
Downstream Power : 0 dBmV (SNR = ----- dBmV)
Timing Offset : 1624
Initial Timing Offset : 2812
Received Power : 0.25
MAC Version : DOC1.0
Qos Provisioned Mode : DOC1.0
Enable DOCSIS2.0 Mode : Y
Phy Operating Mode : atdma
Capabilities : {Frag=N, Concat=N, PHS=N, Priv=BPI}
Sid/Said Limit : {Max Us Sids=0, Max Ds Saids=0}
Optional Filtering Support : {802.1P=N, 802.1Q=N}
Transmit Equalizer Support : {Taps/Symbol= 0, Num of Taps= 0}
Number of CPE IPs : 0(Max CPEs = 1)
CFG Max-CPE : 1
Flaps : 373(Jun 1 13:11:01)
Errors : 0 CRCs, 0 HCSes
Stn Mtn Failures : 0 aborts, 3 exhausted
Total US Flows : 1(1 active)
Total DS Flows : 1(1 active)
Total US Data : 1452082 packets, 171344434 bytes
Total US Throughput : 0 bits/sec, 0 packets/sec
Total DS Data : 1452073 packets, 171343858 bytes
Total DS Throughput : 0 bits/sec, 0 packets/sec
Active Classifiers : 0 (Max = NO LIMIT)
DSA/DSX messages : reject all
Dynamic Secret : A3D1028F36EBD54FDCC2F74719664D3F
Spoof attempt : Dynamic secret check failed
Total Time Online : 16:16
```

# Verifying the Explicit Tracking Feature

To verify explicit tracking information, use the **show cable multicast db** command as shown in the following example:

Router# **show cable multicast db**

```
Interface : Bundle1
Session (S,G) : (*,230.1.1.1)
Fwd Intfc  Sub Intfc  Host Intfc   CM Mac           Hosts
Mo1/1/0:0  Bundle1    Ca5/0/0      0018.6852.8056       1
```

# Verifying the Multicast QoS Feature

To verify the cable MQoS details, use the **show cable multicast qos** commands as shown in the following example:

```
Router# show cable multicast qos ?
group-config Display Multicast Group Config information
group-encryption Display Multicast Group Encryption information
group-qos Display Multicast Group QOS information
Router# show cable multicast qos group-config
Multicast Group Config 1 : Priority 1
Group QOS - 1
Group Encryption - 1
Session Range - Group Prefix 230.0.0.0 Mask 255.0.0.0 Source Prefix 0.0.0.0 Mask 0.0.0.0
Router# show cable multicast qos group-encryption
Multicast Group Encryption 1 : Algorithm 56bit-des
Router# show cable multicast qos group-qos
Group QOS Index Service Class Control Igmp Limit Override
DEFAULT MQOS_DEFAULT Aggregate NO-LIMIT 1 MQOS Aggregate NO-LIMIT
```

To verify the DOCSIS service flows on a given cable interface, use the **show interface service-flow** command as shown in the following example:

```
Router# show interface cable 6/0 service-flow

Sfid  Sid   Mac Address     QoS Param Index Type    Dir    Curr   Active
BG/CH
                             Prov  Adm  Act                 State  Time
4     8193  ffff.ffff.ffff   3     3    3   sec(S)  DS     act    21h57m
5     8196  ffff.ffff.ffff   4     4    4   sec(S)  DS     act    00:17
```

# Verifying the Service Flow Attributes

To verify the configuration of service flow attributes on the service class configuration, use the show cable service-class verbose command as shown in the following example:

```
Router# show cable service-class 10 verbose
Index:                                    10
Name:                                     mcast10
Direction:                                Downstream
Traffic Priority:                         0
Maximum Sustained Rate:                   1000000 bits/sec
Max Burst:                                3044 bytes
Minimum Reserved Rate:                    1000000 bits/sec
Minimum Packet Size                       0 bytes
Admitted QoS Timeout                      200 seconds
Active QoS Timeout                        0 seconds
Required Attribute Mask                   8000000F
Forbidden Attribute Mask                  7FFFFFF0
Scheduling Type:                          Undefined
Max Latency:                              0 usecs
Parameter Presence Bitfield:              {0x3148, 0x0}
```

To verify the configuration of SF attributes on the Wideband interface configuration, use the **show running-config interface** command as shown in the following example:

```
Router# show running-config interface Wideband-Cable 1/0/0:2
interface Wideband-Cable1/0/0:2
 cable bundle 1
 cable bonding-group-id 3
 cable rf-channel 3
 cable downstream attribute-mask 8000000F
end
```

# Verifying the Multicast Group Classifiers

To verify the details of the Group Classifier Rule, use the **show interface wideband-cable multicast-gcr** command as shown in the following example:

```
Router# show interface wideband-cable 1/1/0:0 multicast-gcr
Group Classifier Rules on Wideband-Cable1/1/0:0:
Classifier_id  Group_id  Group_Qos_id  Sid   SFID   ref_count
7              1         1             8196  10     1
8              2         1             8197  11     1
```

## Troubleshooting Tips

Make sure that CM can listen to the RF-frequencies specified for the Wideband interfaced chosen for forwarding multicast traffic.

# Viewing Current Cache

Use this command to show the current multicast replication session per L2 forwarding interface.

- If you do not specify an interface, this command shows a summary of the current L2 forwarding interface. The summary includes the cache number.
- If you specify an interface, this command shows a summary of the interface. Add the verbose option for more detailed information of the cache.

```
Router#show cable multicast ses-cache global summary
```

```
Global Cache Config: 20
-----------------------------------------------------------------
Fwd             Cache      Cache     Cache      Cache
Intfc           Config     Used      Missed     Hitted
Wi7/0/0:1        10         4          4           12
-----------------------------------------------------------------
Total                            4            4              12

Router# show cable multicast ses-cache global

Fwd Intfc          Sub Intfc               Session (S,G)
Wi7/0/0:0          Bundle1                (30.30.30.30,227.0.0.20)
                   Bundle1                (30.30.30.30,227.0.0.22)

Wi7/0/0:1          Bundle1                (30.30.30.30,226.0.0.20)
                   Bundle1                (30.30.30.30,226.0.0.22)
                   Bundle1                (30.30.30.30,226.0.0.23)
                   Bundle1                (30.30.30.30,226.0.0.21)



Router#show cable multicast ses-cache interface wi7/0/0:1

Fwd Intfc          Sub Intfc              Session (S,G)
Wi7/0/0:1          Bundle1                (30.30.30.30,226.0.0.20)
                   Bundle1                (30.30.30.30,226.0.0.22)
                   Bundle1                (30.30.30.30,226.0.0.23)
                   Bundle1                (30.30.30.30,226.0.0.21)

Router# show cable multicast ses-cache interface wi7/0/0:1 summary

Global Cache Config: 20
--------------------------------------------------
Fwd             Cache       Cache    Cache      Cache
Intfc             Config      Used     Missed    Hitted
Wi7/0/0:1    10              4           4              12


Router# show cable multicast ses-cache wi8/0/0:0 verbose

Multicast Group    : 232.10.0.8

         Source    : 100.0.0.2

         Act GCRs  : 1

         Interface : Bu255 State: A GI: Bu255  RC: 0

         GCR       : GC   SAID    SFID    Key   GQC    GEn

                     10   8858    24      0     1      0


Multicast Group    : 232.10.0.16

         Source    : 100.0.0.2

         Act GCRs  : 1

         Interface : Bu255  State: A  GI: Bu255  RC: 0
         GCR       : GC   SAID    SFID    Key   GQC    GEn

                     10   8859    25      0     1      0

Total session cache num: 2
```

For the **Cache Missed** value, the value is increased for a new join request when cached entry is not available for reusing.

# Configuration Examples for DOCSIS 3.0 Multicast Support

This section provides the following configuration examples:

## Example: Configuring Basic Multicast Forwarding

**Note** The commands given below are required to enable the Cisco CMTS to forward multicast packets. However, Multicast QoS, and Authorization features are all optional for multicast packets to be forwarded correctly.

In the following example, a basic multicast forwarding profile is configured.

```
ip multicast-routing
interface TenGigabitEthernet4/1/0
  ip pim sparse-dense-mode
interface Bundle 1
  ip pim sparse-mode
  ip igmp version 3
```

## Example: Configuring Multicast QoS

**Note** A default service class and GQC must be defined before proceeding with configuring Multicast QoS.

In the following example, Multicast QoS is configured. You should define three objects and templates and then associate these to a particular bundle or forwarding interface. The objects are Service-Class, Group-QoS-Config (GQC), and Group-Config.

```
cable service class 1 name MQOS_DEFAULT
cable service class 1 downstream
cable service class 1 max-rate 10000000
cable service class 1 min-rate 1000000
cable multicast group-qos default scn MQOS_DEFAULT aggregate
cable multicast group-qos 10 scn MQOS single
cable multicast qos group 20 priority 1
application-id 10
session-range 230.0.0.0 255.0.0.0
tos 1 6 15
vrf name1
cable multicast qos group 20 priority 63 global
```

## Example: Configuring Forwarding Interface Selection Based on Service Flow Attribute

In the following example, the service flow attribute-based Forwarding Interface Selection is configured. To send multicast traffic for group 230.1.1.1, interface W6/0/0:0 is selected. The multicast QoS parameters are taken from group qos 1 (effectively from service class “mcast10”).

```
cable service class 10 name mcast10
cable service class 10 downstream
cable service class 10 max-rate 1000000
cable service class 10 min-rate 1000000
cable service class 10 req-attr-mask 8000000F
cable service class 10 forb-attr-mask 7FFFFFF0
cable multicast group-qos 1 scn mcast10 aggregate
cable multicast qos group 1 priority 1
session-range 230.1.1.1 255.255.255.255
 group-qos 1
interface Bundle1
 ip address 40.1.1.1 255.255.255.0
 ip pim sparse-mode
 ip helper-address 2.39.16.1
 cable multicast-qos group 1
end
interface Wideband-Cable6/0/0:0
cable bundle 10
cable rf-channels channel-list 0-7 bandwidth-percent 20
cable downstream attribute-mask 8000000F
end
```

## Example: Configuring Multicast Replication Session

The following example shows how to enable the multicast replication session on each L2 forwarding interface.

```
enable
conf t
interface xxx

[no|default] cable multicast ses-cache
 cable multicast ses-cache 3
```

# Additional References

The following sections provide references related to the DOCSIS 3.0 Multicast Support on the CMTS Routers.

### Related Documents

| Related Topic | Document Title |
|---|---|
| CMTS cable commands | http://www.cisco.com/en/US/docs/ios/cable/command/reference/cbl_book.htmlCisco IOS CMTS Cable Command Reference |
| Multicast VPN and DOCSIS 3.0 Multicast QoS | Multicast VPN and DOCSIS 3.0 Multicast QoS Support |

| Related Topic | Document Title |
|---|---|
| DOCSIS 3.0 QoS Support | DOCSIS WFQ Scheduler on the Cisco CMTS Routers |

### Standards

| Standard | Title |
|---|---|
| CM-SP-CMCIv3-I01-080320 | Cable Modem to Customer Premise Equipment Interface Specification |
| CM-SP-MULPIv3.0-I08-080522 | MAC and Upper Layer Protocols Interface Specification |
| CM-SP-OSSIv3.0-I07-080522 | Operations Support System Interface Specification |
| CM-SP-PHYv3.0-I07-080522 | Physical Layer Specification |
| CM-SP-SECv3.0-I08-080522 | Security Specification |

### MIBs

| MIB[1] | MIBs Link |
|---|---|
| • DOCS-MCAST-AUTH-MIB<br>• DOCS-MCAST-MIB | To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL:<br>http://www.cisco.com/go/mibs |

[1] Not all supported MIBs are listed.

### RFCs

| RFC | Title |
|---|---|
| No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature. | — |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for DOCSIS 3.0 Multicast Support

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 13: Feature Information for DOCSIS 3.0 Multicast Support*

| Feature Name | Releases | Feature Information |
|---|---|---|
| DOCSIS 3.0 Multicast Support | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into the Cisco cBR Series Converged Broadband Routers. |
| Dynamic Multicast Replication Sessions | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into the Cisco cBR Series Converged Broadband Routers. |

CHAPTER 8

# IPv6 Segment Routing on Cisco cBR

In Cisco Converged Broadband Router, IPv6 Segment Routing is available as a sub mode of IPv6 address configuration.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 14: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information about IPv6 Segment Routing

IPv6 Segment Routing (SR) is an SDN technology supporting IPv6 forwarding. In SR, a source or edge router performs source routing of traffic and encodes it as a segment list in an IPv6 routing extension header. The network is not required to maintain a per-application or per-flow state.

Any IPv6 capable node in a network may forward IPv6 traffic with an SR extension header to the first segment in the segment list without supporting IPv6 Segment Routing (SRv6).

At the node that hosts the current segment in the segment list, SRv6 is configured to modify the destination address of the traffic containing the SR extension header and destined to that segment ID. As part of SRv6 final processing, the next segment ID in the SR extension header is written to the destination address of the packet and a lookup is performed to forward the traffic to the new destination address.

The forwarding and SRv6 end processing continues at nodes hosting the segment IDs in the SR extension header until the last segment in the list is removed and the traffic is delivered to its ultimate destination.

## Restriction for Configuring IPv6 Segment Routing

Configuring duplicate IPv6 addresses on the same interface is not allowed.

# How to Configure IPv6 Segment Routing

## Configuring IPv6 Segment Routing on cBR

To configure IPv6 segment routing, use the following procedure.

1. Enter segment-routing sub mode when configuring an IPv6 address on an interface.

```
enable
configure terminal
interface type [slot_#/]port_#
ipv6 address ipv6_address_prefix/prefix_length
ipv6 address ipv6_address_prefix/prefix_length segment-routing
```

2. Define a local prefix as an SID

```
ipv6-sr prefix-sid
exit
```

## Verifying IPv6 Segment Routing Configuration

The following example shows how to verify SRv6 configuration:c

```
Router#sh run
*Oct 17 13:13:23.975: %SYS-5-CONFIG_I: Configured from console by console
Router#sh run | sec Ether
interface Ethernet0/0
 no ip address
 shutdown
 ipv6 address 2001::2001/64 segment-routing >>>>>>>
  ipv6-sr prefix-sid >>>>>>>
```

## Configure Multiple IPv6 Addresses for Segment Routing

To configure multiple IPv6 addresses for SRv6 under the same interface, use the following commands.

1. Enter segment-routing sub mode when configuring an IPv6 address on an interface.

```
enable
configure terminal
interface type [slot_#/]port_#
ipv6 address ipv6_address_prefix/prefix_length segment-routing
```

2. Define a local prefix as an SID.

```
ipv6-sr prefix-sid
exit
```

## Verifying IPv6 Segment Routing Configuration on Multiple IPv6 Addresses

The following example shows how to verify SRv6 configuration for multiple IPv6 addresses:

```
Router#sh run | sec Ether
interface Ethernet0/0
 no ip address
 shutdown
 ipv6 address 2001:db8:110::/64 segment-routing  >>> submode 1
  ipv6-sr prefix-sid
 ipv6 address 2001:db9:111::/64 segment-routing  >>> submode 2
  ipv6-sr prefix-sid
interface Ethernet0/1
 no ip address
 shutdown
interface Ethernet0/2
 no ip address
 shutdown
interface Ethernet0/3
 no ip address
 shutdown
interface Ethernet1/0
 no ip address
 shutdown
interface Ethernet1/1
 no ip address
 shutdown
interface Ethernet1/2
```

## Disabling Prefix SID

To disable the local prefix SID associated to the segment ID, use the following commands.

```
enable
configure terminal
interface type [slot_#/]port_#
ipv6 address ipv6_address_prefix/prefix_length segment-routing
no ipv6-sr prefix-sid
end
```

## Verifying whether Prefix SID is Disabled

The following example shows how to verify whether the prefix SID is disabled:

```
Router#sh run | sec Ether
interface Ethernet0/0
 no ip address
 shutdown
 ipv6 address 110::110/64 segment-routing >>> "ipv6-sr prefix sid" is no longer present
ipv6 address 111::111/64 segment-routing
ipv6-sr prefix-sid
```

## Disabling SRv6 for a Prefix-SID

To disable SRv6 configuration for an IPv6 address and remove the IPv6 address, use the following command:

```
enable
configure terminal
interface type [slot_#/]port_#
```

```
no ipv6 address ipv6_address_prefix/prefix_length segment-routing
end
```

## Verifying whether SRv6 is Disabled and Prefix SID Removed

The following example shows how to verify whether SRv6 is disabled and the prefix SID is removed for the prefix SID.

```
Router#sh run |
*Oct 17 13:17:51.523: %SYS-5-CONFIG_I: Configured from console by console
Router#sh run | sec Ether
interface Ethernet0/0
 no ip address
 shutdown
 ipv6 address 110::110/64 segment-routing
ipv6 address 111::111/64 segment-routing is entirely removed from ethernet0/0
```

# Configuration Examples

This section provides examples for IPv6 Segment Routing.

## Example: Configuring IPv6 Segment Routing on Cisco cBR

```
Router#conf terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#inter Ether0/0
Router(config-if)#ipv6 address 110::110/64 ?
  anycast
  eui-64
  segment-routing
  <cr>
Router(config-if)#ipv6 address 110::110/64 segment-routing
Router(config-if-sr-ipv6)#?
ipv6 address segment-routing mode configuration commands:
  default  Set a command to its defaults
  exit     Exit from SR submode
  ipv6-sr  Request options specific to IPV6 segment-routing
  no       Negate a command or set its defaults
Router(config-if-sr-ipv6)#ipv6-sr ?
  prefix-sid  Set host prefix as IPv6 SR identifier prefix-sid
Router(config-if-sr-ipv6)#ipv6-sr prefix-sid
Router(config-if-sr-ipv6)#exit
Router(config-if)#exit
Router(config)#exit
Router#
```

## Example: Configure Multiple IPv6 Addresses for SRv6

```
Router#conf terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#inter Ether 0/0
Router(config-if)# ipv6 address 110::110/64 segment-routing
Router(config-if)# ipv6 address 111::111/64 segment-routing
Router(config-if-sr-ipv6)#ipv6-sr prefix-sid
Router(config-if-sr-ipv6)#end
```

## Example: Disabling Prefix SID

```
Router#conf terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#inter Ether0/0
Router(config-if)#inter Ether0/0
Router(config-if)#ipv6 address 110::110/64 segment-routing
Router(config-if-sr-ipv6)#no ipv6-sr prefix-sid
Router(config-if-sr-ipv6)#end
```

## Example: Disabling SR with an Active Prefix SID

```
Router#conf terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#inter Ether0/0
Router(config-if)#no ipv6 address 111::111/64 segment-routing
Router(config-if)#end
```

# Feature Information for IPv6 Segment Routing

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 15: Feature Information for IPv6 Segment Routing*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IPv6 Segment Routing | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |

PART I

# IP Access Control Lists

CHAPTER 9

# IP Access Control Lists

Access control lists (ACLs) perform packet filtering to control which packets move through a network and to where. The packet filtering provides security by helping to limit the network traffic, restrict the access of users and devices to a network, and prevent the traffic from leaving a network. IP access lists reduce the chance of spoofing and denial-of-service attacks and allow dynamic, temporary user-access through a firewall.

IP access lists can also be used for purposes other than security, such as bandwidth control, restrict the content of routing updates, redistribute routes, trigger dial-on-demand (DDR) calls, limit debug output, and identify or classify traffic for quality of service (QoS) features. This module provides an overview of IP access lists.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 16: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information About IP Access Lists

## Benefits of IP Access Lists

Access control lists (ACLs) perform packet filtering to control the flow of packets through a network. Packet filtering can restrict the access of users and devices to a network, providing a measure of security. Access lists can save network resources by reducing traffic. The benefits of using access lists are as follows:

- Authenticate incoming rsh and rcp requests—Access lists can simplify the identification of local users, remote hosts, and remote users in an authentication database that is configured to control access to a device. The authentication database enables Cisco software to receive incoming remote shell (rsh) and remote copy (rcp) protocol requests.

- Block unwanted traffic or users—Access lists can filter incoming or outgoing packets on an interface, thereby controlling access to a network based on source addresses, destination addresses, or user authentication. You can also use access lists to determine the types of traffic that are forwarded or blocked at device interfaces. For example, you can use access lists to permit e-mail traffic to be routed through a network and to block all Telnet traffic from entering the network.
- Control access to vty—Access lists on an inbound vty (Telnet) can control who can access the lines to a device. Access lists on an outbound vty can control the destinations that the lines from a device can reach.
- Identify or classify traffic for QoS features—Access lists provide congestion avoidance by setting the IP precedence for Weighted Random Early Detection (WRED) and committed access rate (CAR). Access lists also provide congestion management for class-based weighted fair queueing (CBWFQ), priority queueing, and custom queueing.
- Limit debug command output—Access lists can limit debug output based on an IP address or a protocol.
- Provide bandwidth control—Access lists on a slow link can prevent excess traffic on a network.
- Provide NAT control—Access lists can control which addresses are translated by Network Address Translation (NAT).
- Reduce the chance of DoS attacks—Access lists reduce the chance of denial-of-service (DoS) attacks. Specify IP source addresses to control traffic from hosts, networks, or users from accessing your network. Configure the TCP Intercept feature to can prevent servers from being flooded with requests for connection.
- Restrict the content of routing updates—Access lists can control routing updates that are sent, received, or redistributed in networks.
- Trigger dial-on-demand calls—Access lists can enforce dial and disconnect criteria.

# Border Routers and Firewall Routers Should Use Access Lists

There are many reasons to configure access lists; for example, you can use access lists to restrict contents of routing updates or to provide traffic flow control. One of the most important reasons to configure access lists is to provide a basic level of security for your network by controlling access to it. If you do not configure access lists on your router, all packets passing through the router could be allowed onto all parts of your network.

An access list can allow one host to access a part of your network and prevent another host from accessing the same area. In the figure below, by applying an appropriate access list to the interfaces of the router, Host A is allowed to access the Human Resources network and Host B is prevented from accessing the Human Resources network.

Access lists should be used in firewall routers, which are often positioned between your internal network and an external network such as the Internet. You can also use access lists on a router positioned between two parts of your network, to control traffic entering or exiting a specific part of your internal network.

To provide some security benefits of access lists, you should at least configure access lists on border routers--routers located at the edges of your networks. Such an access list provides a basic buffer from the outside network or from a less controlled area of your own network into a more sensitive area of your network. On these border routers, you should configure access lists for each network protocol configured on the router interfaces. You can configure access lists so that inbound traffic or outbound traffic or both are filtered on an interface.

Access lists are defined on a per-protocol basis. In other words, you should define access lists for every protocol enabled on an interface if you want to control traffic flow for that protocol.

# Definition of an Access List

Access control lists (ACLs) perform packet filtering to control the movement of packets through a network. Packet filtering provides security by limiting the access of traffic into a network, restricting user and device access to a network, and preventing traffic from leaving a network. IP access lists reduce the chance of spoofing and denial-of-service attacks, and allow dynamic, temporary user-access through a firewall.

IP access lists can also be used for purposes other than security, such as to control bandwidth, restrict the content of routing updates, redistribute routes, trigger dial-on-demand (DDR) calls, limit debug output, and identify or classify traffic for quality of service (QoS) features.

An access list is a sequential list that consists of at least one **permit** statement and possibly one or more **deny** statements. In the case of IP access lists, these statements can apply to IP addresses, upper-layer IP protocols, or other fields in IP packets.

Access lists are identified and referenced by a name or a number. Access lists act as packet filters, filtering packets based on the criteria defined in each access list.

After you configure an access list, for the access list to take effect, you must either apply the access list to an interface (by using the **ip access-group** command), a vty (by using the **access-class** command), or reference the access list by any command that accepts an access list. Multiple commands can reference the same access list.

In the following configuration, an IP access list named branchoffices is configured on Ten Gigabit Ethernet interface 4/1/0 and applied to incoming packets. Networks other than the ones specified by the source address and mask pair cannot access Ten Gigabit Ethernet interface 4/1/0. The destinations for packets coming from sources on network 172.16.7.0 are unrestricted. The destination for packets coming from sources on network 172.16.2.0 must be 172.31.5.4.

```
ip access-list extended branchoffices
 10 permit 172.16.7.0 0.0.0.3 any
 20 permit 172.16.2.0 0.0.0.255 host 172.31.5.4
!
interface tengigabitethernet 4/1/0
 ip access-group branchoffices in
```

# Access List Rules

The following rules apply to access lists:

- Only one access list per interface, per protocol, and per direction is allowed.
- An access list must contain at least one **permit** statement or all packets are denied entry into the network.
- The order in which access list conditions or match criteria are configured is important. While deciding whether to forward or block a packet, Cisco software tests the packet against each criteria statement in the order in which these statements are created. After a match is found, no more criteria statements are checked. The same **permit** or **deny** statements specified in a different order can result in a packet being passed under one circumstance and denied in another circumstance.

- If an access list is referenced by a name, but the access list does not exist, all packets pass. An interface or command with an empty access list applied to it permits all traffic into the network.
- Standard access lists and extended access lists cannot have the same name.
- Inbound access lists process packets before the packets are routed to an outbound interface. Inbound access lists that have filtering criteria that deny packet access to a network saves the overhead of routing lookup. Packets that are permitted access to a network based on the configured filtering criteria are processed for routing. For inbound access lists, when you configure a **permit** statement, packets are processed after they are received, and when you configure a **deny** statement, packets are discarded.
- Outbound access lists process packets before they leave the device. Incoming packets are routed to the outbound interface and then processed by the outbound access list. For outbound access lists, when you configure a **permit** statement, packets are sent to the output buffer, and when you configure a **deny** statement, packets are discarded.
- An access list can control traffic arriving at a device or leaving a device, but not traffic originating at a device.

## Helpful Hints for Creating IP Access Lists

The following tips will help you avoid unintended consequences and help you create more efficient, useful access lists.

- Create the access list before applying it to an interface (or elsewhere), because if you apply a nonexistent access list to an interface and then proceed to configure the access list, the first statement is put into effect, and the implicit **deny** statement that follows could cause you immediate access problems.
- Another reason to configure an access list before applying it is because an interface with an empty access list applied to it permits all traffic.
- All access lists need at least one **permit** statement; otherwise, all packets are denied and no traffic passes.
- Use the statement **permit any any** if you want to allow all other packets not already denied. Using the statement **permit any any** in effect avoids denying all other packets with the implicit deny statement at the end of an access list. Do not make your first access list entry **permit any any** because all traffic will get through; no packets will reach the subsequent testing. In fact, once you specify **permit any any**, all traffic not already denied will get through.
- Although all access lists end with an implicit **deny** statement, we recommend use of an explicit **deny** statement (for example, **deny ip any any**). On most platforms, you can display the count of packets denied by issuing the **show access-list** command, thus finding out more information about who your access list is disallowing. Only packets denied by explicit **deny** statements are counted, which is why the explicit **deny** statement will yield more complete data for you.
- While you are creating an access list or after it is created, you might want to delete an entry.
  - You cannot delete an entry from a numbered access list; trying to do so will delete the entire access list. If you need to delete an entry, you need to delete the entire access list and start over.
  - You can delete an entry from a named access list. Use the **no permit** or **no deny** command to delete the appropriate entry.
- In order to make the purpose of individual statements more scannable and easily understood at a glance, you can write a helpful remark before or after any statement by using the **remark** command.

- If you want to deny access to a particular host or network and find out if someone from that network or host is attempting to gain access, include the **log** keyword with the corresponding **deny** statement so that the packets denied from that source are logged for you.
- This hint applies to the placement of your access list. When trying to save resources, remember that an inbound access list applies the filter conditions before the routing table lookup. An outbound access list applies the filter conditions after the routing table lookup.

# Named or Numbered Access Lists

All access lists must be identified by a name or a number. Named access lists are more convenient than numbered access lists because you can specify a meaningful name that is easier to remember and associate with a task. You can reorder statements in or add statements to a named access list.

Named access lists support the following features that are not supported by numbered access lists:

- IP options filtering
- Noncontiguous ports
- TCP flag filtering
- Deleting of entries with the **no permit** or **no deny** command

**Note** Not all commands that accept a numbered access list will accept a named access list. For example, vty uses only numbered access lists.

# Standard or Extended Access Lists

All access lists are either standard or extended access lists. If you only intend to filter on a source address, the simpler standard access list is sufficient. For filtering on anything other than a source address, an extended access list is necessary.

- Named access lists are specified as standard or extended based on the keyword **standard** or **extended** in the **ip access-list** command syntax.
- Numbered access lists are specified as standard or extended based on their number in the **access-list** command syntax. Standard IP access lists are numbered 1 to 99 or 1300 to 1999; extended IP access lists are numbered 100 to 199 or 2000 to 2699. The range of standard IP access lists was initially only 1 to 99, and was subsequently expanded with the range 1300 to 1999 (the intervening numbers were assigned to other protocols). The extended access list range was similarly expanded.

### Standard Access Lists

Standard IP access lists test only source addresses of packets (except for two exceptions). Because standard access lists test source addresses, they are very efficient at blocking traffic close to a destination. There are two exceptions when the address in a standard access list is not a source address:

- On outbound VTY access lists, when someone is trying to telnet, the address in the access list entry is used as a destination address rather than a source address.

- When filtering routes, you are filtering the network being advertised to you rather than a source address.

### Extended Access Lists

Extended access lists are good for blocking traffic anywhere. Extended access lists test source and destination addresses and other IP packet data, such as protocols, TCP or UDP port numbers, type of service (ToS), precedence, TCP flags, and IP options. Extended access lists can also provide capabilities that standard access lists cannot, such as the following:

- Filtering IP Options
- Filtering TCP flags
- Filtering noninitial fragments of packets (see the module “Refining an IP Access List”)

**Note** Packets that are subject to an extended access list will not be autonomous switched.

## IP Packet Fields You Can Filter to Control Access

You can use an extended access list to filter on any of the following fields in an IP packet. Source address and destination address are the two most frequently specified fields on which to base an access list:

- Source address--Specifies a source address to control packets coming from certain networking devices or hosts.
- Destination address--Specifies a destination address to control packets being sent to certain networking devices or hosts.
- Protocol--Specifies an IP protocol indicated by the keyword **eigrp**, **gre**, **icmp**, **igmp**, **ip**, **ipinip**, **nos**, **ospf**, **tcp**, or **udp**, or indicated by an integer in the range from 0 to 255 (representing an Internet protocol). If you specify a transport layer protocol (**icmp**, **igmp**, **tcp**, or **udp**), the command has a specific syntax.
    - Ports and non-contiguous ports--Specifies TCP or UDP ports by a port name or port number. The port numbers can be noncontiguous port numbers. Port numbers can be useful to filter Telnet traffic or HTTP traffic, for example.
    - TCP flags--Specifies that packets match any flag or all flags set in TCP packets. Filtering on specific TCP flags can help prevent false synchronization packets.
- IP options--Specifies IP options; one reason to filter on IP options is to prevent routers from being saturated with spurious packets containing them.

## Wildcard Mask for Addresses in an Access List

Address filtering uses wildcard masking to indicate to the software whether to check or ignore corresponding IP address bits when comparing the address bits in an access list entry to a packet being submitted to the access list. By carefully setting wildcard masks, you can specify one or more IP addresses for permit or deny tests.

Wildcard masking for IP address bits uses the number 1 and the number 0 to specify how the software treats the corresponding IP address bits. A wildcard mask is sometimes referred to as an inverted mask because a 1 and 0 mean the opposite of what they mean in a subnet (network) mask.

- A wildcard mask bit 0 means check the corresponding bit value; they must match.
- A wildcard mask bit 1 means ignore that corresponding bit value; they need not match.

If you do not supply a wildcard mask with a source or destination address in an access list statement, the software assumes an implicit wildcard mask of 0.0.0.0, meaning all values must match.

Unlike subnet masks, which require contiguous bits indicating network and subnet to be ones, wildcard masks allow noncontiguous bits in the mask.

The table below shows examples of IP addresses and masks from an access list, along with the corresponding addresses that are considered a match.

***Table 17: Sample IP Addresses, Wildcard Masks, and Match Results***

| Address | Wildcard Mask | Match Results |
|---|---|---|
| 0.0.0.0 | 255.255.255.255 | All addresses will match the access list conditions. |
| 172.18.0.0/16 | 0.0.255.255 | Network 172.18.0.0 |
| 172.18.5.2/16 | 0.0.0.0 | Only host 172.18.5.2 matches |
| 172.18.8.0 | 0.0.0.7 | Only subnet 172.18.8.0/29 matches |
| 172.18.8.8 | 0.0.0.7 | Only subnet 172.18.8.8/29 matches |
| 172.18.8.15 | 0.0.0.3 | Only subnet 172.18.8.15/30 matches |
| 10.1.2.0 | 0.0.252.255 (noncontiguous bits in mask) | Matches any even-numbered network in the range of 10.1.2.0 to 10.1.254.0 |

# Access List Sequence Numbers

The ability to apply sequence numbers to IP access list entries simplifies access list changes. Prior to the IP Access List Entry Sequence Numbering feature, there was no way to specify the position of an entry within an access list. If you wanted to insert an entry in the middle of an existing list, all of the entries after the desired position had to be removed, then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

This feature allows users to add sequence numbers to access list entries and resequence them. When you add a new entry, you specify the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry.

# Access List Logging

The Cisco IOS software can provide logging messages about packets permitted or denied by a single standard or extended IP access list entry. That is, any packet that matches the entry will cause an informational logging message about the packet to be sent to the console. The level of messages logged to the console is controlled by the **logging console** global configuration command.

The first packet that triggers the access list entry causes an immediate logging message, and subsequent packets are collected over 5-minute intervals before they are displayed or logged. The logging message includes the access list number, whether the packet was permitted or denied, the source IP address of the packet, and the number of packets from that source permitted or denied in the prior 5-minute interval.

However, you can use the **ip access-list log-update** command to set the number of packets that, when match an access list (and are permitted or denied), cause the system to generate a log message. You might want to do this to receive log messages more frequently than at 5-minute intervals.

**Caution** If you set the *number-of-matches* argument to 1, a log message is sent right away, rather than caching it; every packet that matches an access list causes a log message. A setting of 1 is not recommended because the volume of log messages could overwhelm the system.

Even if you use the **ip access-list log-update** command, the 5-minute timer remains in effect, so each cache is emptied at the end of 5 minutes, regardless of the count of messages in each cache. Regardless of when the log message is sent, the cache is flushed and the count reset to 0 for that message the same way it is when a threshold is not specified.

**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 router 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.

## Alternative to Access List Logging

Packets matching an entry in an ACL with a log option are process switched. It is not recommended to use the log option on ACLs, but rather use NetFlow export and match on a destination interface of Null0. This is done in the CEF path. The destination interface of Null0 is set for any packet that is dropped by the ACL.

# Additional IP Access List Features

Beyond the basic steps to create a standard or extended access list, you can enhance your access lists as mentioned below. Each of these methods is described completely in the module entitled “Refining an Access List.”

- You can impose dates and times when **permit** or **deny** statements in an extended access list are in effect, making your access list more granular and specific to an absolute or periodic time period.
- After you create a named access list, you might want to add entries or change the order of the entries, known as resequencing an access list.
- You can achieve finer granularity when filtering packets by filtering on noninitial fragments of packets.

# Where to Apply an Access List

You can apply access lists to the inbound or outbound interfaces of a device. Applying an access list to an inbound interface controls the traffic that enters the interface and applying an access list to an outbound interface controls the traffic that exits the interface.

When software receives a packet at the inbound interface, the software checks the packet against the statements that are configured for the access list. If the access list permits packets, the software processes the packet. Applying access lists to filter incoming packets can save device resources because filtered packets are discarded before entering the device.

Access lists on outbound interfaces filter packets that are transmitted (sent) out of the interface. You can use the TCP Access Control List (ACL) Splitting feature of the Rate-Based Satellite Control Protocol (RBSCP) on the outbound interface to control the type of packets that are subject to TCP acknowledgment (ACK) splitting on an outbound interface.

You can reference an access list by using a **debug** command to limit the amount of debug logs. For example, based on the filtering or matching criteria of the access list, debug logs can be limited to source or destination addresses or protocols.

You can use access lists to control routing updates, dial-on-demand (DDR), and quality of service (QoS) features.

# Additional References

### Related Documents

| Related Topic | Document Title |
|---|---|
| IP access list commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples | Cisco IOS IP Addressing Services Command Reference |
| Filtering on source address, destination address, or protocol | Creating an IP Access List and Applying It to an Interface” module |
| Filtering on IP Options, TCP flags, noncontiguous ports, or TTL | Creating an IP Access List to Filter IP Options, TCP Flags, or Noncontiguous Ports module |

### Standards

| Standards & RFCs | Title |
|---|---|
| None | — |

### MIBs

| MIB | MIBs Link |
|---|---|
| None | To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL:<br>http://www.cisco.com/go/mibs |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for IP Access Lists

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 18: Feature Information for IP Access Lists*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IP access lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |

CHAPTER 10

# Creating an IP Access List and Applying It to an Interface

IP access lists provide many benefits for securing a network and achieving nonsecurity goals, such as determining quality of service (QoS) factors or limiting **debug** command output. This module describes how to create standard, extended, named, and numbered IP access lists. An access list can be referenced by a name or a number. Standard access lists filter on only the source address in IP packets. Extended access lists can filter on source address, destination address, and other fields in an IP packet.

After you create an access list, you must apply it to something in order for it to have any effect. This module describes how to apply an access list to an interface. However, there are many other uses for access lists, which are mentioned in this module and described in other modules and in other configuration guides for various technologies.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

# Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 19: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information About Creating an IP Access List and Applying It to an Interface

## Helpful Hints for Creating IP Access Lists

- Create the access list before applying it to an interface (or elsewhere), because if you apply a nonexistent access list to an interface and then proceed to configure the access list, the first statement is put into effect, and the implicit **deny** statement that follows could cause immediate access problems.

- Another reason to configure an access list before applying it is because an interface with an empty access list applied to it permits all traffic.
- All access lists need at least one **permit** statement; otherwise, all packets are denied and no traffic passes.
- Organize your access list so that more specific references in a network or subnet appear before more general ones.
- A packet will match the first ACE in the ACL. Thus, a **permit ip any any** will match all packets, ignoring all subsequent ACES.
- Although all access lists end with an implicit **deny** statement, we recommend use of an explicit **deny** statement (for example, **deny ip any any**). On most platforms, you can display the count of packets denied by issuing the **show access-list** command, thus finding out more information about who your access list is disallowing. Only packets denied by explicit **deny** statements are counted, which is why the explicit **deny** statement will yield more complete data for you.
- While you are creating an access list or after it is created, you might want to delete an entry. You can delete an entry from a named access list. Use the **no permit** or **no deny** command to delete the appropriate entry.
- In order to make the purpose of individual statements more scannable and easily understood at a glance, you can write a helpful remark before or after any statement by using the **remark** command.
- If you want to deny access to a particular host or network and find out if someone from that network or host is attempting to gain access, include the **log** keyword with the corresponding **deny** statement so that the packets denied from that source are logged for you.
- This hint applies to the placement of your access list. When trying to save resources, remember that an inbound access list applies the filter conditions before the routing table lookup. An outbound access list applies the filter conditions after the routing table lookup.

# Access List Remarks

You can include comments or remarks about entries in any IP access list. An access list remark is an optional remark before or after an access list entry that describes the entry so that you do not have to interpret the purpose of the entry. Each remark is limited to 100 characters in length.

The remark can go before or after a **permit** or **deny** statement. Be consistent about where you add remarks. Users may be confused if some remarks precede the associated **permit** or **deny** statements and some remarks follow the associated statements.

The following is an example of a remark that describes function of the subsequent **deny** statement:

```
ip access-list extended telnetting
 remark Do not allow host1 subnet to telnet out
 deny tcp host 172.16.2.88 any eq telnet
```

# Additional IP Access List Features

Beyond the basic steps to create a standard or extended access list, you can enhance your access lists as mentioned below. Each of these methods is described completely in the *Refining an IP Access List module*.

- You can impose dates and times when **permit** or **deny** statements in an extended access list are in effect, making your access list more granular and specific to an absolute or periodic time period.
- After you create a named or numbered access list, you might want to add entries or change the order of the entries, known as resequencing an access list.
- You can achieve finer granularity when filtering packets by filtering on noninitial fragments of packets.

# How to Create an IP Access List and Apply It to an Interface

This section describes the general ways to create a standard or extended access list using either a name or a number. Access lists are very flexible; the tasks simply illustrate one **permit** command and one **deny** command to provide you the command syntax of each. Only you can determine how many **permit** and **deny** commands you need and their order.

**Note** The first two tasks in this module create an access list; you must apply the access list in order for it to function. If you want to apply the access list to an interface, perform the task "Applying the Access List to an Interface".

## Creating a Standard Access List to Filter on Source Address

If you want to filter on source address only, a standard access list is simple and sufficient. There are two alternative types of standard access list: named and numbered. Named access lists allow you to identify your access lists with a more intuitive name rather than a number, and they also support more features than numbered access lists.

### Creating a Named Access List to Filter on Source Address

Use a standard, named access list if you need to filter on source address only. This task illustrates one **permit** statement and one **deny** statement, but the actual statements you use and their order depend on what you want to filter or allow. Define your **permit** and **deny** statements in the order that achieves your filtering goals.

**Step 1** **enable**

**Example:**

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

**Step 2** **configure terminal**

**Example:**

```
Device# configure terminal
```

Enters global configuration mode.

**Step 3** **ip access-list standard** *name*

**Example:**

```
Device(config)# ip access-list standard R&D
```

Defines a standard IP access list using a name and enters standard named access list configuration mode.

**Step 4** **remark** *remark*

**Example:**

```
Device(config-std-nacl)# remark deny Sales network
```

(Optional) Adds a user-friendly comment about an access list entry.

- A remark can precede or follow an access list entry.
- In this example, the remark reminds the network administrator that the subsequent entry denies the Sales network access to the interface (assuming this access list is later applied to an interface).

**Step 5** **deny** {*source* [*source-wildcard*] | **any**} [**log**]

**Example:**

```
Device(config-std-nacl)# deny 172.16.0.0 0.0.255.255 log
```

(Optional) Denies the specified source based on a source address and wildcard mask.

- If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address.
- Optionally use the keyword **any** as a substitute for the *source source-wildcard* to specify the source and source wildcard of 0.0.0.0 255.255.255.255.
- In this example, all hosts on network 172.16.0.0 are denied passing the access list.
- Because this example explicitly denies a source address and the **log** keyword is specified, any packets from that source are logged when they are denied. This is a way to be notified that someone on a network or host is trying to gain access.

**Step 6** **remark** *remark*

**Example:**

```
Device(config-std-nacl)# remark Give access to Tester’s host
```

(Optional) Adds a user-friendly comment about an access list entry.

- A remark can precede or follow an access list entry.
- This remark reminds the network administrator that the subsequent entry allows the Tester’s host access to the interface.

**Step 7** **permit** {*source* [*source-wildcard*] | **any**} [**log**]

**Example:**

```
Device(config-std-nacl)# permit 172.18.5.22 0.0.0.0
```

Permits the specified source based on a source address and wildcard mask.

- Every access list needs at least one **permit** statement; it need not be the first entry.
- If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address.
- Optionally use the keyword **any** as a substitute for the *source source-wildcard* to specify the source and source wildcard of 0.0.0.0 255.255.255.255.
- In this example, host 172.18.5.22 is allowed to pass the access list.

**Step 8** Repeat some combination of Steps 4 through 7 until you have specified the sources on which you want to base your access list.

Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

**Step 9** **end**

**Example:**

```
Device(config-std-nacl)# end
```

Exits standard named access list configuration mode and enters privileged EXEC mode.

**Step 10** **show ip access-list**

**Example:**

```
Device# show ip access-list
```

(Optional) Displays the contents of all current IP access lists.

## Creating a Numbered Access List to Filter on Source Address

Configure a standard, numbered access list if you need to filter on source address only and you prefer not to use a named access list.

IP standard access lists are numbered 1 to 99 or 1300 to 1999. This task illustrates one **permit** statement and one **deny** statement, but the actual statements you use and their order depend on what you want to filter or allow. Define your **permit** and **deny** statements in the order that achieves your filtering goals.

**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** **access-list** *access-list-number permit* {*source* [*source-wildcard*] | **any**} [**log**]

**Example:**

```
Device(config)# access-list 1 permit 172.16.5.22 0.0.0.0
```

Permits the specified source based on a source address and wildcard mask.

- Every access list needs at least one permit statement; it need not be the first entry.
- Standard IP access lists are numbered 1 to 99 or 1300 to 1999.
- If the source-wildcard is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address.
- Optionally use the keyword any as a substitute for the source source-wildcard to specify the source and source wildcard of 0.0.0.0 255.255.255.255.
- In this example, host 172.16.5.22 is allowed to pass the access list.

**Step 4** **access-list** *access-list-number* **deny** {*source* [*source-wildcard*] | **any**} [**log**]

**Example:**

```
Device(config)# access-list 1 deny 172.16.7.34 0.0.0.0
```

Denies the specified source based on a source address and wildcard mask.

- If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address.
- Optionally use the abbreviation **any** as a substitute for the *source source-wildcard* to specify the source and source wildcard of 0.0.0.0 255.255.255.255.
- In this example, host 172.16.7.34 is denied passing the access list.

**Step 5** Repeat some combination of Steps 3 through 6 until you have specified the sources on which you want to base your access list.

Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

**Step 6** **end**

**Example:**

```
Device(config)# end
```

Exits global configuration mode and enters privileged EXEC mode.

**Step 7** **show ip access-list**

**Example:**

```
Device# show ip access-list
```

(Optional) Displays the contents of all current IP access lists.

# Creating an Extended Access List

If you want to filter on anything other than source address, you need to create an extended access list. There are two alternative types of extended access list: named and numbered. Named access lists allow you to identify your access lists with a more intuitive name rather than a number, and they also support more features.

For details on how to filter something other than source or destination address, see the syntax descriptions in the command reference documentation.

## Creating a Named Extended Access List

Create a named extended access list if you want to filter the source and destination address or filter a combination of addresses and other IP fields.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *name*
4. **deny** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
5. **permit** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
6. Repeat some combination of Steps 4 through 7 until you have specified the fields and values on which you want to base your access list.
7. **end**
8. **show ip access-list**

### DETAILED STEPS

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

| | Command or Action | Purpose |
|---|---|---|
| **Step 3** | **ip access-list extended** *name*<br>**Example:**<br>`Device(config)# ip access-list extended acl1` | Defines an extended IP access list using a name and enters extended named access list configuration mode. |
| **Step 4** | **deny** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** \| **log-input**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Device(config-ext-nacl)# deny ip 172.18.0.0 0.0.255.255 host 172.16.40.10 log` | (Optional) Denies any packet that matches all of the conditions specified in the statement.<br>• If the *source-wildcard* or *destination-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively.<br>• Optionally use the keyword **any** as a substitute for the *source source-wildcard* or *destination destination-wildcard* to specify the address and wildcard of 0.0.0.0 255.255.255.255.<br>• Optionally use the keyword **host** *source* to indicate a source and source wildcard of *source* 0.0.0.0 or the abbreviation **host** *destination* to indicate a destination and destination wildcard of *destination* 0.0.0.0.<br>• In this example, packets from all sources are denied access to the destination network 172.18.0.0. Logging messages about packets permitted or denied by the access list are sent to the facility configured by the **logging facility** command (for example, console, terminal, or syslog). That is, any packet that matches the access list will cause an informational logging message about the packet to be sent to the configured facility. The level of messages logged to the console is controlled by the **logging console** command. |
| **Step 5** | **permit** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** \| **log-input**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Device(config-ext-nacl)# permit tcp any any` | Permits any packet that matches all of the conditions specified in the statement.<br>• Every access list needs at least one permit statement.<br>• If the *source-wildcard* or *destination-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively.<br>• Optionally use the keyword **any** as a substitute for the *source source-wildcard* or *destination destination-wildcard* to specify the address and wildcard of 0.0.0.0 255.255.255.255.<br>• In this example, TCP packets are allowed from any source to any destination. |

| | Command or Action | Purpose |
|---|---|---|
| | | - Use the **log-input** keyword to include input interface, source MAC address, or virtual circuit in the logging output. |
| **Step 6** | Repeat some combination of Steps 4 through 7 until you have specified the fields and values on which you want to base your access list. | Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list. |
| **Step 7** | **end**<br>**Example:**<br>`Device(config-ext-nacl)# end` | Exits standard named access list configuration mode and enters privileged EXEC mode. |
| **Step 8** | **show ip access-list**<br>**Example:**<br>`Device# show ip access-list` | (Optional) Displays the contents of all current IP access lists. |

## Creating a Numbered Extended Access List

Create a numbered extended access list if you want to filter on source and destination address, or a combination of addresses and other IP fields, and you prefer not to use a name. Extended IP access lists are numbered 100 to 199 or 2000 to 2699.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list** *access-list-number* **remark** *remark*
4. **access-list** *access-list-number* **permit** *protocol* {*source* [*source-wildcard*] | **any**} {*destination* [*destination-wildcard*] | **any**} [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
5. **access-list** *access-list-number* **remark** *remark*
6. **access-list** *access-list-number* **deny** *protocol* {*source* [*source-wildcard*] | **any**} {*destination* [*destination-wildcard*] | **any**} [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
7. Repeat some combination of Steps 3 through 6 until you have specified the fields and values on which you want to base your access list.
8. **end**
9. **show ip access-list**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:** | Enables privileged EXEC mode.<br>- Enter your password if prompted. |

| | Command or Action | Purpose |
|---|---|---|
| | `Device> enable` | |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **access-list** *access-list-number* **remark** *remark*<br>**Example:**<br>`Device(config)# access-list 107 remark allow Telnet packets from any source to network 172.69.0.0 (headquarters)` | (Optional) Adds a user-friendly comment about an access list entry.<br>• A remark of up to 100 characters can precede or follow an access list entry. |
| **Step 4** | **access-list** *access-list-number* **permit** *protocol* {*source* [*source-wildcard*] \| **any**} {*destination* [*destination-wildcard*] \| **any**} [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** \| **log-input**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Device(config)# access-list 107 permit tcp any 172.69.0.0 0.0.255.255 eq telnet` | Permits any packet that matches all of the conditions specified in the statement.<br>• Every access list needs at least one **permit** statement; it need not be the first entry.<br>• Extended IP access lists are numbered 100 to 199 or 2000 to 2699.<br>• If the *source-wildcard* or *destination-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively.<br>• Optionally use the keyword **any** as a substitute for the *source source-wildcard* or *destination destination-wildcard* to specify the address and wildcard of 0.0.0.0 255.255.255.255.<br>• TCP and other protocols have additional syntax available. See the **access-list** command in the command reference for complete syntax. |
| **Step 5** | **access-list** *access-list-number* **remark** *remark*<br>**Example:**<br>`Device(config)# access-list 107 remark deny all other TCP packets` | (Optional) Adds a user-friendly comment about an access list entry.<br>• A remark of up to 100 characters can precede or follow an access list entry. |
| **Step 6** | **access-list** *access-list-number* **deny** *protocol* {*source* [*source-wildcard*] \| **any**} {*destination* [*destination-wildcard*] \| **any**} [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** \| **log-input**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Device(config)# access-list 107 deny tcp any any` | Denies any packet that matches all of the conditions specified in the statement.<br>• If the *source-wildcard* or *destination-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively. |

| | Command or Action | Purpose |
|---|---|---|
| | | • Optionally use the keyword **any** as a substitute for the *source source-wildcard* or *destination destination-wildcard* to specify the address and wildcard of 0.0.0.0 255.255.255.255. |
| **Step 7** | Repeat some combination of Steps 3 through 6 until you have specified the fields and values on which you want to base your access list. | Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list. |
| **Step 8** | **end**<br>**Example:**<br>`Device(config)# end` | Exits global configuration mode and enters privileged EXEC mode. |
| **Step 9** | **show ip access-list**<br>**Example:**<br>`Device# show ip access-list` | (Optional) Displays the contents of all current IP access lists. |

# Applying an Access List to an Interface

## SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ip access-group** {*access-list-number* | *access-list-name*} {**in** | **out**}
5. **end**

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface** *type number*<br>**Example:**<br>`Device(config)# interface TenGigabitEthernet4/1/0` | Specifies an interface and enters interface configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 4** | **ip access-group** {*access-list-number* \| *access-list-name*} {**in** \| **out**}<br>**Example:**<br>`Device(config-if)# ip access-group acl1 in` | Applies the specified access list to the inbound interface.<br>• To filter source addresses, apply the access list to the inbound interface. |
| **Step 5** | **end**<br>**Example:**<br>`Device(config-if)# end` | Exits interface configuration mode and returns to privileged EXEC mode. |

# Configuration Examples for Creating an IP Access List and Applying It to an Interface

## Example: Filtering on Host Source Address

In the following example, the workstation belonging to user1 is allowed access to Ten Gigabit Ethernet interface 4/1/0, and the workstation belonging to user2 is not allowed access:

```
interface TenGigabitEthernet4/1/0
 ip access-group workstations in
!
ip access-list standard workstations
 remark Permit only user1 workstation through
 permit 172.16.2.88
 remark Do not allow user2 workstation through
 deny 172.16.3.13
```

## Example: Filtering on Subnet Source Address

In the following example, the user1 subnet is not allowed access to Ten Gigabit Ethernet interface 4/1/0, but the Main subnet is allowed access:

```
interface TenGigabitEthernet4/1/0
 ip access-group prevention in
!
ip access-list standard prevention
 remark Do not allow user1 subnet through
 deny 172.22.0.0 0.0.255.255
 remark Allow Main subnet
 permit 172.25.0.0 0.0.255.255
```

## Example: Filtering on Source and Destination Addresses and IP Protocols

The following configuration example shows an interface with two access lists, one applied to outgoing packets and one applied to incoming packets. The standard access list named Internet-filter filters outgoing packets on source address. The only packets allowed out the interface must be from source 172.16.3.4.

The extended access list named marketing-group filters incoming packets. The access list permits Telnet packets from any source to network 172.26.0.0 and denies all other TCP packets. It permits any ICMP packets. It denies UDP packets from any source to network 172.26.0 0 on port numbers less than 1024. Finally, the access list denies all other IP packets and performs logging of packets passed or denied by that entry.

```
interface TenGigabitEthernet4/1/0
 ip address 172.20.5.1 255.255.255.0
 ip access-group Internet-filter out
 ip access-group marketing-group in
!
ip access-list standard Internet-filter
 permit 172.16.3.4
ip access-list extended marketing-group
 permit tcp any 172.26.0.0 0.0.255.255 eq telnet
 deny tcp any any
 permit icmp any any
 deny udp any 172.26.0.0 0.0.255.255 lt 1024
 deny ip any any
```

## Example: Filtering on Source Addresses Using a Numbered Access List

In the following example, network 10.0.0.0 is a Class A network whose second octet specifies a subnet; that is, its subnet mask is 255.255.0.0. The third and fourth octets of a network 10.0.0.0 address specify a particular host. Using access list 2, the Cisco IOS-XE software would accept one address on subnet 48 and reject all others on that subnet. The last line of the list shows that the software would accept addresses on all other network 10.0.0.0 subnets.

```
interface TenGigabitEthernet4/1/0
 ip access-group 2 in
!
access-list 2 permit 10.48.0.3
access-list 2 deny 10.48.0.0  0.0.255.255
access-list 2 permit 10.0.0.0  0.255.255.255
```

## Example: Preventing Telnet Access to a Subnet

In the following example, the user1 subnet is not allowed to telnet out of Ten Gigabit Ethernet interface 4/1/0:

```
interface TenGigabitEthernet4/1/0
 ip access-group telnetting out
!
ip access-list extended telnetting
 remark Do not allow user1 subnet to telnet out
 deny tcp 172.20.0.0 0.0.255.255 any eq telnet
 remark Allow Top subnet to telnet out
 permit tcp 172.33.0.0 0.0.255.255 any eq telnet
```

## Example: Filtering on TCP and ICMP Using Port Numbers

In the following example, the first line of the extended access list named acl1 permits any incoming TCP connections with destination ports greater than 1023. The second line permits incoming TCP connections to the Simple Mail Transfer Protocol (SMTP) port of host 172.28.1.2. The last line permits incoming ICMP messages for error feedback.

```
interface TenGigabitEthernet4/1/0
 ip access-group acl1 in
!
ip access-list extended acl1
 permit tcp any 172.28.0.0 0.0.255.255 gt 1023
 permit tcp any host 172.28.1.2 eq 25
 permit icmp any 172.28.0.0 255.255.255.255
```

## Example: Allowing SMTP E-mail and Established TCP Connections

Suppose you have a network connected to the Internet, and you want any host on an Ethernet to be able to form TCP connections to any host on the Internet. However, you do not want IP hosts to be able to form TCP connections to hosts on the Ten Gigabit Ethernet except to the mail (SMTP) port of a dedicated mail host.

SMTP uses TCP port 25 on one end of the connection and a random port number on the other end. The same two port numbers are used throughout the life of the connection. Mail packets coming in from the Internet will have a destination port of 25. Outbound packets will have the port numbers reversed. The fact that the secure system behind the router always will accept mail connections on port 25 is what makes possible separate control of incoming and outgoing services. The access list can be configured on either the outbound or inbound interface.

In the following example, the Ten Gigabit Ethernet network is a Class B network with the address 172.18.0.0, and the address of the mail host is 172.18.1.2. The **established** keyword is used only for the TCP protocol to indicate an established connection. A match occurs if the TCP datagram has the ACK or RST bits set, which indicate that the packet belongs to an existing connection.

```
interface TenGigabitEthernet4/1/0
 ip access-group 102 in
!
access-list 102 permit tcp any 172.18.0.0 0.0.255.255 established
access-list 102 permit tcp any host 172.18.1.2 eq 25
```

## Example: Preventing Access to the Web by Filtering on Port Name

In the following example, the w1and w2 workstations are not allowed web access; other hosts on network 172.20.0.0 are allowed web access:

```
interface TenGigabitEthernet4/1/0
 ip access-group no-web out
!
ip access-list extended no-web
 remark Do not allow w1 to browse the web
 deny host 172.20.3.85 any eq http
 remark Do not allow w2 to browse the web
 deny host 172.20.3.13 any eq http
 remark Allow others on our network to browse the web
 permit 172.20.0.0 0.0.255.255 any eq http
```

## Example: Filtering on Source Address and Logging the Packets

The following example defines access lists 1 and 2, both of which have logging enabled:

```
interface TenGigabitEthernet4/1/0
 ip address 172.16.1.1 255.0.0.0
 ip access-group 1 in
 ip access-group 2 out
!
access-list 1 permit 172.25.0.0 0.0.255.255 log
access-list 1 deny 172.30.0.0 0.0.255.255 log
!
access-list 2 permit 172.27.3.4 log
access-list 2 deny 172.17.0.0 0.0.255.255 log
```

If the interface receives 10 packets from 172.25.7.7 and 14 packets from 172.17.23.21, the first log will look like the following:

```
list 1 permit 172.25.7.7 1 packet
list 2 deny 172.17.23.21 1 packet
```

Five minutes later, the console will receive the following log:

```
list 1 permit 172.25.7.7 9 packets
list 2 deny 172.17.23.21 13 packets
```

# Example: Limiting Debug Output

The following sample configuration uses an access list to limit the **debug** command output. Limiting the **debug** output restricts the volume of data to what you are interested in, saving you time and resources.

```
Device(config)# ip access-list acl1
Device(config-std-nacl)# remark Displays only advertisements for LDP peer in acl1
Device(config-std-nacl)# permit host 10.0.0.44

Device# debug mpls ldp advertisements peer-acl acl1

tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.17.0.33
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.16.0.31
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.22.0.33
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.0.1
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.0.3
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.1.33
```

# Additional References Creating an IP Access List and Applying It to an Interface

### Related Documents

| Related Topic | Document Title |
|---|---|
| Security commands | • Cisco IOS Security Command Reference: Commands A to C<br>• Cisco IOS Security Command Reference: Commands D to L<br>• Cisco IOS Security Command Reference: Commands M to R<br>• Cisco IOS Security Command Reference: Commands S to Z |
| • Order of access list entries<br>• Access list entries based on time of day or week<br>• Packets with noninitial fragments | Refining an IP Access List |
| Filtering on IP options, TCP flags, or noncontiguous ports | Creating an IP Access List for Filtering |
| Controlling logging-related parameters | Understanding Access Control List Loggingl |

### Standards and RFCs

| Standard/RFC | Title |
|---|---|
| No new or modified standards or RFCs are supported by this feature, and support for existing standards or RFCs has not been modified by this feature. | — |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information Creating an IP Access List and Applying It to an Interface

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 20: Feature Information for Creating an IP Access List and Applying It to an Interface*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IP access lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |

CHAPTER 11

# Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports

This module describes how to use an IP access list to filter IP packets that contain certain IP Options, TCP flags, noncontiguous ports.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 21: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Prerequisites for Creating an IP Access List to Filter IP Options TCP Flags Noncontiguous Ports

Before you perform any of the tasks in this module, you should be familiar with the information in the following modules:

- “IP Access List Overview”
- “Creating an IP Access List and Applying It to an Interface”

# Information About Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports

## IP Options

IP uses four key mechanisms in providing its service: Type of Service, Time to Live, Options, and Header Checksum.

The Options, commonly referred to as IP Options, provide for control functions that are required in some situations but unnecessary for the most common communications. IP Options include provisions for time stamps, security, and special routing.

IP Options may or may not appear in datagrams. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular datagram, not their implementation. In some environments the security option may be required in all datagrams.

The option field is variable in length. There may be zero or more options. IP Options can have one of two formats:

- Format 1: A single octet of option-type.
- Format 2: An option-type octet, an option-length octet, and the actual option-data octets.

The option-length octet counts the option-type octet, the option-length octet, and the option-data octets.

The option-type octet is viewed as having three fields: a 1-bit copied flag, a 2-bit option class, and a 5-bit option number. These fields form an 8-bit value for the option type field. IP Options are commonly referred to by their 8-bit value.

For a complete list and description of IP Options, refer to RFC 791, *Internet Protocol* at the following URL: http://www.faqs.org/rfcs/rfc791.html

## Benefits of Filtering IP Options

- Filtering of packets that contain IP Options from the network relieves downstream devices and hosts of the load from options packets.
- This feature also minimizes load to the Route Processor (RP) for packets with IP Options that require RP processing on distributed systems. Previously, the packets were always routed to or processed by the RP CPU. Filtering the packets prevents them from impacting the RP.

## Benefits of Filtering on TCP Flags

The ACL TCP Flags Filtering feature provides a flexible mechanism for filtering on TCP flags. Previously, an incoming packet was matched as long as any TCP flag in the packet matched a flag specified in the access control entry (ACE). This behavior allows for a security loophole, because packets with all flags set could get past the access control list (ACL). The ACL TCP Flags Filtering feature allows you to select any combination of flags on which to filter. The ability to match on a flag set and on a flag not set gives you a greater degree of control for filtering on TCP flags, thus enhancing security.

Because TCP packets can be sent as false synchronization packets that can be accepted by a listening port, it is recommended that administrators of firewall devices set up some filtering rules to drop false TCP packets.

The ACEs that make up an access list can be configured to detect and drop unauthorized TCP packets by allowing only the packets that have a very specific group of TCP flags set or not set. The ACL TCP Flags Filtering feature provides a greater degree of packet-filtering control in the following ways:

- You can select any desired combination of TCP flags on which to filter TCP packets.
- You can configure ACEs to allow matching on a flag that is set, as well as on a flag that is not set.

# TCP Flags

The table below lists the TCP flags, which are further described in RFC 793, *Transmission Control Protocol*.

***Table 22: TCP Flags***

| TCP Flag | Purpose |
|---|---|
| ACK | Acknowledge flag—Indicates that the acknowledgment field of a segment specifies the next sequence number the sender of this segment is expecting to receive. |
| FIN | Finish flag—Used to clear connections. |
| PSH | Push flag—Indicates the data in the call should be immediately pushed through to the receiving user. |
| RST | Reset flag—Indicates that the receiver should delete the connection without further interaction. |
| SYN | Synchronize flag—Used to establish connections. |
| URG | Urgent flag—Indicates that the urgent field is meaningful and must be added to the segment sequence number. |

# Benefits of Using the Named ACL Support for Noncontiguous Ports on an Access Control Entry Feature

This feature greatly reduces the number of access control entries (ACEs) required in an access control list to handle multiple entries for the same source address, destination address, and protocol. If you maintain large numbers of ACEs, use this feature to consolidate existing groups of access list entries wherever it is possible and when you create new access list entries. When you configure access list entries with noncontiguous ports, you will have fewer access list entries to maintain.

# How Filtering on TTL Value Works

IP extended named and numbered access lists may filter on the TTL value of packets arriving at or leaving an interface. Packets with any possible TTL values 0 through 255 may be permitted or denied (filtered). Like filtering on other fields, such as source or destination address, the **ip access-group** command specifies **in** or **out**, which makes the access list ingress or egress and applies it to incoming or outgoing packets, respectively.

The TTL value is checked in conjunction with the specified protocol, application, and any other settings in the access list entry, and all conditions must be met.

### Special Handling for Packets with TTL Value of 0 or 1 Arriving at an Ingress Interface

The software switching paths—distributed Cisco Express Forwarding (dCEF), CEF, fast switching, and process switching—will usually permit or discard the packets based on the access list statements. However, when the TTL value of packets arriving at an ingress interface have a TTL of 0 or 1, special handling is required. The packets with a TTL value of 0 or 1 get sent to the process level before the ingress access list is checked in CEF, dCEF, or the fast switching paths. The ingress access list is applied to packets with TTL values 2 through 255 and a permit or deny decision is made.

Packets with a TTL value of 0 or 1 are sent to the process level because they will never be forwarded out of the device; the process level must check whether each packet is destined for the device and whether an Internet Control Message Protocol (ICMP) TTL Expire message needs to be sent back. This means that even if an ACL with TTL value 0 or 1 filtering is configured on the ingress interface with the intention to drop packets with a TTL of 0 or 1, the dropping of the packets will not happen in the faster paths. It will instead happen in the process level when the process applies the ACL. This is also true for hardware switching platforms. Packets with TTL value of 0 or 1 are sent to the process level of the route processor (RP) or Multilayer Switch Feature Card (MSFC).

On egress interfaces, access list filtering on TTL value works just like other access list features. The check will happen in the fastest switching path enabled in the device. This is because the faster switching paths handle all the TTL values (0 through 255) equally on the egress interface.

### Control Plane Policing for Filtering TTL Values 0 and 1

The special behavior for packets with a TTL value of 0 or 1 results in higher CPU usage for the device. If you are filtering on TTL value of 0 or 1, you should use control plane policing (CPP) to protect the CPU from being overwhelmed. In order to leverage CPP, you must configure an access list especially for filtering TTL values 0 and 1 and apply the access list through CPP. This access list will be a separate access list from any other interface access lists. Because CPP works for the entire system, not just on individual interfaces, you would need to configure only one such special access list for the entire device. This task is described in the section "Enabling Control Plane Policing to Filter on TTL Values 0 and 1".

## Benefits of Filtering on TTL Value

- Filtering on time-to-live (TTL) value provides a way to control which packets are allowed to reach the device or are prevented from reaching the device. By looking at your network layout, you can choose whether to accept or deny packets from a certain device based on how many hops away it is. For example, in a small network, you can deny packets from a location more than three hops away. Filtering on TTL value allows you to validate if the traffic originated from a neighboring device. You can accept only packets that reach you in one hop, for example, by accepting only packets with a TTL value of one less than the initial TTL value of a particular protocol.

- Many control plane protocols communicate only with their neighbors, but receive packets from everyone. By applying an access list that filters on TTL to receiving routers, you can block unwanted packets.

- The Cisco software sends all packets with a TTL value of 0 or 1 to the process level. The device must then send an Internet Control Message Protocol (ICMP) TTL value expire message to the source. By filtering packets that have a TTL value of 0 through 2, you can reduce the load on the process level.

# How to Create an IP Access List to Filter IP Options TCP Flags Noncontiguous Ports

## Filtering Packets That Contain IP Options

Complete these steps to configure an access list to filter packets that contain IP options and to verify that the access list has been configured correctly.

**Note**

- The ACL Support for Filtering IP Options feature can be used only with named, extended ACLs.
- Resource Reservation Protocol (RSVP) Multiprotocol Label Switching Traffic Engineering (MPLS TE), Internet Group Management Protocol Version 2 (IGMPV2), and other protocols that use IP options packets may not function in drop or ignore mode if this feature is configured.
- On most Cisco devices, a packet with IP options is not switched in hardware, but requires control plane software processing (primarily because there is a need to process the options and rewrite the IP header), so all IP packets with IP options will be filtered and switched in software.

**Step 1** **enable**

**Example:**

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

**Step 2** **configure terminal**

**Example:**

```
Device# configure terminal
```

Enters global configuration mode.

**Step 3** **ip access-list extended** *access-list-name*

**Example:**

```
Device(config)# ip access-list extended mylist1
```

Specifies the IP access list by name and enters named access list configuration mode.

**Step 4** [*sequence-number*] **deny** *protocol source source-wildcard destination destination-wildcard* [**option** *option-value*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]

**Example:**

```
Device(config-ext-nacl)# deny ip any any option traceroute
```

(Optional) Specifies a **deny** statement in named IP access list mode.

- This access list happens to use a **deny**statement first, but a **permit** statement could appear first, depending on the order of statements you need.

- Use the **option** keyword and *option-value* argument to filter packets that contain a particular IP Option.
- In this example, any packet that contains the traceroute IP option will be filtered out.
- Use the **no** *sequence-number* form of this command to delete an entry.

**Step 5** [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* [**option** *option-value*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]

**Example:**

```
Device(config-ext-nacl)# permit ip any any option security
```

Specifies a **permit** statement in named IP access list mode.

- In this example, any packet (not already filtered) that contains the security IP option will be permitted.
- Use the **no** *sequence-number* form of this command to delete an entry.

**Step 6** Repeat Step 4 or Step 5 as necessary.

Allows you to revise the access list.

**Step 7** **end**

**Example:**

```
Device(config-ext-nacl)# end
```

(Optional) Exits named access list configuration mode and returns to privileged EXEC mode.

**Step 8** **show ip access-lists** *access-list-name*

**Example:**

```
Device# show ip access-lists mylist1
```

(Optional) Displays the contents of the IP access list.

## What to Do Next

Apply the access list to an interface or reference it from a command that accepts an access list.

**Note** To effectively eliminate all packets that contain IP Options, we recommend that you configure the global **ip options drop** command.

# Filtering Packets That Contain TCP Flags

This task configures an access list to filter packets that contain TCP flags and verifies that the access list has been configured correctly.

**Note**

- TCP flag filtering can be used only with named, extended ACLs.
- The ACL TCP Flags Filtering feature is supported only for Cisco ACLs.
- Previously, the following command-line interface (CLI) format could be used to configure a TCP flag-checking mechanism:

**permit tcp any any rst** The following format that represents the same ACE can now be used: **permit tcp any any match-any +rst** Both the CLI formats are accepted; however, if the new keywords **match-all** or **match-any** are chosen, they must be followed by the new flags that are prefixed with "+" or "-". It is advisable to use only the old format or the new format in a single ACL. You cannot mix and match the old and new CLI formats.

**Caution** If a device having ACEs with the new syntax format is reloaded with a previous version of the Cisco software that does not support the ACL TCP Flags Filtering feature, the ACEs will not be applied, leading to possible security loopholes.

**Step 1** **enable**

**Example:**

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

**Step 2** **configure terminal**

**Example:**

```
Device# configure terminal
```

Enters global configuration mode.

**Step 3** **ip access-list extended** *access-list-name*

**Example:**

```
Device(config)# ip access-list extended kmd1
```

Specifies the IP access list by name and enters named access list configuration mode.

**Step 4** [*sequence-number*] **permit tcp** *source source-wildcard* [*operator* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established**|{**match-any** | **match-all**} {+ | -} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]

**Example:**

```
Device(config-ext-nacl)# permit tcp any any match-any +rst
```

Specifies a **permit** statement in named IP access list mode.

- This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need.
- Use the TCP command syntax of the **permit**command.
- Any packet with the RST TCP header flag set will be matched and allowed to pass the named access list kmd1 in Step 3.

**Step 5** [*sequence-number*] **deny tcp** *source source-wildcard* [*operator* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established**|{**match-any** | **match-all**} {+ | -} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]

**Example:**

```
Device(config-ext-nacl)# deny tcp any any match-all -ack -fin
```

(Optional) Specifies a **deny** statement in named IP access list mode.

- This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need.
- Use the TCP command syntax of the **deny**command.
- Any packet that does not have the ACK flag set, and also does not have the FIN flag set, will not be allowed to pass the named access list kmd1 in Step 3.
- See the **deny**(IP) command for additional command syntax to permit upper-layer protocols (ICMP, IGMP, TCP, and UDP).

**Step 6** Repeat Step 4 or Step 5 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number*command to delete an entry.

Allows you to revise the access list.

**Step 7** **end**

**Example:**

```
Device(config-ext-nacl)# end
```

(Optional) Exits the configuration mode and returns to privileged EXEC mode.

**Step 8** **show ip access-lists** *access-list-name*

**Example:**

```
Device# show ip access-lists kmd1
```

(Optional) Displays the contents of the IP access list.

- Review the output to confirm that the access list includes the new entry.

# Configuring an Access Control Entry with Noncontiguous Ports

Perform this task to create access list entries that use noncontiguous TCP or UDP port numbers. Although this task uses TCP ports, you could use the UDP syntax of the **permit** and **deny** commands to filter noncontiguous UDP ports.

Although this task uses a **permit** command first, use the **permit** and **deny** commands in the order that achieves your filtering goals.

**Note** The ACL—Named ACL Support for Noncontiguous Ports on an Access Control Entry feature can be used only with named, extended ACLs.

**Step 1** **enable**

**Example:**

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

**Step 2** **configure terminal**

**Example:**

```
Device# configure terminal
```

Enters global configuration mode.

**Step 3** **ip access-list extended** *access-list-name*

**Example:**

```
Device(config)# ip access-list extended acl-extd-1
```

Specifies the IP access list by name and enters named access list configuration mode.

**Step 4** [*sequence-number*] **permit tcp** *source source-wildcard* [*operator port* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established** {**match-any** | **match-all**} {**+** | **-**} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]

**Example:**

```
Device(config-ext-nacl)# permit tcp any eq telnet ftp any eq 450 679
```

Specifies a **permit** statement in named IP access list configuration mode.

- Operators include **lt** (less than), **gt** (greater than), **eq** (equal), **neq** (not equal), and **range** (inclusive range).
- If the operator is positioned after the source and source-wildcard arguments, it must match the source port. If the operator is positioned after the destination and destination-wildcard arguments, it must match the destination port.
- The **range** operator requires two port numbers. You can configure up to 10 ports after the **eq** and **neq**operators. All other operators require one port number.
- To filter UDP ports, use the UDP syntax of this command.

**Step 5** [*sequence-number*] **deny tcp** *source source-wildcard* [*operator port* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established** {**match-any** | **match-all**} {+ | -} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]

**Example:**

```
Device(config-ext-nacl)# deny tcp any neq 45 565 632 any
```

(Optional) Specifies a **deny** statement in named access list configuration mode.

- Operators include **lt** (less than), **gt** (greater than), **eq** (equal), **neq** (not equal), and **range** (inclusive range).
- If the *operator* is positioned after the *source* and *source-wildcard* arguments, it must match the source port. If the *operator* is positioned after the *destination* and *destination-wildcard* arguments, it must match the destination port.
- The **range** operator requires two port numbers. You can configure up to 10 ports after the **eq** and **neq**operators. All other operators require one port number.
- To filter UDP ports, use the UDP syntax of this command.

**Step 6** Repeat Step 4 or Step 5 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.

Allows you to revise the access list.

**Step 7** **end**

**Example:**

```
Device(config-ext-nacl)# end
```

(Optional) Exits named access list configuration mode and returns to privileged EXEC mode.

**Step 8** **show ip access-lists** *access-list-name*

**Example:**

```
Device# show ip access-lists kmd1
```

(Optional) Displays the contents of the access list.

# Consolidating Access List Entries with Noncontiguous Ports into One Access List Entry

Perform this task to consolidate a group of access list entries with noncontiguous ports into one access list entry.

Although this task uses TCP ports, you could use the UDP syntax of the **permit** and **deny** commands to filter noncontiguous UDP ports.

Although this task uses a **permit** command first, use the **permit** and **deny** commands in the order that achieves your filtering goals.

**Step 1** **enable**

**Example:**

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

**Step 2** **show ip access-lists** *access-list-name*

**Example:**

```
Device# show ip access-lists mylist1
```

(Optional) Displays the contents of the IP access list.

- Review the output to see if you can consolidate any access list entries.

**Step 3** **configure terminal**

**Example:**

```
Device# configure terminal
```

Enters global configuration mode.

**Step 4** **ip access-list extended** *access-list-name*

**Example:**

```
Device(config)# ip access-list extended mylist1
```

Specifies the IP access list by name and enters named access list configuration mode.

**Step 5** **no** [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard*[**option** *option-name*] [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]

**Example:**

```
Device(config-ext-nacl)# no 10
```

Removes the redundant access list entry that can be consolidated.

- Repeat this step to remove entries to be consolidated because only the port numbers differ.
- After this step is repeated to remove the access list entries 20, 30, and 40, for example, those entries are removed because they will be consolidated into one **permit** statement.
- If a *sequence-number* is specified, the rest of the command syntax is optional.

**Step 6** [*sequence-number*] **permit** *protocol source source-wildcard*[*operator port*[*port*]] *destination destination-wildcard*[*operator port*[*port*]] [**option** *option-name*] [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]

**Example:**

```
Device(config-ext-nacl)# permit tcp any neq 45 565 632 any eq 23 45 34 43
```

Specifies a **permit** statement in named access list configuration mode.

- In this instance, a group of access list entries with noncontiguous ports was consolidated into one **permit** statement.
- You can configure up to 10 ports after the **eq** and **neq** operators.

**Step 7** Repeat Steps 5 and 6 as necessary, adding **permit** or **deny** statements to consolidate access list entries where possible. Use the **no** *sequence-number* command to delete an entry.

Allows you to revise the access list.

**Step 8** **end**

**Example:**

```
Device(config-std-nacl)# end
```

(Optional) Exits named access list configuration mode and returns to privileged EXEC mode.

**Step 9** **show ip access-lists** *access-list-name*

**Example:**

```
Device# show ip access-lists mylist1
```

(Optional) Displays the contents of the access list.

## What To Do Next

Apply the access list to an interface or reference it from a command that accepts an access list.

# Filtering Packets Based on TTL Value

Because access lists are very flexible, it is not possible to define only one combination of **permit** and **deny** commands to filter packets based on the TTL value. This task illustrates just one example that achieves TTL filtering. Configure the appropriate **permit** and **deny** statements that will accomplish your filtering plan.

**Note** When the access list specifies the operation EQ or NEQ, depending on the Cisco software release in use on the device, the access lists can specify up to ten TTL values. The number of TTL values can vary by the Cisco software release.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *access-list-name*
4. [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard*[**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**ttl** *operator value*] [**log**] [**time-range** *time-range-name*] [**fragments**]
5. Continue to add **permit** or **deny** statements to achieve the filtering you want.
6. **exit**
7. **interface** *type number*
8. **ip access-group** *access-list-name* {**in** | **out**}

### DETAILED STEPS

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

| | Command or Action | Purpose |
|---|---|---|
| | **Example:**<br>`Device> enable` | • Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip access-list extended** *access-list-name*<br>**Example:**<br>`Device(config)# ip access-list extended ttlfilter` | Defines an IP access list by name.<br>• An access list that filters on TTL value must be an extended access list. |
| **Step 4** | [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard*[**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**ttl** *operator value*] [**log**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Device(config-ext-nacl)# permit ip host 172.16.1.1 any ttl lt 2` | Sets conditions to allow a packet to pass a named IP access list.<br>• Every access list must have at least one **permit** statement.<br>• This example permits packets from source 172.16.1.1 to any destination with a TTL value less than 2. |
| **Step 5** | Continue to add **permit** or **deny** statements to achieve the filtering you want. | -- |
| **Step 6** | **exit**<br>**Example:**<br>`Device(config-ext-nacl)# exit` | Exits any configuration mode to the next highest mode in the command-line interface (CLI) mode hierarchy. |
| **Step 7** | **interface** *type number*<br>**Example:**<br>`Device(config)# interface TenGigabitEthernet4/1/0` | Configures an interface type and enters interface configuration mode. |
| **Step 8** | **ip access-group** *access-list-name* {**in** \| **out**}<br>**Example:**<br>`Device(config-if)# ip access-group ttlfilter in` | Applies the access list to an interface. |

# Enabling Control Plane Policing to Filter on TTL Values 0 and 1

Perform this task to filter IP packets based on a TTL value of 0 or 1 and to protect the CPU from being overwhelmed. This task configures an access list for classification on TTL value 0 and 1, configures the Modular QoS Command-Line Interface (CLI) (MQC), and applies a policy map to the control plane. Any packets that pass the access list are dropped. This special access list is separate from any other interface access lists.

Because access lists are very flexible, it is not possible to define only one combination of **permit** and **deny** commands to filter packets based on the TTL value. This task illustrates just one example that achieves TTL filtering. Configure the appropriate **permit** and **deny** statements that will accomplish your filtering plan.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *access-list-name*
4. [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* **ttl** *operator* value
5. Continue to add **permit** or **deny** statements to achieve the filtering you want.
6. **exit**
7. **class-map** *class-map-name* [**match-all** | **match-any**]
8. **match access-group** {*access-group* | **name** *access-group-name*}
9. **exit**
10. **policy-map** *policy-map-name*
11. **class** {*class-name* | **class-default**}
12. **drop**
13. **exit**
14. **exit**
15. **control-plane**
16. **service-policy** {**input** | **output**} *policy-map-name*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip access-list extended** *access-list-name*<br>**Example:**<br>`Device(config)# ip access-list extended ttlfilter` | Defines an IP access list by name.<br>• An access list that filters on a TTL value must be an extended access list. |
| **Step 4** | [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* **ttl** *operator* value<br>**Example:** | Sets conditions to allow a packet to pass a named IP access list.<br>• Every access list must have at least one **permit** statement. |

| | Command or Action | Purpose |
|---|---|---|
| | `Device(config-ext-nacl)# permit ip host 172.16.1.1 any ttl lt 2` | • This example permits packets from source 172.16.1.1 to any destination with a TTL value less than 2. |
| **Step 5** | Continue to add **permit** or **deny** statements to achieve the filtering you want. | The packets that pass the access list will be dropped. |
| **Step 6** | **exit**<br>**Example:**<br>`Device(config-ext-nacl)# exit` | Exits any configuration mode to the next highest mode in the CLI mode hierarchy. |
| **Step 7** | **class-map** *class-map-name* [**match-all** \| **match-any**]<br>**Example:**<br>`Device(config)# class-map acl-filtering` | Creates a class map to be used for matching packets to a specified class. |
| **Step 8** | **match access-group** {*access-group* \| **name** *access-group-name*}<br>**Example:**<br>`Device(config-cmap)# match access-group name ttlfilter` | Configures the match criteria for a class map on the basis of the specified access control list. |
| **Step 9** | **exit**<br>**Example:**<br>`Device(config-cmap)# exit` | Exits any configuration mode to the next highest mode in the CLI mode hierarchy. |
| **Step 10** | **policy-map** *policy-map-name*<br>**Example:**<br>`Device(config)# policy-map acl-filter` | Creates or modifies a policy map that can be attached to one or more interface to specify a service policy. |
| **Step 11** | **class** {*class-name* \| **class-default**}<br>**Example:**<br>`Device(config-pmap)# class acl-filter-class` | Specifies the name of the class whose policy you want to create or change or to specify the default class (commonly known as the class-default class) before you configure its policy. |
| **Step 12** | **drop**<br>**Example:**<br>`Device(config-pmap-c)# drop` | Configures a traffic class to discard packets belonging to a specific class. |
| **Step 13** | **exit**<br>**Example:**<br>`Device(config-pmap-c)# exit` | Exits any configuration mode to the next highest mode in the CLI mode hierarchy. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 14** | **exit**<br>**Example:**<br>`Device(config-pmap)# exit` | Exits any configuration mode to the next highest mode in the CLI mode hierarchy. |
| **Step 15** | **control-plane**<br>**Example:**<br>`Device(config)# control-plane` | Associates or modifies attributes or parameters that are associated with the control plane of the device. |
| **Step 16** | **service-policy** {**input** \| **output**} *policy-map-name*<br>**Example:**<br>`Device(config-cp)# service-policy input acl-filter` | Attaches a policy map to a control plane for aggregate control plane services. |

# Configuration Examples for Filtering IP Options, TCP Flags, Noncontiguous Ports

## Example: Filtering Packets That Contain IP Options

The following example shows an extended access list named mylist2 that contains access list entries (ACEs) that are configured to permit TCP packets only if they contain the IP Options that are specified in the ACEs:

```
ip access-list extended mylist2
 10 permit ip any any option eool
 20 permit ip any any option record-route
 30 permit ip any any option zsu
 40 permit ip any any option mtup
```

The **show access-list** command has been entered to show how many packets were matched and therefore permitted:

```
Device# show ip access-list mylist2
Extended IP access list test
10 permit ip any any option eool (1 match)
20 permit ip any any option record-route (1 match)
30 permit ip any any option zsu (1 match)
40 permit ip any any option mtup (1 match)
```

## Example: Filtering Packets That Contain TCP Flags

The following access list allows TCP packets only if the TCP flags ACK and SYN are set and the FIN flag is not set:

```
ip access-list extended aaa
```

```
 permit tcp any any match-all +ack +syn -fin
 end
```

The **show access-list** command has been entered to display the ACL:

```
Device# show access-list aaa

Extended IP access list aaa
 10 permit tcp any any match-all +ack +syn -fin
```

## Example: Creating an Access List Entry with Noncontiguous Ports

The following access list entry can be created because up to ten ports can be entered after the **eq** and **neq** operators:

```
ip access-list extended aaa
 permit tcp any eq telnet ftp any eq 23 45 34
 end
```

Enter the **show access-lists** command to display the newly created access list entry.

```
Device# show access-lists aaa

Extended IP access list aaa
 10 permit tcp any eq telnet ftp any eq 23 45 34
```

## Example: Consolidating Some Existing Access List Entries into One Access List Entry with Noncontiguous Ports

The **show access-lists** command is used to display a group of access list entries for the access list named abc:

```
Device# show access-lists abc
Extended IP access list abc
 10 permit tcp any eq telnet any eq 450
 20 permit tcp any eq telnet any eq 679
 30 permit tcp any eq ftp any eq 450
 40 permit tcp any eq ftp any eq 679
```

Because the entries are all for the same **permit** statement and simply show different ports, they can be consolidated into one new access list entry. The following example shows the removal of the redundant access list entries and the creation of a new access list entry that consolidates the previously displayed group of access list entries:

```
ip access-list extended abc
 no 10
 no 20
 no 30
 no 40
 permit tcp any eq telnet ftp any eq 450 679
 end
```

When the **show access-lists** command is reentered, the consolidated access list entry is displayed:

```
Device# show access-lists abc
```

```
Extended IP access list abc
 10 permit tcp any eq telnet ftp any eq 450 679
```

## Example: Filtering on TTL Value

The following access list filters IP packets containing type of service (ToS) level 3 with time-to-live (TTL) values 10 and 20. It also filters IP packets with a TTL greater than 154 and applies that rule to noninitial fragments. It permits IP packets with a precedence level of flash and a TTL value not equal to 1, and it sends log messages about such packets to the console. All other packets are denied.

```
ip access-list extended incomingfilter
 deny ip any any tos 3 ttl eq 10 20
 deny ip any any ttl gt 154 fragments
 permit ip any any precedence flash ttl neq 1 log
!
interface TenGigabitEthernet4/1/0

ip access-group incomingfilter in
```

## Example: Control Plane Policing to Filter on TTL Values 0 and 1

The following example configures a traffic class called acl-filter-class for use in a policy map called acl-filter. An access list permits IP packets from any source having a time-to-live (TTL) value of 0 or 1. Any packets matching the access list are dropped. The policy map is attached to the control plane.

```
ip access-list extended ttlfilter

 permit ip any any ttl eq 0 1

class-map acl-filter-class

 match access-group name ttlfilter

policy-map acl-filter

 class acl-filter-class

 drop

control-plane

 service-policy input acl-filter
```

# Additional References

### Related Documents

| Related Topic | Document Title |
|---|---|
| Security commands | *Cisco IOS Security Command Reference* |
| Configuring the device to drop or ignore packets containing IP Options by using the **no ip options** command. | *ACL IP Options Selective Drop* |
| Overview information about access lists. | *IP Access List Overview* |
| Information about creating an IP access list and applying it to an interface | *Creating an IP Access List and Applying It to an Interface* |
| QoS commands | *Cisco IOS Quality of Service Solutions Command Reference* |

### RFCs

| RFC | Title |
|---|---|
| RFC 791 | *Internet Protocol*<br>http://www.faqs.org/rfcs/rfc791.html |
| RFC 793 | *Transmission Control Protocol* |
| RFC 1393 | *Traceroute Using an IP Option* |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

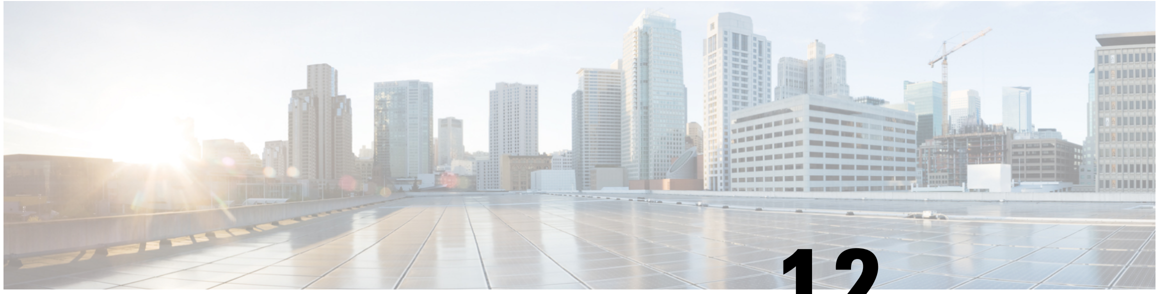
# Feature Information for Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports, or TTL Values

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 23: Feature Information for Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports, or TTL Values*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IP access lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on the Cisco cBR Series Converged Broadband Routers. |



CHAPTER 12

# Refining an IP Access List

There are several ways to refine an access list while or after you create it. You can change the order of the entries in an access list or add entries to an access list. You can restrict access list entries to a certain time of day or week, or achieve finer granularity when filtering packets by filtering noninitial fragments of packets.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 24: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information About Refining an IP Access List

## Access List Sequence Numbers

The ability to apply sequence numbers to IP access list entries simplifies access list changes. Prior to the IP Access List Entry Sequence Numbering feature, there was no way to specify the position of an entry within an access list. If you wanted to insert an entry in the middle of an existing list, all of the entries after the desired position had to be removed, then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

Sequence numbers allow users to add access list entries and resequence them. When you add a new entry, you specify the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry.

# Benefits of Access List Sequence Numbers

An access list sequence number is a number at the beginning of a **permit** or **deny** command in an access list. The sequence number determines the order that the entry appears in the access list. The ability to apply sequence numbers to IP access list entries simplifies access list changes.

Prior to having sequence numbers, users could only add access list entries to the end of an access list; therefore, needing to add statements anywhere except the end of the list required reconfiguring the entire access list. There was no way to specify the position of an entry within an access list. If a user wanted to insert an entry (statement) in the middle of an existing list, all of the entries after the desired position had to be removed, then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

This feature allows users to add sequence numbers to access list entries and resequence them. When a user adds a new entry, the user chooses the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry. Sequence numbers make revising an access list much easier.

# Sequence Numbering Behavior

- For backward compatibility with previous releases, if entries with no sequence numbers are applied, the first entry is assigned a sequence number of 10, and successive entries are incremented by 10. The maximum sequence number is 2147483647. If the generated sequence number exceeds this maximum number, the following message is displayed:

```
Exceeded maximum sequence number.
```

- If the user enters an entry without a sequence number, it is assigned a sequence number that is 10 greater than the last sequence number in that access list and is placed at the end of the list.
- If the user enters an entry that matches an already existing entry (except for the sequence number), then no changes are made.
- If the user enters a sequence number that is already present, the following error message is generated:

```
Duplicate sequence number.
```

- If a new access list is entered from global configuration mode, then sequence numbers for that access list are generated automatically.
- Sequence numbers are not nvgened. That is, the sequence numbers themselves are not saved. In the event that the system is reloaded, the configured sequence numbers revert to the default sequence starting number and increment. The function is provided for backward compatibility with software releases that do not support sequence numbering.
- This feature works with named and numbered, standard and extended IP access lists.

# Benefits of Time Ranges

Benefits and possible uses of time ranges include the following:

- The network administrator has more control over permitting or denying a user access to resources. These resources could be an application (identified by an IP address/mask pair and a port number), policy routing, or an on-demand link (identified as interesting traffic to the dialer).
- Network administrators can set time-based security policy, including the following:
  - Perimeter security using access lists
  - Data confidentiality with IP Security Protocol (IPsec)
- When provider access rates vary by time of day, it is possible to automatically reroute traffic cost effectively.
- Network administrators can control logging messages. Access list entries can log traffic at certain times of the day, but not constantly. Therefore, administrators can simply deny access without needing to analyze many logs generated during peak hours.

# Benefits Filtering Noninitial Fragments of Packets

Filter noninitial fragments of packets with an extended access list if you want to block more of the traffic you intended to block, not just the initial fragment of such packets. You should first understand the following concepts.

If the **fragments**keyword is used in additional IP access list entries that deny fragments, the fragment control feature provides the following benefits:

### Additional Security

You are able to block more of the traffic you intended to block, not just the initial fragment of such packets. The unwanted fragments no longer linger at the receiver until the reassembly timeout is reached because they are blocked before being sent to the receiver. Blocking a greater portion of unwanted traffic improves security and reduces the risk from potential hackers.

### Reduced Cost

By blocking unwanted noninitial fragments of packets, you are not paying for traffic you intended to block.

### Reduced Storage

By blocking unwanted noninitial fragments of packets from ever reaching the receiver, that destination does not have to store the fragments until the reassembly timeout period is reached.

### Expected Behavior Is Achieved

The noninitial fragments will be handled in the same way as the initial fragment, which is what you would expect. There are fewer unexpected policy routing results and fewer fragments of packets being routed when they should not be.

# Access List Processing of Fragments

The behavior of access list entries regarding the use or lack of use of the **fragments** keyword can be summarized as follows:

| If the Access-List Entry Has... | Then... |
|---|---|
| ...no **fragments** keyword (the default), and assuming all of the access-list entry information matches, | For an access list entry that contains only Layer 3 information:<br>• The entry is applied to nonfragmented packets, initial fragments, and noninitial fragments.<br>For an access list entry that contains Layer 3 and Layer 4 information:<br>• The entry is applied to nonfragmented packets and initial fragments.<br>  • If the entry is a **permit** statement, then the packet or fragment is permitted.<br>  • If the entry is a **deny** statement, then the packet or fragment is denied.<br>• The entry is also applied to noninitial fragments in the following manner. Because noninitial fragments contain only Layer 3 information, only the Layer 3 portion of an access list entry can be applied. If the Layer 3 portion of the access list entry matches, and<br>  • If the entry is a **permit** statement, then the noninitial fragment is permitted.<br>  • If the entry is a **deny** statement, then the next access list entry is processed.<br>**Note** The **deny** statements are handled differently for noninitial fragments versus nonfragmented or initial fragments. |
| ...the **fragments** keyword, and assuming all of the access-list entry information matches, | The access list entry is applied only to noninitial fragments.<br>The **fragments** keyword cannot be configured for an access list entry that contains any Layer 4 information. |

Be aware that you should not add the **fragments** keyword to every access list entry because the first fragment of the IP packet is considered a nonfragment and is treated independently of the subsequent fragments. An initial fragment will not match an access list **permit** or **deny** entry that contains the **fragments** keyword. The packet is compared to the next access list entry, and so on, until it is either permitted or denied by an access list entry that does not contain the **fragments** keyword. Therefore, you may need two access list entries for every **deny** entry. The first **deny** entry of the pair will not include the **fragments** keyword and applies to the initial fragment. The second **deny** entry of the pair will include the **fragments** keyword and applies to the subsequent fragments. In the cases in which there are multiple **deny** entries for the same host but with different Layer 4 ports, a single **deny** access list entry with the **fragments** keyword for that host is all that needs to be added. Thus all the fragments of a packet are handled in the same manner by the access list.

Packet fragments of IP datagrams are considered individual packets, and each counts individually as a packet in access list accounting and access list violation counts.

# How to Refine an IP Access List

The tasks in this module provide you with various ways to refine an access list if you did not already do so while you were creating it. You can change the order of the entries in an access list, add entries to an access list, restrict access list entries to a certain time of day or week, or achieve finer granularity when filtering packets by filtering on noninitial fragments of packets.

## Revising an Access List Using Sequence Numbers

Perform this task if you want to add entries to an existing access list, change the order of entries, or simply number the entries in an access list to accommodate future changes.

**Note** Remember that if you want to delete an entry from an access list, you can simply use the **no deny** or **no permit** form of the command, or the **no** *sequence-number* command if the statement already has a sequence number.

**Note**

- Access list sequence numbers do not support dynamic, reflexive, or firewall access lists.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list resequence** *access-list-name starting-sequence-number increment*
4. **ip access-list** {**standard**| **extended**} *access-list-name*
5. Do one of the following:
   - *sequence-number* **permit** *source source-wildcard*
   - *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
6. Do one of the following:
   - *sequence-number* **deny** *source source-wildcard*
   - *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
7. Repeat Step 5 and Step 6 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
8. **end**
9. **show ip access-lists** *access-list-name*

### DETAILED STEPS

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

| | Command or Action | Purpose |
|---|---|---|
| | **Example:**<br>`Router> enable` | • Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip access-list resequence** *access-list-name starting-sequence-number increment*<br>**Example:**<br>`Router(config)# ip access-list resequence kmd1 100 15` | Resequences the specified IP access list using the starting sequence number and the increment of sequence numbers.<br>• This example resequences an access list named kmd1. The starting sequence number is 100 and the increment is 15. |
| **Step 4** | **ip access-list** {**standard**\| **extended**} *access-list-name*<br>**Example:**<br>`Router(config)# ip access-list standard xyz123` | Specifies the IP access list by name and enters named access list configuration mode.<br>• If you specify **standard**, make sure you specify subsequent **permit** and **deny** statements using the standard access list syntax.<br>• If you specify **extended**, make sure you specify subsequent **permit** and **deny** statements using the extended access list syntax. |
| **Step 5** | Do one of the following:<br>• *sequence-number* **permit** *source source-wildcard*<br>• *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Router(config-std-nacl)# 105 permit 10.5.5.5 0.0.0.255` | Specifies a permit statement in named IP access list mode.<br>• This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need.<br>• See the **permit** (IP) command for additional command syntax to permit upper layer protocols (ICMP, IGMP, TCP, and UDP).<br>• Use the **no** *sequence-number* command to delete an entry.<br>• As the prompt indicates, this access list was a standard access list. If you had specified **extended** in Step 4, the prompt for this step would be Router(config-ext-nacl)# and you would use the extended **permit** command syntax. |
| **Step 6** | Do one of the following:<br>• *sequence-number* **deny** *source source-wildcard*<br>• *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* | (Optional) Specifies a deny statement in named IP access list mode.<br>• This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need. |

| | Command or Action | Purpose |
|---|---|---|
| | [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Router(config-std-nacl)# 110 deny 10.6.6.7 0.0.0.255` | • See the **deny** (IP) command for additional command syntax to permit upper layer protocols (ICMP, IGMP, TCP, and UDP).<br>• Use the **no** *sequence-number* command to delete an entry.<br>• As the prompt indicates, this access list was a standard access list. If you had specified **extended** in Step 4, the prompt for this step would be Router(config-ext-nacl)# and you would use the extended **deny** command syntax. |
| **Step 7** | Repeat Step 5 and Step 6 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry. | Allows you to revise the access list. |
| **Step 8** | **end**<br>**Example:**<br>`Router(config-std-nacl)# end` | (Optional) Exits the configuration mode and returns to privileged EXEC mode. |
| **Step 9** | **show ip access-lists** *access-list-name*<br>**Example:**<br>`Router# show ip access-lists xyz123` | (Optional) Displays the contents of the IP access list.<br>• Review the output to see that the access list includes the new entry. |

### Examples

The following is sample output from the **show ip access-lists** command when the **xyz123** access list is specified.

```
Router# show ip access-lists xyz123
Standard IP access list xyz123
100 permit 10.4.4.0, wildcard bits 0.0.0.255
105 permit 10.5.5.5, wildcard bits 0.0.0.255
115 permit 10.0.0.0, wildcard bits 0.0.0.255
130 permit 10.5.5.0, wildcard bits 0.0.0.255
145 permit 10.0.0.0, wildcard bits 0.0.0.255
```

# Restricting an Access List Entry to a Time of Day or Week

By default, access list statements are always in effect once they are applied. However, you can define the times of the day or week that **permit** or **deny** statements are in effect by defining a time range, and then referencing the time range by name in an individual access list statement. IP and Internetwork Packet Exchange (IPX) named or numbered extended access lists can use time ranges.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *name*
4. [*sequence-number*] **deny** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]]
5. [*sequence-number*] **deny** *protocol source*[*source-wildcard*][*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]] **fragments**
6. [*sequence-number*] **permit** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]]
7. Repeat some combination of Steps 4 through 6 until you have specified the values on which you want to base your access list.
8. **end**
9. **show ip access-list**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip access-list extended** *name*<br>**Example:**<br>`Router(config)# ip access-list extended rstrct4` | Defines an extended IP access list using a name and enters extended named access list configuration mode. |
| **Step 4** | [*sequence-number*] **deny** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]]<br>**Example:**<br>`Router(config-ext-nacl)# deny ip any 172.20.1.1` | (Optional) Denies any packet that matches all of the conditions specified in the statement.<br>• This statement will apply to nonfragmented packets and initial fragments. |
| **Step 5** | [*sequence-number*] **deny** *protocol source*[*source-wildcard*][*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]] **fragments**<br>**Example:**<br>`Router(config-ext-nacl)# deny ip any 172.20.1.1 fragments` | (Optional) Denies any packet that matches all of the conditions specified in the statement<br>• This statement will apply to noninitial fragments. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 6** | [*sequence-number*] **permit** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]]<br>**Example:**<br>`Router(config-ext-nacl)# permit tcp any any` | Permits any packet that matches all of the conditions specified in the statement.<br>• Every access list needs at least one **permit** statement.<br>• If the *source-wildcard* or *destination-wildcard*isomitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively.<br>• Optionally use the keyword **any** as a substitute for the *source source-wildcard*or *destination destination-wildcard*to specify the address and wildcard of 0.0.0.0 255.255.255.255. |
| **Step 7** | Repeat some combination of Steps 4 through 6 until you have specified the values on which you want to base your access list. | Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list. |
| **Step 8** | **end**<br>**Example:**<br>`Router(config-ext-nacl)# end` | Ends configuration mode and returns the system to privileged EXEC mode. |
| **Step 9** | **show ip access-list**<br>**Example:**<br>`Router# show ip access-list` | (Optional) Displays the contents of all current IP access lists. |

## What to Do Next

Apply the access list to an interface or reference it from a command that accepts an access list.

**Note** To effectively eliminate all packets that contain IP Options, we recommend that you configure the global **ip options drop** command.

# Configuration Examples for Refining an IP Access List

## Example Resequencing Entries in an Access List

The following example shows an access list before and after resequencing. The starting value is 1, and increment value is 2. The subsequent entries are ordered based on the increment values that users provide, and the range is from 1 to 2147483647.

When an entry with no sequence number is entered, by default it has a sequence number of 10 more than the last entry in the access list.

```
Router# show access-list carls
Extended IP access list carls
    10 permit ip host 10.3.3.3 host 172.16.5.34
    20 permit icmp any any
    30 permit tcp any host 10.3.3.3
    40 permit ip host 10.4.4.4 any
    50 Dynamic test permit ip any any
    60 permit ip host 172.16.2.2 host 10.3.3.12
    70 permit ip host 10.3.3.3 any log
    80 permit tcp host 10.3.3.3 host 10.1.2.2
    90 permit ip host 10.3.3.3 any
    100 permit ip any any
Router(config)# ip access-list extended carls
Router(config)# ip access-list resequence carls 1 2
Router(config)# end
Router# show access-list carls
Extended IP access list carls
    1 permit ip host 10.3.3.3 host 172.16.5.34
    3 permit icmp any any
    5 permit tcp any host 10.3.3.3
    7 permit ip host 10.4.4.4 any
    9 Dynamic test permit ip any any
    11 permit ip host 172.16.2.2 host 10.3.3.12
    13 permit ip host 10.3.3.3 any log
    15 permit tcp host 10.3.3.3 host 10.1.2.2
    17 permit ip host 10.3.3.3 any
    19 permit ip any any
```

# Example Adding an Entry with a Sequence Number

In the following example, an new entry (sequence number 15) is added to an access list:

```
Router# show ip access-list
Standard IP access list tryon
2 permit 10.4.4.2, wildcard bits 0.0.255.255
5 permit 10.0.0.44, wildcard bits 0.0.0.255
10 permit 10.0.0.1, wildcard bits 0.0.0.255
20 permit 10.0.0.2, wildcard bits 0.0.0.255
Router(config)# ip access-list standard tryon
Router(config-std-nacl)# 15 permit 10.5.5.5 0.0.0.255
Router# show ip access-list
Standard IP access list tryon
2 permit 10.4.0.0, wildcard bits 0.0.255.255
5 permit 10.0.0.0, wildcard bits 0.0.0.255
10 permit 10.0.0.0, wildcard bits 0.0.0.255
15 permit 10.5.5.0, wildcard bits 0.0.0.255
20 permit 10.0.0.0, wildcard bits 0.0.0.255
```

# Example Adding an Entry with No Sequence Number

The following example shows how an entry with no specified sequence number is added to the end of an access list. When an entry is added without a sequence number, it is automatically given a sequence number that puts it at the end of the access list. Because the default increment is 10, the entry will have a sequence number 10 higher than the last entry in the existing access list.

```
Router(config)# ip access-list standard resources
Router(config-std-nacl)# permit 10.1.1.1 0.0.0.255
Router(config-std-nacl)# permit 10.2.2.2 0.0.0.255
Router(config-std-nacl)# permit 10.3.3.3 0.0.0.255
Router# show access-list
Standard IP access list resources
10 permit 10.1.1.1, wildcard bits 0.0.0.255
20 permit 10.2.2.2, wildcard bits 0.0.0.255
30 permit 10.3.3.3, wildcard bits 0.0.0.255
Router(config)# ip access-list standard resources
Router(config-std-nacl)# permit 10.4.4.4 0.0.0.255
Router(config-std-nacl)# end
Router# show access-list
Standard IP access list resources
10 permit 10.1.1.1, wildcard bits 0.0.0.255
20 permit 10.2.2.2, wildcard bits 0.0.0.255
30 permit 10.3.3.3, wildcard bits 0.0.0.255
40 permit 10.4.4.4, wildcard bits 0.0.0.255
```

# Example Time Ranges Applied to IP Access List Entries

The following example creates a time range called no-http, which extends from Monday to Friday from 8:00 a.m. to 6:00 p.m. That time range is applied to the **deny** statement, thereby denying HTTP traffic on Monday through Friday from 8:00 a.m. to 6:00 p.m.

The time range called udp-yes defines weekends from noon to 8:00 p.m. That time range is applied to the **permit** statement, thereby allowing UDP traffic on Saturday and Sunday from noon to 8:00 p.m. only. The access list containing both statements is applied to inbound packets on Ten Gigabit Ethernet interface 4/1/0.

```
time-range no-http
 periodic weekdays 8:00 to 18:00
!
time-range udp-yes
 periodic weekend 12:00 to 20:00
!
ip access-list extended strict
 deny tcp any any eq http time-range no-http
 permit udp any any time-range udp-yes
!
interface TenGigabitEthernet4/1/0
 ip access-group strict in
```

# Example Filtering IP Packet Fragments

In the following access list, the first statement will deny only noninitial fragments destined for host 172.16.1.1. The second statement will permit only the remaining nonfragmented and initial fragments that are destined for host 172.16.1.1 TCP port 80. The third statement will deny all other traffic. In order to block noninitial fragments for any TCP port, we must block noninitial fragments for all TCP ports, including port 80 for host 172.16.1.1. That is, non-initial fragments will not contain Layer 4 port information, so, in order to block such traffic for a given port, we have to block fragments for all ports.

```
access-list 101 deny ip any host 172.16.1.1 fragments
access-list 101 permit tcp any host 172.16.1.1 eq 80
access-list 101 deny ip any any
```

# Additional References

### Related Documents

| Related Topic | Document Title |
|---|---|
| Using the **time-range** command to establish time ranges | The chapter *Performing Basic System Management* in the *Cisco IOS XE Network Management Configuration Guide* |
| Network management command descriptions | *Cisco IOS Network Management Command Reference* |

### Standards

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

### MIBs

| MIB | MIBs Link |
|---|---|
| No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature. | To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL:<br>http://www.cisco.com/go/mibs |

### RFCs

| RFC | Title |
|---|---|
| No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature. | -- |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for Refining an IP Access List

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 25: Feature Information for Refining an IP Access List*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IP access lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |

CHAPTER 13

# IP Named Access Control Lists

Access control lists (ACLs) perform packet filtering to control the movement of packets through a network. Packet filtering provides security by limiting the access of traffic into a network, restricting user and device access to a network, and preventing traffic from leaving a network. IP access lists reduce the chance of spoofing and denial-of-service attacks, and allow dynamic, temporary user-access through a firewall.

The IP Named Access Control Lists feature gives network administrators the option of using names to identify their access lists.

This module describes IP named access lists and how to configure them.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 26: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information About IP Named Access Control Lists

## Definition of an Access List

Access control lists (ACLs) perform packet filtering to control the movement of packets through a network. Packet filtering provides security by limiting the access of traffic into a network, restricting user and device access to a network, and preventing traffic from leaving a network. IP access lists reduce the chance of spoofing and denial-of-service attacks, and allow dynamic, temporary user-access through a firewall.

IP access lists can also be used for purposes other than security, such as to control bandwidth, restrict the content of routing updates, redistribute routes, trigger dial-on-demand (DDR) calls, limit debug output, and identify or classify traffic for quality of service (QoS) features.

An access list is a sequential list that consists of at least one **permit** statement and possibly one or more **deny** statements. In the case of IP access lists, these statements can apply to IP addresses, upper-layer IP protocols, or other fields in IP packets.

Access lists are identified and referenced by a name or a number. Access lists act as packet filters, filtering packets based on the criteria defined in each access list.

After you configure an access list, for the access list to take effect, you must either apply the access list to an interface (by using the **ip access-group** command), a vty (by using the **access-class** command), or reference the access list by any command that accepts an access list. Multiple commands can reference the same access list.

In the following configuration, an IP access list named branchoffices is configured on Ten Gigabit Ethernet interface 4/1/0 and applied to incoming packets. Networks other than the ones specified by the source address and mask pair cannot access Ten Gigabit Ethernet interface 4/1/0. The destinations for packets coming from sources on network 172.16.7.0 are unrestricted. The destination for packets coming from sources on network 172.16.2.0 must be 172.31.5.4.

```
ip access-list extended branchoffices
 10 permit 172.16.7.0 0.0.0.3 any
 20 permit 172.16.2.0 0.0.0.255 host 172.31.5.4
!
interface TenGigabitEthernet4/1/0
 ip access-group branchoffices in
```

# Named or Numbered Access Lists

All access lists must be identified by a name or a number. Named access lists are more convenient than numbered access lists because you can specify a meaningful name that is easier to remember and associate with a task. You can reorder statements in or add statements to a named access list.

Named access lists support the following features that are not supported by numbered access lists:

- IP options filtering
- Noncontiguous ports
- TCP flag filtering
- Deleting of entries with the **no permit** or **no deny** command

**Note** Not all commands that accept a numbered access list will accept a named access list. For example, vty uses only numbered access lists.

# Benefits of IP Access Lists

Access control lists (ACLs) perform packet filtering to control the flow of packets through a network. Packet filtering can restrict the access of users and devices to a network, providing a measure of security. Access lists can save network resources by reducing traffic. The benefits of using access lists are as follows:

- Authenticate incoming rsh and rcp requests—Access lists can simplify the identification of local users, remote hosts, and remote users in an authentication database that is configured to control access to a device. The authentication database enables Cisco software to receive incoming remote shell (rsh) and remote copy (rcp) protocol requests.
- Block unwanted traffic or users—Access lists can filter incoming or outgoing packets on an interface, thereby controlling access to a network based on source addresses, destination addresses, or user authentication. You can also use access lists to determine the types of traffic that are forwarded or blocked at device interfaces. For example, you can use access lists to permit e-mail traffic to be routed through a network and to block all Telnet traffic from entering the network.
- Control access to vty—Access lists on an inbound vty (Telnet) can control who can access the lines to a device. Access lists on an outbound vty can control the destinations that the lines from a device can reach.
- Identify or classify traffic for QoS features—Access lists provide congestion avoidance by setting the IP precedence for Weighted Random Early Detection (WRED) and committed access rate (CAR). Access lists also provide congestion management for class-based weighted fair queueing (CBWFQ), priority queueing, and custom queueing.
- Limit debug command output—Access lists can limit debug output based on an IP address or a protocol.
- Provide bandwidth control—Access lists on a slow link can prevent excess traffic on a network.
- Provide NAT control—Access lists can control which addresses are translated by Network Address Translation (NAT).
- Reduce the chance of DoS attacks—Access lists reduce the chance of denial-of-service (DoS) attacks. Specify IP source addresses to control traffic from hosts, networks, or users from accessing your network. Configure the TCP Intercept feature to can prevent servers from being flooded with requests for connection.
- Restrict the content of routing updates—Access lists can control routing updates that are sent, received, or redistributed in networks.
- Trigger dial-on-demand calls—Access lists can enforce dial and disconnect criteria.

# Access List Rules

The following rules apply to access lists:

- Only one access list per interface, per protocol, and per direction is allowed.
- An access list must contain at least one **permit** statement or all packets are denied entry into the network.
- The order in which access list conditions or match criteria are configured is important. While deciding whether to forward or block a packet, Cisco software tests the packet against each criteria statement in the order in which these statements are created. After a match is found, no more criteria statements are checked. The same **permit** or **deny** statements specified in a different order can result in a packet being passed under one circumstance and denied in another circumstance.

- If an access list is referenced by a name, but the access list does not exist, all packets pass. An interface or command with an empty access list applied to it permits all traffic into the network.
- Standard access lists and extended access lists cannot have the same name.
- Inbound access lists process packets before the packets are routed to an outbound interface. Inbound access lists that have filtering criteria that deny packet access to a network saves the overhead of routing lookup. Packets that are permitted access to a network based on the configured filtering criteria are processed for routing. For inbound access lists, when you configure a **permit** statement, packets are processed after they are received, and when you configure a **deny** statement, packets are discarded.
- Outbound access lists process packets before they leave the device. Incoming packets are routed to the outbound interface and then processed by the outbound access list. For outbound access lists, when you configure a **permit** statement, packets are sent to the output buffer, and when you configure a **deny** statement, packets are discarded.
- An access list can control traffic arriving at a device or leaving a device, but not traffic originating at a device.

# Helpful Hints for Creating IP Access Lists

The following tips will help you avoid unintended consequences and help you create more efficient, useful access lists.

- Create the access list before applying it to an interface (or elsewhere), because if you apply a nonexistent access list to an interface and then proceed to configure the access list, the first statement is put into effect, and the implicit **deny** statement that follows could cause you immediate access problems.
- Another reason to configure an access list before applying it is because an interface with an empty access list applied to it permits all traffic.
- All access lists need at least one **permit** statement; otherwise, all packets are denied and no traffic passes.
- Because the software stops testing conditions after it encounters the first match (to either a **permit** or **deny** statement), you will reduce processing time and resources if you put the statements that packets are most likely to match at the beginning of the access list. Place more frequently occurring conditions before less frequent conditions.
- Organize your access list so that more specific references in a network or subnet appear before more general ones.
- Use the statement **permit any any** if you want to allow all other packets not already denied. Using the statement **permit any any** in effect avoids denying all other packets with the implicit deny statement at the end of an access list. Do not make your first access list entry **permit any any** because all traffic will get through; no packets will reach the subsequent testing. In fact, once you specify **permit any any**, all traffic not already denied will get through.
- Although all access lists end with an implicit **deny** statement, we recommend use of an explicit **deny** statement (for example, **deny ip any any**). On most platforms, you can display the count of packets denied by issuing the **show access-list**command, thus finding out more information about who your access list is disallowing. Only packets denied by explicit **deny** statements are counted, which is why the explicit **deny** statement will yield more complete data for you.
- While you are creating an access list or after it is created, you might want to delete an entry.

  - You cannot delete an entry from a numbered access list; trying to do so will delete the entire access list. If you need to delete an entry, you need to delete the entire access list and start over.
  - You can delete an entry from a named access list. Use the **no permit**or **no deny** command to delete the appropriate entry.

- In order to make the purpose of individual statements more scannable and easily understood at a glance, you can write a helpful remark before or after any statement by using the **remark** command.

- If you want to deny access to a particular host or network and find out if someone from that network or host is attempting to gain access, include the **log** keyword with the corresponding **deny** statement so that the packets denied from that source are logged for you.

- This hint applies to the placement of your access list. When trying to save resources, remember that an inbound access list applies the filter conditions before the routing table lookup. An outbound access list applies the filter conditions after the routing table lookup.

## Where to Apply an Access List

You can apply access lists to the inbound or outbound interfaces of a device. Applying an access list to an inbound interface controls the traffic that enters the interface and applying an access list to an outbound interface controls the traffic that exits the interface.

When software receives a packet at the inbound interface, the software checks the packet against the statements that are configured for the access list. If the access list permits packets, the software processes the packet. Applying access lists to filter incoming packets can save device resources because filtered packets are discarded before entering the device.

Access lists on outbound interfaces filter packets that are transmitted (sent) out of the interface. You can use the TCP Access Control List (ACL) Splitting feature of the Rate-Based Satellite Control Protocol (RBSCP) on the outbound interface to control the type of packets that are subject to TCP acknowledgment (ACK) splitting on an outbound interface.

You can reference an access list by using a **debug** command to limit the amount of debug logs. For example, based on the filtering or matching criteria of the access list, debug logs can be limited to source or destination addresses or protocols.

You can use access lists to control routing updates, dial-on-demand (DDR), and quality of service (QoS) features.

# How to Configure IP Named Access Control Lists

## Creating an IP Named Access List

You can create an IP named access list to filter source addresses and destination addresses or a combination of addresses and other IP fields. Named access lists allow you to identify your access lists with an intuitive name.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. **ip access-list extended** *name*
4. **remark** *remark*
5. **deny** *protocol* [*source source-wildcard*] {**any** | **host** {*address* | *name*} {*destination* [*destination-wildcard*] {**any** | **host** {*address* | *name*} [**log**]
6. **remark** *remark*
7. **permit** *protocol* [*source source-wildcard*] {**any** | **host** {*address* | *name*} {*destination* [*destination-wildcard*] {**any** | **host** {*address* | *name*} [**log**]
8. Repeat Steps 4 through 7 to specify more statements for your access list.
9. **end**
10. **show ip access-lists**

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip access-list extended** *name*<br>**Example:**<br>`Device(config)# ip access-list extended acl1` | Defines an extended IP access list using a name and enters extended named access list configuration mode. |
| **Step 4** | **remark** *remark*<br>**Example:**<br>`Device(config-ext-nacl)# remark protect server by denying sales access to the acl1 network` | (Optional) Adds a description for an access list statement.<br>• A remark can precede or follow an IP access list entry.<br>• In this example, the **remark** command reminds the network administrator that the **deny** command configured in Step 5 denies the Sales network access to the interface. |
| **Step 5** | **deny** *protocol* [*source source-wildcard*] {**any** | **host** {*address* | *name*} {*destination* [*destination-wildcard*] {**any** | **host** {*address* | *name*} [**log**]<br>**Example:**<br>`Device(config-ext-nacl)# deny ip 192.0.2.0 0.0.255.255 host 192.0.2.10 log` | (Optional) Denies all packets that match all conditions specified by the remark. |
| **Step 6** | **remark** *remark*<br>**Example:**<br>`Device(config-ext-nacl)# remark allow TCP from any source to any destination` | (Optional) Adds a description for an access list statement.<br>• A remark can precede or follow an IP access list entry. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 7** | **permit** *protocol* [*source source-wildcard*] {**any** \| **host** {*address* \| *name*} {*destination* [*destination-wildcard*] {**any** \| **host** {*address* \| *name*} [**log**]<br>**Example:**<br>`Device(config-ext-nacl)# permit tcp any any` | Permits all packets that match all conditions specified by the statement. |
| **Step 8** | Repeat Steps 4 through 7 to specify more statements for your access list. | **Note** All source addresses that are not specifically permitted by a statement are denied by an implicit deny statement at the end of the access list. |
| **Step 9** | **end**<br>**Example:**<br>`Device(config-ext-nacl)# end` | Exits extended named access list configuration mode and returns to privileged EXEC mode. |
| **Step 10** | **show ip access-lists**<br>**Example:**<br>`Device# show ip access-lists` | Displays the contents of all current IP access lists. |

**Example:**

The following is sample output from the **show ip access-lists** command:

```
Device# show ip access-lists acl1

Extended IP access list acl1
   permit tcp any 192.0.2.0 255.255.255.255 eq telnet
   deny tcp any any
   deny udp any 192.0.2.0 255.255.255.255 lt 1024
   deny ip any any log
```

# Applying an Access List to an Interface

### Procedure

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

| | Command or Action | Purpose |
|---|---|---|
| **Step 3** | **interface** *type number*<br>**Example:**<br>`Device(config)# interface TenGigabitEthernet4/1/0` | Specifies an interface and enters interface configuration mode. |
| **Step 4** | **ip access-group** {*access-list-number* \| *access-list-name*} {**in** \| **out**}<br>**Example:**<br>`Device(config-if)# ip access-group acl1 in` | Applies the specified access list to the inbound interface.<br>• To filter source addresses, apply the access list to the inbound interface. |
| **Step 5** | **end**<br>**Example:**<br>`Device(config-if)# end` | Exits interface configuration mode and returns to privileged EXEC mode. |

# Additional References for IP Named Access Control Lists

### Related Documents

| Related Topic | Document Title |
|---|---|
| Security commands | • Cisco IOS Security Command Reference: Commands A to C<br>• Cisco IOS Security Command Reference: Commands D to L<br>• Cisco IOS Security Command Reference: Commands M to R<br>• Cisco IOS Security Command Reference: Commands S to Z |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for IP Named Access Control Lists

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 27: Feature Information for IP Named Access Control Lists*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IP access lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |

CHAPTER 14

# IPv4 ACL Chaining Support

ACL Chaining, also known as Multi-Access Control List, allows you to split access control lists (ACLs). This module describes how with the IPv4 ACL Chaining Support feature, you can explicitly split ACLs into common and user-specific ACLs and bind both ACLs to a target for traffic filtering on a device. In this way, the common ACLs in Ternary Content Addressable Memory (TCAM) are shared by multiple targets, thereby reducing the resource usage.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 28: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Restrictions for IPv4 ACL Chaining Support

- A single access control List (ACL) cannot be used for both common and regular ACLs for the same target in the same direction.
- ACL chaining applies to only security ACLs. It is not supported for feature policies, such as Quality of Service (QoS), Firewall Services Module (FW) and Policy Based Routing (PBR).
- Per-target statistics are not supported for common ACLs.

# Information About IPv4 ACL Chaining Support

## ACL Chaining Overview

The packet filter process supports only a single Access control list (ACL) to be applied per direction and per protocol on an interface. This leads to manageability and scalability issues if there are common ACL entries needed on many interfaces. Duplicate Access control entries (ACEs) are configured for all those interfaces, and any modification to the common ACEs needs to be performed for all ACLs.

A typical ACL on the edge box for an Internet Service Provider (ISP) has two sets of ACEs:

- Common ISP specific ACEs
- Customer/interface specific ACEs

The purpose of these address blocks is to deny access to ISP's protected infrastructure networks and anti-spoofing protection by allowing only customer source address blocks. This results in configuring unique ACL per interface and most of the ACEs being common across all ACLs on a device. ACL provisioning and modification is very cumbersome, hence, any changes to the ACE impacts every target.

## IPv4 ACL Chaining Support

IPv4 ACL Chaining Support allows you to split the Access control list (ACL) into common and customer-specific ACLs and attach both ACLs to a common session. In this way, only one copy of the common ACL is attached to Ternary Content Addressable Memory (TCAM) and shared by all users, thereby making it easier to maintain the common ACEs.

The IPv4 ACL Chaining feature allows two IPV4 ACLs to be active on an interface per direction:

- Common
- Regular
- Common and Regular

**Note** If you configure both common and regular ACLs on an interface, the common ACL is considered over a regular ACL.

# How to Configure IPv4 ACL Chaining Support

ACL chaining is supported by extending the **ip traffic filter** command.

The **ip traffic filter** command is not additive. When you use this command, it replaces earlier instances of the command.

For more information, refer to the *IPv6 ACL Chaining with a Common ACL* section in the Security Configuration Guide: Access Control Lists Configuration Guide.

# Configuring an Interface to Accept Common ACL

Perform this task to configure the interface to accept a common Access control list (ACL) along with an interface-specific ACL:

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface** *type number*<br>**Example:**<br>`Device(config)# interface TenGigabitEthernet4/1/0` | Configures an interface and enters the interface configuration mode. |
| **Step 4** | **ip access-group** {**common** {*common-access-list-name* {*regular-access-list* \| **acl**}} {**in** \| **out**}}<br>**Example:**<br>`Device(config-if)# ipv4 access-group common acl-p`<br>` acl1 in` | Configures the interface to accept a common ACL along with the interface-specific ACL. |
| **Step 5** | **end**<br>**Example:**<br>`Device(config-if)# end` | (Optional) Exits the configuration mode and returns to privileged EXEC mode. |

# Configuration Examples for IPv4 ACL Chaining Support

This section provides configuration examples of Common Access Control List (ACL).

## Example: Configuring an Interface to Accept a Common ACL

This example shows how to replace an Access Control List (ACL) configured on the interface without explicitly deleting the ACL:

```
interface TenGigabitEthernet4/1/0
ipv4 access-group common  C_acl ACL1 in
end
replace interface acl ACL1 by ACL2
interface TenGigabitEthernet4/1/0
ipv4 access-group common  C_acl ACL2 in
```

```
end
```

This example shows how common ACL cannot be replaced on interfaces without deleting it explicitly from the interface:

```
interface TenGigabitEthernet4/1/0
ipv4 access-group common  C_acl1 ACL1 in
end
change the common acl to C_acl2
interface TenGigabitEthernet4/1/0
no ipv4 access-group common  C_acl1  ACL1 in
end
interface TenGigabitEthernet4/1/0
ipv4 access-group common  C_acl2  ACL1 in
end
```

**Note** When reconfiguring a common ACL, you must ensure that no other interface on the line card is attached to the common ACL.

**Note** If both common ACL and interface ACL are attached to an interface and only one of the above is reconfigured on the interface, then the other is removed automatically.

This example shows how the interface ACL is removed:

```
interface TenGigabitEthernet4/1/0
ipv4 access-group common C_acl1 ACL1 in
end
```

# Additional References for IPv4 ACL Chaining Support

### Related Documents

| Related Topic | Document Title |
|---|---|
| IPv6 ACL Chaining Support | Security Configuration Guide: Access Control Lists |
| Security commands | • Cisco IOS Security Command Reference: Commands A to C<br>• Cisco IOS Security Command Reference: Commands D to L<br>• Cisco IOS Security Command Reference: Commands M to R<br>• Cisco IOS Security Command Reference: Commands S to Z |

### 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.<br>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.<br>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password. | http://www.cisco.com/support |

# Feature Information for IPv4 ACL Chaining Support

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 29: Feature Information for IPv4 ACL Chaining Support*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IP access lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on the Cisco cBR Series Converged Broadband Routers. |

CHAPTER 15

# IPv6 ACL Chaining with a Common ACL

ACL Chaining, also known as Multi-Access Control List (ACL), allows you to split ACLs. This document describes how with the IPv6 ACL Chaining Support feature, you can explicitly split ACLs into common and user-specific ACLs and bind both ACLs to a target for traffic filtering on a device. In this way, the common ACLs in Ternary Content Addressable Memory (TCAM) are shared by multiple targets, thereby reducing the resource usage.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 30: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information About IPv6 ACL Chaining with a Common ACL

## ACL Chaining Overview

The packet filter process supports only a single Access control list (ACL) to be applied per direction and per protocol on an interface. This leads to manageability and scalability issues if there are common ACL entries needed on many interfaces. Duplicate Access control entries (ACEs) are configured for all those interfaces, and any modification to the common ACEs needs to be performed for all ACLs.

A typical ACL on the edge box for an Internet Service Provider (ISP) has two sets of ACEs:

- Common ISP specific ACEs

- Customer/interface specific ACEs

The purpose of these address blocks is to deny access to ISP's protected infrastructure networks and anti-spoofing protection by allowing only customer source address blocks. This results in configuring unique ACL per interface and most of the ACEs being common across all ACLs on a device. ACL provisioning and modification is very cumbersome, hence, any changes to the ACE impacts every target.

## IPv6 ACL Chaining with a Common ACL

With IPv6 ACL Chaining, you can configure a traffic filter with the following:

- Common ACL
- Specific ACL
- Common and Specific ACL

Each Access control list (ACL) is matched in a sequence. For example, if you have specified both the ACLs - a common and a specific ACL, the packet is first matched against the common ACL; if a match is not found, it is then matched against the specific ACL.

**Note** Any IPv6 ACL may be configured on a traffic filter as a common or specific ACL. However, the same ACL cannot be specified on the same traffic filter as both common and specific.

# How to Configure IPv6 ACL Chaining with a Common ACL

### Before you begin

IPv6 ACL chaining is configured on an interface using an extension of the existing IPv6 traffic-filter command: **ipv6 traffic-filter** [**common** *common-acl*] [*specific-acl*] [ **in** | **out**]

**Note** You may choose to configure either of the following:

- Only a common ACL. For example: **ipv6 traffic-filter common** *common-acl*
- Only a specific ACL. For example: **ipv6 traffic-filter** *common-acl*
- Both ACLs. For example: **ipv6 traffic-filter common** *common-acl specific-acl*

The ipv6 traffic-filter command is not additive. When you use the command, it replaces earlier instances of the command. For example, the command sequence: **ipv6 traffic-filter** [**common** *common-acl*] [*specific-acl*] **in** **ipv6 traffic-filter** [*specific-acl*] **in** binds a common ACL to the traffic filter, removes the common ACL and then binds a specific ACL.

## Configuring IPv6 ACL to an Interface

Perform this task to configure the interface to accept a common access control list (ACL) along with an interface-specific ACL:

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ipv6 traffic filter** {*common-access-list-name* {**in** | **out**}}
5. **end**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface** *type number*<br>**Example:**<br>`Device(config)# interface TenGigabitEthernet4/1/0` | Specifies the interface type and number, and enters interface configuration mode. |
| **Step 4** | **ipv6 traffic filter** {*common-access-list-name* {**in** \| **out**}}<br>**Example:**<br>`Device(config)# ipv6 traffic-filter outbound out` | Applies the specified IPv6 access list to the interface specified in the previous step. |
| **Step 5** | **end**<br>**Example:**<br>`Device(config-if)# end` | (Optional) Exits the configuration mode and returns to privileged EXEC mode. |

# Configuration Examples for IPv6 ACL Chaining with a Common ACL

You may configure the following combinations in no particular order:

- A common ACL, for example: **ipv6 traffic-filter common** *common-acl* **in**
- A specific ACL, for example: **ipv6 traffic-filter** *specific-acl* **in**
- Both ACLs, for example: **ipv6 traffic-filter common** *common-acl* *specific-acl* **in**

# Example: Configuring an Interface to Accept a Common ACL

This example shows how to replace an access control list (ACL) configured on the interface without explicitly deleting the ACL:

```
interface TenGigabitEthernet4/1/0
ipv6 access-group common  C_acl ACL1 in
end
replace interface acl ACL1 by ACL2
interface TenGigabitEthernet4/1/0
ipv6 access-group common  C_acl ACL2 in
end
```

This example shows how to delete a common ACL from an interface. A common ACL cannot be replaced on interfaces without deleting it explicitly from the interface.

```
interface TenGigabitEthernet4/1/0
ipv6 access-group common  C_acl1 ACL1 in
end
change the common acl to C_acl2
interface TenGigabitEthernet4/1/0
no ipv6 access-group common  C_acl1  ACL1 in
end
interface TenGigabitEthernet4/1/0
ipv6 access-group common  C_acl2  ACL1 in
end
```

**Note** When reconfiguring a common ACL, you must ensure that no other interface on the line card is attached to the common ACL.

**Note** If both common ACL and interface ACL are attached to an interface and only one of the above is reconfigured on the interface, then the other is removed automatically.

This example shows how to remove the interface ACL:

```
interface TenGigabitEthernet4/1/0
ipv6 access-group common C_acl1 ACL1 in
end
```

# Additional References for IPv6 ACL Chaining with a Common ACL

### Related Documents

| Related Topic | Document Title |
|---|---|
| IPv4 ACL Chaining Support | Security Configuration Guide: Access Control Lists, Cisco IOS XE Release 3S |

| Related Topic | Document Title |
|---|---|
| Security commands | • Cisco IOS Security Command Reference: Commands A to C<br>• Cisco IOS Security Command Reference: Commands D to L<br>• Cisco IOS Security Command Reference: Commands M to R<br>• Cisco IOS Security Command Reference: Commands S to Z |

### 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.<br>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.<br>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password. | http://www.cisco.com/support |

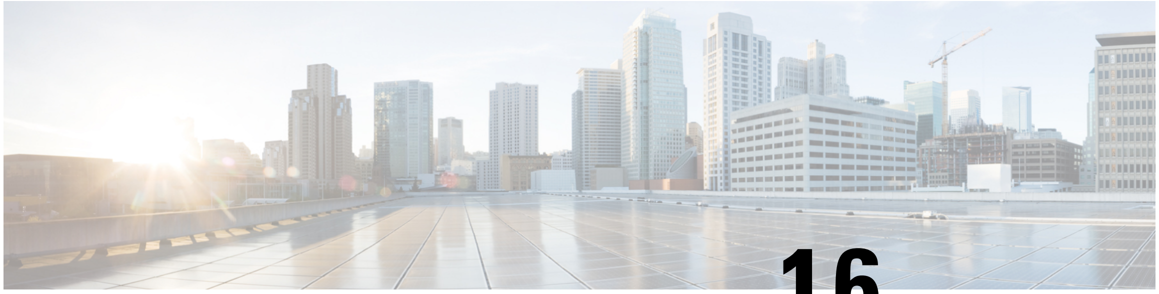
# Feature Information for IPv6 ACL Chaining with a Common ACL

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 31: Feature Information for IPv6 ACL Chaining with a Common ACL*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IPv6 access lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |



CHAPTER 16

# Commented IP Access List Entries

The Commented IP Access List Entries feature allows you to include comments or remarks about **deny** or **permit** conditions in any IP access list. These remarks make access lists easier for network administrators to understand. Each remark is limited to 100 characters in length.

This module provides information about the Commented IP Access List Entries feature.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 32: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information About Commented IP Access List Entries

## Benefits of IP Access Lists

Access control lists (ACLs) perform packet filtering to control the flow of packets through a network. Packet filtering can restrict the access of users and devices to a network, providing a measure of security. Access lists can save network resources by reducing traffic. The benefits of using access lists are as follows:

- Authenticate incoming rsh and rcp requests—Access lists can simplify the identification of local users, remote hosts, and remote users in an authentication database that is configured to control access to a device. The authentication database enables Cisco software to receive incoming remote shell (rsh) and remote copy (rcp) protocol requests.

- Block unwanted traffic or users—Access lists can filter incoming or outgoing packets on an interface, thereby controlling access to a network based on source addresses, destination addresses, or user authentication. You can also use access lists to determine the types of traffic that are forwarded or blocked at device interfaces. For example, you can use access lists to permit e-mail traffic to be routed through a network and to block all Telnet traffic from entering the network.
- Control access to vty—Access lists on an inbound vty (Telnet) can control who can access the lines to a device. Access lists on an outbound vty can control the destinations that the lines from a device can reach.
- Identify or classify traffic for QoS features—Access lists provide congestion avoidance by setting the IP precedence for Weighted Random Early Detection (WRED) and committed access rate (CAR). Access lists also provide congestion management for class-based weighted fair queueing (CBWFQ), priority queueing, and custom queueing.
- Limit debug command output—Access lists can limit debug output based on an IP address or a protocol.
- Provide bandwidth control—Access lists on a slow link can prevent excess traffic on a network.
- Provide NAT control—Access lists can control which addresses are translated by Network Address Translation (NAT).
- Reduce the chance of DoS attacks—Access lists reduce the chance of denial-of-service (DoS) attacks. Specify IP source addresses to control traffic from hosts, networks, or users from accessing your network. Configure the TCP Intercept feature to can prevent servers from being flooded with requests for connection.
- Restrict the content of routing updates—Access lists can control routing updates that are sent, received, or redistributed in networks.
- Trigger dial-on-demand calls—Access lists can enforce dial and disconnect criteria.

# Access List Remarks

You can include comments or remarks about entries in any IP access list. An access list remark is an optional remark before or after an access list entry that describes the entry so that you do not have to interpret the purpose of the entry. Each remark is limited to 100 characters in length.

The remark can go before or after a **permit** or **deny** statement. Be consistent about where you add remarks. Users may be confused if some remarks precede the associated **permit** or **deny** statements and some remarks follow the associated statements.

The following is an example of a remark that describes function of the subsequent **deny** statement:

```
ip access-list extended telnetting
 remark Do not allow host1 subnet to telnet out
 deny tcp host 172.16.2.88 any eq telnet
```

# How to Configure Commented IP Access List Entries

## Writing Remarks in a Named or Numbered Access List

You can use a named or numbered access list configuration. You must apply the access list to an interface or terminal line after the access list is created for the configuration to work.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list** {**standard** | **extended**} {*name* | *number*}
4. **remark** *remark*
5. **deny** *protocol* **host** *host-address* **any eq** *port*
6. **end**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip access-list** {**standard** \| **extended**} {*name* \| *number*}<br>**Example:**<br>`Device(config)# ip access-list extended telnetting` | Identifies the access list by a name or number and enters extended named access list configuration mode. |
| **Step 4** | **remark** *remark*<br>**Example:**<br>`Device(config-ext-nacl)# remark Do not allow host1 subnet to telnet out` | Adds a remark for an entry in a named IP access list.<br>• The remark indicates the purpose of the **permit** or **deny** statement. |
| **Step 5** | **deny** *protocol* **host** *host-address* **any eq** *port*<br>**Example:**<br>`Device(config-ext-nacl)# deny tcp host 172.16.2.88 any eq telnet` | Sets conditions in a named IP access list that denies packets. |
| **Step 6** | **end**<br>**Example:**<br>`Device(config-ext-nacl)# end` | Exits extended named access list configuration mode and enters privileged EXEC mode. |

# Additional References for Commented IP Access List Entries

### Related Documents

| Related Topic | Document Title |
|---|---|
| Security commands | • Cisco IOS Security Command Reference: Commands A to C<br>• Cisco IOS Security Command Reference: Commands D to L<br>• Cisco IOS Security Command Reference: Commands M to R<br>• Cisco IOS Security Command Reference: Commands S to Z |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for Commented IP Access List Entries

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 33: Feature Information for Commented IP Access List Entries*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IP Access Lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |

CHAPTER 17

# Standard IP Access List Logging

The Standard IP Access List Logging feature provides the ability to log messages about packets that are permitted or denied by a standard IP access list. Any packet that matches the access list logs an information message about the packet at the device console.

This module provides information about standard IP access list logging.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 34: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Restrictions for Standard IP Access List Logging

IP access list logging is supported only for routed interfaces or router access control lists (ACLs).

# Information About Standard IP Access List Logging

## Standard IP Access List Logging

The Standard IP Access List Logging feature provides the ability to log messages about packets that are permitted or denied by a standard IP access list. Any packet that matches the access list causes an information log message about the packet to be sent to the device console. The log level of messages that are printed to the device console is controlled by the **logging console** command.

The first packet that the access list inspects triggers the access list to log a message at the device console. Subsequent packets are collected over 5-minute intervals before they are displayed or logged. Log messages include information about the access list number, the source IP address of packets, the number of packets from the same source that were permitted or denied in the previous 5-minute interval, and whether a packet was permitted or denied. You can also monitor the number of packets that are permitted or denied by a particular access list, including the source address of each packet.

# How to Configure Standard IP Access List Logging

## Creating a Standard IP Access List Using Numbers

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list** *access-list-number* {**deny** | **permit**} **host** *address* [**log**]
4. **access-list** *access-list-number* {**deny** | **permit**} **any** [**log**]
5. **interface** *type number*
6. **ip access-group** *access-list-number* {**in** | **out**}
7. **end**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **access-list** *access-list-number* {**deny** \| **permit**} **host** *address* [**log**] | Defines a standard numbered IP access list using a source address and wildcard, and configures the logging of |

| | Command or Action | Purpose |
|---|---|---|
| | **Example:**<br>`Device(config)# access-list 1 permit host 10.1.1.1 log` | informational messages about packets that match the access list entry at the device console. |
| **Step 4** | **access-list** *access-list-number* {**deny** \| **permit**} **any** [**log**]<br>**Example:**<br>`Device(config)# access-list 1 permit any log` | Defines a standard numbered IP access list by using an abbreviation for the source and source mask 0.0.0.0 255.255.255.255. |
| **Step 5** | **interface** *type number*<br>**Example:**<br>`Device(config)# interface TenGigabitEthernet4/1/0` | Configures an interface and enters interface configuration mode. |
| **Step 6** | **ip access-group** *access-list-number* {**in** \| **out**}<br>**Example:**<br>`Device(config-if)# ip access-group 1 in` | Applies the specified numbered access list to the incoming or outgoing interface.<br>• When you filter based on source addresses, you typically apply the access list to an incoming interface. |
| **Step 7** | **end**<br>**Example:**<br>`Device(config-if)# end` | Exits interface configuration mode and enters privileged EXEC mode. |

# Creating a Standard IP Access List Using Names

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list standard** *name*
4. {**deny** | **permit**} {**host** *address* | **any**} **log**
5. **exit**
6. **interface** *type number*
7. **ip access-group** *access-list-name* {**in** | **out**}
8. **end**

### DETAILED STEPS

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

| | Command or Action | Purpose |
|---|---|---|
| **Step 3** | **ip access-list standard** *name*<br>**Example:**<br>`Device(config)# ip access-list standard acl1` | Defines a standard IP access list and enters standard named access list configuration mode. |
| **Step 4** | {**deny** \| **permit**} {**host** *address* \| **any**} **log**<br>**Example:**<br>`Device(config-std-nacl)# permit host 10.1.1.1 log` | Sets conditions in a named IP access list that will deny packets from entering a network or permit packets to enter a network, and configures the logging of informational messages about packets that match the access list entry at the device console. |
| **Step 5** | **exit**<br>**Example:**<br>`Device(config-std-nacl)# exit` | Exits standard named access list configuration mode and enters global configuration mode. |
| **Step 6** | **interface** *type number*<br>**Example:**<br>`Device(config)# interface TenGigabitEthernet4/1/0` | Configures an interface and enters interface configuration mode. |
| **Step 7** | **ip access-group** *access-list-name* {**in** \| **out**}<br>**Example:**<br>`Device(config-if)# ip access-group acl1 in` | Applies the specified access list to the incoming or outgoing interface.<br>• When you filter based on source addresses, you typically apply the access list to an incoming interface. |
| **Step 8** | **end**<br>**Example:**<br>`Device(config-if)# end` | Exits interface configuration mode and enters privileged EXEC mode. |

# Configuration Examples for Standard IP Access List Logging

## Example: Limiting Debug Output

The following sample configuration uses an access list to limit the **debug** command output. Limiting the **debug** output restricts the volume of data to what you are interested in, saving you time and resources.

```
Device(config)# ip access-list acl1
Device(config-std-nacl)# remark Displays only advertisements for LDP peer in acl1
Device(config-std-nacl)# permit host 10.0.0.44

Device# debug mpls ldp advertisements peer-acl acl1

tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.17.0.33
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.16.0.31
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.22.0.33
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.0.1
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.0.3
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.1.33
```

# Additional References for Standard IP Access List Logging

### Related Documents

| Related Topic | Document Title |
|---|---|
| Security commands | • Cisco IOS Security Command Reference: Commands A to C<br>• Cisco IOS Security Command Reference: Commands D to L<br>• Cisco IOS Security Command Reference: Commands M to R<br>• Cisco IOS Security Command Reference: Commands S to Z |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for Standard IP Access List Logging

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 35: Feature Information for Standard IP Access List Logging*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IP Access Lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on the Cisco cBR Series Converged Broadband Routers. |

CHAPTER 18

# IP Access List Entry Sequence Numbering

The IP Access List Entry Sequence Numbering feature allows you to apply sequence numbers to **permit** or **deny** statements as well as reorder, add, or remove such statements from a named IP access list. The IP Access List Entry Sequence Numbering feature makes revising IP access lists much easier. Prior to this feature, you could add access list entries to the end of an access list only; therefore, needing to add statements anywhere except at the end of a named IP access list required reconfiguring the entire access list.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 36: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Restrictions for IP Access List Entry Sequence Numbering

- This feature does not support dynamic, reflexive, or firewall access lists.
- This feature does not support old-style numbered access lists, which existed before named access lists. Keep in mind that you can name an access list with a number, so numbers are allowed when they are entered in the standard or extended named access list (NACL) configuration mode.

# Information About IP Access List Entry Sequence Numbering

## Purpose of IP Access Lists

Access lists perform packet filtering to control which packets move through the network and where. Such control can help limit network traffic and restrict the access of users and devices to the network. Access lists have many uses, and therefore many commands accept a reference to an access list in their command syntax. Access lists can be used to do the following:

- Filter incoming packets on an interface.
- Filter outgoing packets on an interface.
- Restrict the contents of routing updates.
- Limit debug output based on an address or protocol.
- Control virtual terminal line access.
- Identify or classify traffic for advanced features, such as congestion avoidance, congestion management, and priority and custom queuing.
- Trigger dial-on-demand routing (DDR) calls.

## How an IP Access List Works

An access list is a sequential list consisting of a permit statement and a deny statement that apply to IP addresses and possibly upper-layer IP protocols. The access list has a name by which it is referenced. Many software commands accept an access list as part of their syntax.

An access list can be configured and named, but it is not in effect until the access list is referenced by a command that accepts an access list. Multiple commands can reference the same access list. An access list can control traffic arriving at the device or leaving the device, but not traffic originating at the device.

## IP Access List Process and Rules

- The software tests the source or destination address or the protocol of each packet being filtered against the conditions in the access list, one condition (**permit** or **deny** statement) at a time.
- If a packet does not match an access list statement, the packet is then tested against the next statement in the list.
- If a packet and an access list statement match, the rest of the statements in the list are skipped and the packet is permitted or denied as specified in the matched statement. The first entry that the packet matches determines whether the software permits or denies the packet. That is, after the first match, no subsequent entries are considered.
- If the access list denies the address or protocol, the software discards the packet and returns an Internet Control Message Protocol (ICMP) Host Unreachable message.

- If no conditions match, the packet is dropped. This is because each access list ends with an unwritten or implicit **deny** statement. That is, if the packet has not been permitted by the time it was tested against each statement, it is denied.
- Because the software stops testing conditions after the first match, the order of the conditions is critical. The same **permit** or **deny** statements specified in a different order could result in a packet being passed under one circumstance and denied in another circumstance.
- If an access list is referenced by name in a command, but the access list does not exist, all packets pass.
- Only one access list per interface, per protocol, per direction is allowed.
- Inbound access lists process packets arriving at the device. Incoming packets are processed before being routed to an outbound interface. An inbound access list is efficient because it saves the overhead of routing lookups if the packet is to be discarded because it is denied by the filtering tests. If the packet is permitted by the tests, it is then processed for routing. For inbound lists, **permit** means continue to process the packet after receiving it on an inbound interface; **deny** means discard the packet.
- Outbound access lists process packets before they leave the device. Incoming packets are routed to the outbound interface and then processed through the outbound access list. For outbound lists, **permit** means send it to the output buffer; **deny** means discard the packet.

# Helpful Hints for Creating IP Access Lists

The following tips will help you avoid unintended consequences and help you create more efficient, useful access lists.

- Create the access list before applying it to an interface (or elsewhere), because if you apply a nonexistent access list to an interface and then proceed to configure the access list, the first statement is put into effect, and the implicit **deny** statement that follows could cause you immediate access problems.
- Another reason to configure an access list before applying it is because an interface with an empty access list applied to it permits all traffic.
- All access lists need at least one **permit** statement; otherwise, all packets are denied and no traffic passes.
- Because the software stops testing conditions after it encounters the first match (to either a **permit** or **deny** statement), you will reduce processing time and resources if you put the statements that packets are most likely to match at the beginning of the access list. Place more frequently occurring conditions before less frequent conditions.
- Organize your access list so that more specific references in a network or subnet appear before more general ones.
- Use the statement **permit any any** if you want to allow all other packets not already denied. Using the statement **permit any any** in effect avoids denying all other packets with the implicit deny statement at the end of an access list. Do not make your first access list entry **permit any any** because all traffic will get through; no packets will reach the subsequent testing. In fact, once you specify **permit any any**, all traffic not already denied will get through.
- Although all access lists end with an implicit **deny** statement, we recommend use of an explicit **deny** statement (for example, **deny ip any any**). On most platforms, you can display the count of packets denied by issuing the **show access-list**command, thus finding out more information about who your access list is disallowing. Only packets denied by explicit **deny** statements are counted, which is why the explicit **deny** statement will yield more complete data for you.

- While you are creating an access list or after it is created, you might want to delete an entry.
    - You cannot delete an entry from a numbered access list; trying to do so will delete the entire access list. If you need to delete an entry, you need to delete the entire access list and start over.
    - You can delete an entry from a named access list. Use the **no permit**or **no deny** command to delete the appropriate entry.
- In order to make the purpose of individual statements more scannable and easily understood at a glance, you can write a helpful remark before or after any statement by using the **remark** command.
- If you want to deny access to a particular host or network and find out if someone from that network or host is attempting to gain access, include the **log** keyword with the corresponding **deny** statement so that the packets denied from that source are logged for you.
- This hint applies to the placement of your access list. When trying to save resources, remember that an inbound access list applies the filter conditions before the routing table lookup. An outbound access list applies the filter conditions after the routing table lookup.

# Source and Destination Addresses

Source and destination address fields in an IP packet are two typical fields on which to base an access list. Specify source addresses to control the packets being sent from certain networking devices or hosts. Specify destination addresses to control the packets being sent to certain networking devices or hosts.

# Wildcard Mask and Implicit Wildcard Mask

When comparing the address bits in an access list entry to a packet being submitted to the access list, address filtering uses wildcard masking to determine whether to check or ignore the corresponding IP address bits. By carefully setting wildcard masks, an administrator can select one or more IP addresses for permit or deny tests.

Wildcard masking for IP address bits uses the number 1 and the number 0 to specify how the software treats the corresponding IP address bits. A wildcard mask is sometimes referred to as an inverted mask because a 1 and 0 mean the opposite of what they mean in a subnet (network) mask.

- A wildcard mask bit 0 means check the corresponding bit value.
- A wildcard mask bit 1 means ignore that corresponding bit value.

If you do not supply a wildcard mask with a source or destination address in an access list statement, the software assumes a default wildcard mask of 0.0.0.0.

Unlike subnet masks, which require contiguous bits indicating network and subnet to be ones, wildcard masks allow noncontiguous bits in the mask.

# Transport Layer Information

You can filter packets based on transport layer information, such as whether the packet is a TCP, UDP, Internet Control Message Protocol (ICMP) or Internet Group Management Protocol (IGMP) packet.

## Benefits IP Access List Entry Sequence Numbering

The ability to apply sequence numbers to IP access list entries simplifies access list changes. Prior to the IP Access List Entry Sequence Numbering feature, there was no way to specify the position of an entry within an access list. If you wanted to insert an entry (statement) in the middle of an existing list, all of the entries *after* the desired position had to be removed. Then, once you added the new entry, you needed to reenter all of the entries you removed earlier. This method was cumbersome and error prone.

The IP Access List Entry Sequence Numbering feature allows you to add sequence numbers to access list entries and resequence them. When you add a new entry, you can choose the sequence number so that the entry is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced (reordered) to create room to insert the new entry.

## Sequence Numbering Behavior

- For backward compatibility with previous releases, if entries with no sequence numbers are applied, the first entry is assigned a sequence number of 10, and successive entries are incremented by 10. The maximum sequence number is 2147483647. If the generated sequence number exceeds this maximum number, the following message is displayed:

```
Exceeded maximum sequence number.
```

- If you enter an entry without a sequence number, it is assigned a sequence number that is 10 greater than the last sequence number in that access list and is placed at the end of the list.
- If you enter an entry that matches an already existing entry (except for the sequence number), then no changes are made.
- If you enter a sequence number that is already present, the following error message is generated:

```
Duplicate sequence number.
```

- If a new access list is entered from global configuration mode, then sequence numbers for that access list are generated automatically.
- Entries that contain a fully qualified 32-bit host address are hashed instead of linked. And entries that define a sub-net are maintained in a linked list that is sorted by the sequence number for speed of ACL classification. When a packet is matched against a standard ACL, the source address is hashed and matched against the hash table. If no match is found, it then searches the linked list for a possible match.
- Distributed support is provided so that the sequence numbers of entries in the Route Processor (RP) and line card (LC) are always synchronized.
- Sequence numbers are not nvgened. That is, the sequence numbers themselves are not saved. In the event that the system is reloaded, the configured sequence numbers revert to the default sequence starting number and increment from that number. The function is provided for backward compatibility with software releases that do not support sequence numbering.
- The IP Access List Entry Sequence Numbering feature works with named standard and extended IP access lists. Because the name of an access list can be designated as a number, numbers are acceptable.

# How to Use Sequence Numbers in an IP Access List

## Sequencing Access-List Entries and Revising the Access List

This task shows how to assign sequence numbers to entries in a named IP access list and how to add or delete an entry to or from an access list. When completing this task, keep the following points in mind:

- Resequencing the access list entries is optional. The resequencing step in this task is shown as required because that is one purpose of this feature and this task demonstrates that functionality.
- In the following procedure, the **permit** command is shown in Step 5 and the **deny** command is shown in Step 6. However, that order can be reversed. Use the order that suits the need of your configuration.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list resequence** *access-list-name starting-sequence-number increment*
4. **ip access-list** {**standard**| **extended**} *access-list-name*
5. Do one of the following:
    - *sequence-number* **permit** *source source-wildcard*
    - *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
6. Do one of the following:
    - *sequence-number* **deny** *source source-wildcard*
    - *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
7. Do one of the following:
    - *sequence-number* **permit** *source source-wildcard*
    - *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
8. Do one of the following:
    - *sequence-number* **deny** *source source-wildcard*
    - *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
9. Repeat Step 5 and/or Step 6 to add sequence number statements, as applicable.
10. **end**
11. **show ip access-lists** *access-list-name*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode. Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip access-list resequence** *access-list-name starting-sequence-number increment*<br>**Example:**<br>`Device(config)# ip access-list resequence kmd1 100 15` | Resequences the specified IP access list using the starting sequence number and the increment of sequence numbers. |
| **Step 4** | **ip access-list** {**standard**\| **extended**} *access-list-name*<br>**Example:**<br>`Device(config)# ip access-list standard kmd1` | Specifies the IP access list by name and enters named access list configuration mode.<br>• If you specify **standard**, make sure you subsequently specify **permit** and/or **deny** statements using the standard access list syntax.<br>• If you specify **extended**, make sure you subsequently specify **permit** and/or **deny** statements using the extended access list syntax. |
| **Step 5** | Do one of the following:<br>• *sequence-number* **permit** *source source-wildcard*<br>• *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Device(config-std-nacl)# 105 permit 10.5.5.5 0.0.0 255` | Specifies a permit statement in named IP access list mode.<br>• This access list happens to use a **permit** statement first, but a **deny** statement could appear first, depending on the order of statements you need.<br>• As the prompt indicates, this access list was a standard access list. If you had specified **extended** in Step 4, the prompt for this step would be Device(config-ext-nacl) and you would use the extended **permit** command syntax. |
| **Step 6** | Do one of the following:<br>• *sequence-number* **deny** *source source-wildcard*<br>• *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:** | (Optional) Specifies a deny statement in named IP access list mode.<br>• This access list uses a **permit** statement first, but a **deny** statement could appear first, depending on the order of statements you need.<br>• As the prompt indicates, this access list was a standard access list. If you had specified **extended** in Step 4, |

| | Command or Action | Purpose |
|---|---|---|
| | `Device(config-std-nacl)# 105 deny 10.6.6.7 0.0.0 255` | the prompt for this step would be Device(config-ext-nacl) and you would use the extended **deny** command syntax. |
| **Step 7** | Do one of the following:<br>• *sequence-number* **permit** *source source-wildcard*<br>• *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Device(config-ext-nacl)# 150 permit tcp any any log` | Specifies a permit statement in named IP access list mode.<br>• This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need.<br>• See the permit (IP) command for additional command syntax to permit upper layer protocols (ICMP, IGMP, TCP, and UDP).<br>• Use the **no** *sequence-number* command to delete an entry. |
| **Step 8** | Do one of the following:<br>• *sequence-number* **deny** *source source-wildcard*<br>• *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Device(config-ext-nacl)# 150 deny tcp any any log` | (Optional) Specifies a deny statement in named IP access list mode.<br>• This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need.<br>• See the deny (IP) command for additional command syntax to permit upper layer protocols (ICMP, IGMP, TCP, and UDP).<br>• Use the **no** *sequence-number* command to delete an entry. |
| **Step 9** | Repeat Step 5 and/or Step 6 to add sequence number statements, as applicable. | Allows you to revise the access list. |
| **Step 10** | **end**<br>**Example:**<br>`Device(config-std-nacl)# end` | (Optional) Exits the configuration mode and returns to privileged EXEC mode. |
| **Step 11** | **show ip access-lists** *access-list-name*<br>**Example:**<br>`Device# show ip access-lists kmd1` | (Optional) Displays the contents of the IP access list. |

### Examples

Review the output of the **show ip access-lists** command to see that the access list includes the new entries:

```
Device# show ip access-lists kmd1
```

```
Standard IP access list kmd1
100 permit 10.4.4.0, wildcard bits 0.0.0.255
105 permit 10.5.5.0, wildcard bits 0.0.0.255
115 permit 10.0.0.0, wildcard bits 0.0.0.255
130 permit 10.5.5.0, wildcard bits 0.0.0.255
145 permit 10.0.0.0, wildcard bits 0.0.0.255
```

# Configuration Examples for IP Access List Entry Sequence Numbering

## Example: Resequencing Entries in an Access List

The following example shows access list resequencing. The starting value is 1, and increment value is 2. The subsequent entries are ordered based on the increment values specified, and the range is from 1 to 2147483647.

When an entry with no sequence number is entered, by default the entry has a sequence number of 10 more than the last entry in the access list.

```
Device# show access-list 150

Extended IP access list 150
    10 permit ip host 10.3.3.3 host 172.16.5.34
    20 permit icmp any any
    30 permit tcp any host 10.3.3.3
    40 permit ip host 10.4.4.4 any
    50 Dynamic test permit ip any any
    60 permit ip host 172.16.2.2 host 10.3.3.12
    70 permit ip host 10.3.3.3 any log
    80 permit tcp host 10.3.3.3 host 10.1.2.2
    90 permit ip host 10.3.3.3 any
    100 permit ip any any

Device(config)# ip access-list extended 150
Device(config)# ip access-list resequence 150 1 2
Device(config)# exit

Device# show access-list 150

Extended IP access list 150
    1 permit ip host 10.3.3.3 host 172.16.5.34
    3 permit icmp any any
    10 permit tcp any any eq 22 log
    5 permit tcp any host 10.3.3.3
    7 permit ip host 10.4.4.4 any
    9 Dynamic test permit ip any any
    11 permit ip host 172.16.2.2 host 10.3.3.12
    13 permit ip host 10.3.3.3 any log
    15 permit tcp host 10.3.3.3 host 10.1.2.2
    17 permit ip host 10.3.3.3 any
    19 permit ip any any
```

# Example: Adding Entries with Sequence Numbers

In the following example, an new entry is added to a specified access list:

```
Device# show ip access-list

Standard IP access list tryon
2 permit 10.4.4.2, wildcard bits 0.0.255.255
5 permit 10.0.0.44, wildcard bits 0.0.0.255
10 permit 10.0.0.1, wildcard bits 0.0.0.255
20 permit 10.0.0.2, wildcard bits 0.0.0.255

Device(config)# ip access-list standard tryon
Device(config-std-nacl)# 15 permit 10.5.5.5 0.0.0.255
Device(config-std-nacl)# exit
Device(config)# exit
Device# show ip access-list

Standard IP access list tryon
2 permit 10.4.0.0, wildcard bits 0.0.255.255
5 permit 10.0.0.0, wildcard bits 0.0.0.255
10 permit 10.0.0.0, wildcard bits 0.0.0.255
15 permit 10.5.5.0, wildcard bits 0.0.0.255
20 permit 10.0.0.0, wildcard bits 0.0.0.255
```

# Example: Entry Without Sequence Number

The following example shows how an entry with no specified sequence number is added to the end of an access list. When an entry is added without a sequence number, it is automatically given a sequence number that puts it at the end of the access list. Because the default increment is 10, the entry will have a sequence number 10 higher than the last entry in the existing access list.

```
Device(config)# ip access-list standard 1
Device(config-std-nacl)# permit 10.1.1.1 0.0.0.255
Device(config-std-nacl)# permit 10.2.2.2 0.0.0.255
Device(config-std-nacl)# permit 10.3.3.3 0.0.0.255
Device(config-std-nacl)## exit
Device# show access-list

Standard IP access list 1
10 permit 0.0.0.0, wildcard bits 0.0.0.255
20 permit 0.0.0.0, wildcard bits 0.0.0.255
30 permit 0.0.0.0, wildcard bits 0.0.0.255

Device(config)# ip access-list standard 1
Device(config-std-nacl)# permit 10.4.4.4 0.0.0.255
Device(config-std-nacl)# end
Device(config-std-nacl)## exit
Device# show access-list

Standard IP access list 1
10 permit 0.0.0.0, wildcard bits 0.0.0.255
20 permit 0.0.0.0, wildcard bits 0.0.0.255
30 permit 0.0.0.0, wildcard bits 0.0.0.255
40 permit 0.0.0.0, wildcard bits 0.0.0.255
```

# Additional References

### Related Documents

| Related Topic | Document Title |
|---|---|
| IP access list commands | *Cisco IOS Security Command Reference* |
| Configuring IP access lists | *Creating an IP Access List and Applying It to an Interface* |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for IP Access List Entry Sequence Numbering

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 37: Feature Information for IP Access List Entry Sequence Numbering*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IP Access Lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into the Cisco cBR Series Converged Broadband Routers. |

CHAPTER 19

# ACL IP Options Selective Drop

The ACL IP Options Selective Drop feature allows Cisco routers to filter packets containing IP options or to mitigate the effects of IP options on a router or downstream routers by dropping these packets or ignoring the processing of the IP options.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 38: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Restrictions for ACL IP Options Selective Drop

Resource Reservation Protocol (RSVP) (Multiprotocol Label Switching traffic engineering [MPLS TE]), Internet Group Management Protocol Version 2 (IGMPv2), and other protocols that use IP options packets may not function in drop or ignore modes.

# Information About ACL IP Options Selective Drop

## Using ACL IP Options Selective Drop

The ACL IP Options Selective Drop feature allows a router to filter IP options packets, thereby mitigating the effects of these packets on a router and downstream routers, and perform the following actions:

- Drop all IP options packets that it receives and prevent options from going deeper into the network.
- Ignore IP options packets destined for the router and treat them as if they had no IP options.

For many users, dropping the packets is the best solution. However, in environments in which some IP options may be legitimate, reducing the load that the packets present on the routers is sufficient. Therefore, users may prefer to skip options processing on the router and forward the packet as though it were pure IP.

## Benefits of Using ACL IP Options Selective Drop

- Drop mode filters packets from the network and relieves downstream routers and hosts of the load from options packets.
- Drop mode minimizes loads to the Route Processor (RP) for options that require RP processing on distributed systems. Previously, the packets were always routed to or processed by the RP CPU. Now, the ignore and drop forms prevent the packets from impacting the RP performance.

# How to Configure ACL IP Options Selective Drop

## Configuring ACL IP Options Selective Drop

This section describes how to configure the ACL IP Options Selective Drop feature.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip options** {**drop** | **ignore**}
4. **exit**
5. **show ip traffic**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip options** {**drop** \| **ignore**}<br>**Example:**<br>`Router(config)# ip options drop` | Drops or ignores IP options packets that are sent to the router. |
| **Step 4** | **exit**<br>**Example:**<br>`Router(config)# exit` | Returns to privileged EXEC mode. |
| **Step 5** | **show ip traffic**<br>**Example:**<br>`Router# show ip traffic` | (Optional) Displays statistics about IP traffic. |

# Configuration Examples for ACL IP Options Selective Drop

## Example Configuring ACL IP Options Selective Drop

The following example shows how to configure the router (and downstream routers) to drop all options packets that enter the network:

```
Router(config)# ip options drop
% Warning:RSVP and other protocols that use IP Options packets may not function in drop or
 ignore modes.
end
```

## Example Verifying ACL IP Options Selective Drop

The following sample output is displayed after using the **ip options drop** command:

```
Router# show ip traffic
IP statistics:
  Rcvd:  428 total, 323 local destination
         0 format errors, 0 checksum errors, 0 bad hop count
         0 unknown protocol, 0 not a gateway
         0 security failures, 0 bad options, 0 with options
  Opts:  0 end, 0 nop, 0 basic security, 0 loose source route
         0 timestamp, 0 extended security, 0 record route
         0 stream ID, 0 strict source route, 0 alert, 0 cipso, 0 ump
         0 other, 30 ignored
  Frags: 0 reassembled, 0 timeouts, 0 couldn't reassemble
```

```
       0 fragmented, 0 fragments, 0 couldn't fragment
Bcast: 0 received, 0 sent
Mcast: 323 received, 809 sent
Sent:  809 generated, 591 forwarded
Drop:  0 encapsulation failed, 0 unresolved, 0 no adjacency
       0 no route, 0 unicast RPF, 0 forced drop, 0 unsupported-addr
       0 options denied, 0 source IP address zero
```

# Additional References for IP Access List Entry Sequence Numbering

The following sections provide references related to IP access lists.

### Related Documents

| Related Topic | Document Title |
|---|---|
| Configuring IP access lists | "Creating an IP Access List and Applying It to an Interface" |
| IP access list commands | • Cisco IOS Security Command Reference: Commands A to C<br>• Cisco IOS Security Command Reference: Commands D to L<br>• Cisco IOS Security Command Reference: Commands M to R<br>• Cisco IOS Security Command Reference: Commands S to Z |

### 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.<br>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.<br>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password. | http://www.cisco.com/techsupport |

# Feature Information for ACL IP Options Selective Drop

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 39: Feature Information for ACL IP Options Selective Drop*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IP access lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |

CHAPTER 20

# ACL Syslog Correlation

The Access Control List (ACL) Syslog Correlation feature appends a tag (either a user-defined cookie or a device-generated MD5 hash value) to access control entry (ACE) syslog entries. This tag uniquely identifies the ACE , within the ACL, that generated the syslog entry.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 40: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Prerequisites for ACL Syslog Correlation

Before you configure the ACL Syslog Correlation feature, you must understand the concepts in the "IP Access List Overview" module.

The ACL Syslog Correlation feature appends a user-defined cookie or a device-generated hash value to ACE messages in the syslog. These values are only appended to ACE messages when the log option is enabled for the ACE.

# Information About ACL Syslog Correlation

## ACL Syslog Correlation Tags

The ACL Syslog Correlation feature appends a tag (either a user-defined cookie or a device-generated MD5 hash value) to access control entry (ACE) syslog entries. This tag uniquely identifies an ACE that generated the syslog entry.

Network management software can use the tag to identify which ACE generated a specific syslog event. For example, network administrators can select an ACE rule in the network management application and can then view the corresponding syslog events for that ACE rule.

To append a tag to the syslog message, the ACE that generates the syslog event must have the log option enabled. The system appends only one type of tag (either a user-defined cookie or a device-generated MD5 hash value) to each message.

To specify a user-defined cookie tag, the user must enter the cookie value when configuring the ACE log option. The cookie must be in alpha-numeric form, it cannot be greater than 64 characters, and it cannot start with hex-decimal notation (such as 0x).

To specify a device-generated MD5 hash value tag, the hash-generation mechanism must be enabled on the device and the user must not enter a cookie value while configuring the ACE log option.

## ACE Syslog Messages

When a packet is matched against an access control entry (ACE) in an ACL, the system checks whether the log option is enabled for that event. If the log option is enabled and the ACL Syslog Correlation feature is configured on the device, the system attaches the tag to the syslog message. The tag is displayed at the end of the syslog message, in addition to the standard information.

The following is a sample syslog message showing a user-defined cookie tag:

```
Jun 5 12:55:44.359: %SEC-6-IPACCESSLOGP: list logacl permitted tcp 192.168.16.1(38402) ->
192.168.16.2(23), 1 packet [User_permiited_ACE]
```

The following is a sample syslog message showing a hash value tag:

```
Jun 5 12:55:44.359: %SEC-6-IPACCESSLOGP: list logacl permitted tcp 192.168.16.1(38402) ->
192.168.16.2(23), 1 packet [0x723E6E12]
```

# How to Configure ACL Syslog Correlation

## Enabling Hash Value Generation on a Device

Perform this task to configure the device to generate an MD5 hash value for each log-enabled access control entry (ACE) in the system that is not configured with a user-defined cookie.

When the hash value generation setting is enabled, the system checks all existing ACEs and generates a hash value for each ACE that requires one. When the hash value generation setting is disabled, all previously generated hash values are removed from the system.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list logging hash-generation**
4. **end**
5. Do one of the following:
    - **show ip access-list** *access-list-number*
    - **show ip access-list** *access-list-name*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip access-list logging hash-generation**<br>**Example:**<br>`Device(config)# ip access-list logging hash-generation` | Enables hash value generation on the device.<br>• If an ACE exists that is log enabled, and requires a hash value, the device automatically generates the value and displays the value on the console. |
| **Step 4** | **end**<br>**Example:**<br>`Device(config)# end` | (Optional) Exits global configuration mode and returns to privileged EXEC mode. |
| **Step 5** | Do one of the following:<br>• **show ip access-list** *access-list-number*<br>• **show ip access-list** *access-list-name*<br>**Example:**<br>`Device# show ip access-list 101`<br>**Example:**<br>`Device# show ip access-list acl` | (Optional) Displays the contents of the numbered or named IP access list.<br>• Review the output to confirm that the access list for a log-enabled ACE includes the generated hash value. |

# Disabling Hash Value Generation on a Device

Perform this task to disable hash value generation on the device. When the hash value generation setting is disabled, all previously generated hash values are removed from the system.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no ip access-list logging hash-generation**
4. **end**
5. Do one of the following:
   - **show ip access-list** *access-list-number*
   - **show ip access-list** *access-list-name*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **no ip access-list logging hash-generation**<br>**Example:**<br>`Device(config)# no ip access-list logging hash-generation` | Disables hash value generation on the device.<br>• The system removes any previously created hash values from the system. |
| **Step 4** | **end**<br>**Example:**<br>`Device(config)# end` | (Optional) Exits global configuration mode and returns to privileged EXEC mode. |
| **Step 5** | Do one of the following:<br>• **show ip access-list** *access-list-number*<br>• **show ip access-list** *access-list-name*<br>**Example:**<br>`Device# show ip access-list 101`<br>**Example:** | (Optional) Displays the contents of the IP access list.<br>• Review the output to confirm that the access list for a log-enabled ACE does not have a generated hash value. |

| | Command or Action | Purpose |
|---|---|---|
| | `Device# show ip access-list acl` | |

# Configuring ACL Syslog Correlation Using a User-Defined Cookie

Perform this task to configure the ACL Syslog Correlation feature on a device for a specific access list, using a user-defined cookie as the syslog message tag.

The example in this section shows how to configure the ACL Syslog Correlation feature using a user-defined cookie for a numbered access list. However, you can configure the ACL Syslog Correlation feature using a user-defined cookie for both numbered and named access lists, and for both standard and extended access lists.

**Note** The following restrictions apply when choosing the user-defined cookie value:

- The maximum number of characters is 64.
- The cookie cannot start with hexadecimal notation (such as 0x).
- The cookie cannot be the same as, or a subset of, the following keywords: **reflect**, **fragment**, **time-range**. For example, reflect and ref are not valid values. However, the cookie can start with the keywords. For example, reflectedACE and fragment_33 are valid values
- The cookie must contains only alphanumeric characters.

>

## SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list** *access-list-number* **permit** *protocol source destination* **log** *word*
4. **end**
5. **show ip access-list** *access-list-number*

## DETAILED STEPS

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

| | Command or Action | Purpose |
|---|---|---|
| **Step 3** | **access-list** *access-list-number* **permit** *protocol source destination* **log** *word*<br>**Example:**<br>`Device(config)# access-list 101 permit tcp host 10.1.1.1 host 10.1.1.2 log UserDefinedValue` | Defines an extended IP access list and a user-defined cookie value.<br>• Enter the cookie value as the *word*argument. |
| **Step 4** | **end**<br>**Example:**<br>`Device(config)# end` | (Optional) Exits global configuration mode and returns to privileged EXEC mode. |
| **Step 5** | **show ip access-list** *access-list-number*<br>**Example:**<br>`Device# show ip access-list 101` | (Optional) Displays the contents of the IP access list.<br>• Review the output to confirm that the access list includes the user-defined cookie value. |

### Examples

The following is sample output from the **show ip access-list** command for an access list with a user-defined cookie value.

```
Device# show ip access-list
101
Extended IP access list 101
30 permit tcp host 10.1.1.1 host 10.1.1.2 log (tag = UserDefinedValue)
```

# Configuring ACL Syslog Correlation Using a Hash Value

Perform this task to configure the ACL Syslog Correlation feature on a device for a specific access list, using a device-generated hash value as the syslog message tag.

The steps in this section shows how to configure the ACL Syslog Correlation feature using a device-generated hash value for a numbered access list. However, you can configure the ACL Syslog Correlation feature using a device-generated hash value for both numbered and named access lists, and for both standard and extended access lists.

## SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list logging hash-generation**
4. access-list *access-list-number* **permit** *protocol source destination* **log**
5. **end**
6. **show ip access-list** *access-list-number*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip access-list logging hash-generation**<br>**Example:**<br>`Device(config)# ip access-list logging hash-generation` | Enables hash value generation on the device.<br>• If an ACE exists that is log enabled, and requires a hash value, the device automatically generates the value and displays the value on the console. |
| **Step 4** | access-list *access-list-number* **permit** *protocol source destination* **log**<br>**Example:**<br>`Device(config)# access-list 102 permit tcp host 10.1.1.1 host 10.1.1.2 log` | Defines an extended IP access list.<br>• Enable the log option for the access list, but do not specify a cookie value.<br>• The device automatically generates a hash value for the newly defined access list. |
| **Step 5** | **end**<br>**Example:**<br>`Device(config)# end` | (Optional) Exits global configuration mode and returns to privileged EXEC mode. |
| **Step 6** | **show ip access-list** *access-list-number*<br>**Example:**<br>`Device# show ip access-list 102` | (Optional) Displays the contents of the IP access list.<br>• Review the output to confirm that the access list includes the router-generated hash value. |

### Examples

The following is sample output from the **show ip access-list** command for an access list with a device-generated hash value.

```
Device# show ip access-list
102
Extended IP access list 102
10 permit tcp host 10.1.1.1 host 10.1.1.2 log (hash = 0x7F9CF6B9)
```

# Changing the ACL Syslog Correlation Tag Value

Perform this task to change the value of the user-defined cookie or replace a device-generated hash value with a user-defined cookie.

The steps in this section shows how to change the ACL Syslog Correlation tag value on a numbered access list. However, you can change the ACL Syslog Correlation tag value for both numbered and named access lists, and for both standard and extended access lists.

### SUMMARY STEPS

1. **enable**
2. show access-list
3. **configure terminal**
4. access-list *access-list-number* **permit** *protocol source destination* **log** *word*
5. **end**
6. **show ip access-list** *access-list-number*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | show access-list<br>**Example:**<br>`Device(config)# show access-list` | (Optional) Displays the contents of the access list. |
| **Step 3** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 4** | access-list *access-list-number* **permit** *protocol source destination* **log** *word*<br>**Example:**<br>`Device(config)# access-list 101 permit tcp host 10.1.1.1 host 10.1.1.2 log NewUDV`<br>**Example:**<br>`OR`<br>**Example:**<br>**Example:** | Modifies the cookie or changes the hash value to a cookie.<br>• You must enter the entire access list configuration command, replacing the previous tag value with the new tag value. |

| | Command or Action | Purpose |
|---|---|---|
| | `Device(config)# access-list 101 permit tcp any any log replacehash` | |
| **Step 5** | **end**<br>**Example:**<br>`Device(config)# end` | (Optional) Exits global configuration mode and returns to privileged EXEC mode. |
| **Step 6** | **show ip access-list** *access-list-number*<br>**Example:**<br>`Device# show ip access-list 101` | (Optional) Displays the contents of the IP access list.<br>• Review the output to confirm the changes. |

## Troubleshooting Tips

Use the **debug ip access-list hash-generation** command to display access list debug information. The following is an example of the **debug** command output:

```
Device# debug ip access-list hash-generation
 Syslog hash code generation debugging is on
Device# show debug
IP ACL:
 Syslog hash code generation debugging is on
Device# no debug ip access-list hash-generation

 Syslog hash code generation debugging is off
Device# show debug
Device#
```

# Configuration Examples for ACL Syslog Correlation

## Example: Configuring ACL Syslog Correlation Using a User-Defined Cookie

The following example shows how to configure the ACL Syslog Correlation feature on a device using a user-defined cookie.

```
Device#
Device# debug ip access-list hash-generation
Syslog MD5 hash code generation debugging is on
Device# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Device(config)# access-list 33 permit 10.10.10.6 log cook_33_std
Device(config)# do show ip access 33
Standard IP access list 33
10 permit 10.10.10.6 log (tag = cook_33_std)
Device(config)# end
```

# Example: Configuring ACL Syslog Correlation using a Hash Value

The following examples shows how to configure the ACL Syslog Correlation feature on a device using a device-generated hash value.

```
Device# debug ip access-list hash-generation
Syslog MD5 hash code generation debugging is on
Device# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Device(config)# access-list 33 permit 10.10.10.7 log
Device(config)#
*Nov 7 13:51:23.615: %IPACL-HASHGEN: Hash Input: 33 standard permit 10.10.10.7
Hash Output: 0xCE87F535
Device(config)#
do show ip access 33

Standard IP access list 33
    10 permit 10.10.10.6 log (tag = cook_33_std)
    20 permit 10.10.10.7 log (hash = 0xCE87F535)
```

# Example: Changing the ACL Syslog Correlation Tag Value

The following example shows how to replace an existing access list user-defined cookie with a new cookie value, and how to replace a device-generated hash value with a user-defined cookie value.

```
Device# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Device(config)# do show ip access-list 101
Extended IP access list 101
    10 permit tcp host 10.1.1.1 host 10.1.1.2 log (tag = MyCookie)
    20 permit tcp any any log (hash = 0x75F078B9)
Device(config)# access-list 101 permit tcp host 10.1.1.1 host 10.1.1.2 log NewUDV
Device(config)# do show access-list
Extended IP access list 101
    10 permit tcp host 10.1.1.1 host 10.1.1.2 log (tag = NewUDV)
    20 permit tcp any any log (hash = 0x75F078B9)
Device(config)# access-list 101 permit tcp any any log replacehash
Device(config)# do show access-list
Extended IP access list 101
    10 permit tcp host 10.1.1.1 host 10.1.1.2 log (tag = NewUDV)
    20 permit tcp any any log (tag = replacehash)
```

# Additional References for IPv6 IOS Firewall

### Related Documents

| Related Topic | Document Title |
|---|---|
| Security commands | • *Cisco IOS Security Command Reference: Commands A to C*<br>• *Cisco IOS Security Command Reference: Commands D to L*<br>• *Cisco IOS Security Command Reference: Commands M to R*<br>• *Cisco IOS Security Command Reference: Commands S to Z* |
| IPv6 commands | *Cisco IOS IPv6 Command Reference* |
| IPv6 addressing and connectivity | *IPv6 Configuration Guide* |
| Cisco IOS IPv6 features | *Cisco IOS IPv6 Feature Mapping* |

### Standards and RFCs

| Standard/RFC | Title |
|---|---|
| RFCs for IPv6 | *IPv6 RFCs* |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for ACL Syslog Correlation

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 41: Feature Information for ACL Syslog Correlation*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IP access lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |

CHAPTER 21

# IPv6 Access Control Lists

Access lists determine what traffic is blocked and what traffic is forwarded at device interfaces and allow filtering of traffic based on source and destination addresses, and inbound and outbound traffic to a specific interface. Standard IPv6 ACL functionality was extended to support traffic filtering based on IPv6 option headers and optional, upper-layer protocol type information for finer granularity of control. Standard IPv6 ACL functionality was extended to support traffic filtering based on IPv6 option headers and optional, upper-layer protocol type information for finer granularity of control.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 42: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information About IPv6 Access Control Lists

## Access Control Lists for IPv6 Traffic Filtering

The standard ACL functionality in IPv6 is similar to standard ACLs in IPv4. Access lists determine what traffic is blocked and what traffic is forwarded at device interfaces and allow filtering based on source and destination addresses, inbound and outbound to a specific interface. Each access list has an implicit deny statement at the end. IPv6 ACLs are defined and their deny and permit conditions are set using the **ipv6 access-list**command with the **deny** and **permit** keywords in global configuration mode.

IPv6 extended ACLs augments standard IPv6 ACL functionality to support traffic filtering based on IPv6 option headers and optional, upper-layer protocol type information for finer granularity of control (functionality similar to extended ACLs in IPv4).

### IPv6 Packet Inspection

The following header fields are used for IPv6 inspection: traffic class, flow label, payload length, next header, hop limit, and source or destination IP address. For further information on and descriptions of the IPv6 header fields, see RFC 2474.

### Access Class Filtering in IPv6

Filtering incoming and outgoing connections to and from the device based on an IPv6 ACL is performed using the **ipv6 access-class** command in line configuration mode. The **ipv6 access-class** command is similar to the **access-class** command, except the IPv6 ACLs are defined by a name. If the IPv6 ACL is applied to inbound traffic, the source address in the ACL is matched against the incoming connection source address and the destination address in the ACL is matched against the local device address on the interface. If the IPv6 ACL is applied to outbound traffic, the source address in the ACL is matched against the local device address on the interface and the destination address in the ACL is matched against the outgoing connection source address. We recommend that identical restrictions are set on all the virtual terminal lines because a user can attempt to connect to any of them.

# How to Configure IPv6 Access Control Lists

## Configuring IPv6 Traffic Filtering

### Creating and Configuring an IPv6 ACL for Traffic Filtering

**Note** IPv6 ACLs on the Cisco cBR router do not contain implicit permit rules. The IPv6 neighbor discovery process uses the IPv6 network-layer service; therefore, to enable IPv6 neighbor discovery, you must add IPv6 ACLs to allow IPv6 neighbor discovery packets to be sent and received on an interface. In IPv4, the Address Resolution Protocol (ARP), which is equivalent to the IPv6 neighbor discovery process, uses a separate data-link-layer protocol; therefore, by default IPv4 ACLs implicitly allow ARP packets to be sent and received on an interface.

#### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 access-list** *access-list-name*
4. Do one of the following:
    - **permit protocol** { *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* ]] [ **dest-option-type** [ *doh-number* | *doh-type* ]] [ **dscp** *value* ] [

**flow-label** *value* ] [ **fragments** ] [ **log** ] [ **log-input** ] [ **mobility** ] [ **mobility-type** [ *mh-number* | *mh-type* ]] [ **routing** ] [ **routing-type** *routing-number* ] [ **sequence** *value* ] [ **time-range** *name* ]

- **deny** *protocol* { *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* ]] [ **dest-option-type** [ *doh-number* | *doh-type* ] ] [ **dscp** *value* ] [ **flow-label** *value* ] [ **fragments** ] [ **log** ] [ **log-input** ] [ **mobility** ] [ **mobility-type** [ *mh-number* | *mh-type* ]] [ **routing** ] [ **routing-type** *routing-number* ] [ **sequence** *value* ] [ **time-range** *name* ] [ **undetermined-transport**

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ipv6 access-list** *access-list-name*<br>**Example:**<br>`Device(config)# ipv6 access-list inbound` | Defines an IPv6 ACL, and enters IPv6 access list configuration mode.<br>• The *access-list name* argument specifies the name of the IPv6 ACL. IPv6 ACL names cannot contain a space or quotation mark, or begin with a numeral. |
| **Step 4** | Do one of the following:<br>• **permit protocol** { *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* ]] [ **dest-option-type** [ *doh-number* \| *doh-type* ]] [ **dscp** *value* ] [ **flow-label** *value* ] [ **fragments** ] [ **log** ] [ **log-input** ] [ **mobility** ] [ **mobility-type** [ *mh-number* \| *mh-type* ]] [ **routing** ] [ **routing-type** *routing-number* ] [ **sequence** *value* ] [ **time-range** *name* ]<br>• **deny** *protocol* { *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* ]] [ **dest-option-type** [ *doh-number* \| *doh-type* ] ] [ **dscp** *value* ] [ **flow-label** *value* ] [ **fragments** ] [ **log** ] [ **log-input** ] [ **mobility** ] [ **mobility-type** [ *mh-number* \| *mh-type* ]] [ **routing** ] | Specifies permit or deny conditions for an IPv6 ACL. |

| | Command or Action | Purpose |
|---|---|---|
| | [ **routing-type** *routing-number* ] [ **sequence** *value* ] [ **time-range** *name* ] [ **undetermined-transport**<br>**Example:**<br>`Device(config-ipv6-acl)# permit tcp 2001:DB8:0300:0201::/32 eq telnet any`<br>**Example:**<br>`Device(config-ipv6-acl)# deny tcp host 2001:DB8:1::1 any log-input` | |

## Applying the IPv6 ACL to an Interface

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ipv6 traffic-filter** *access-list-name* {**in**\| **out**}

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **interface** *type number*<br>**Example:**<br>`Device(config)# interface TenGigabitEthernet4/1/0` | Specifies the interface type and number, and enters interface configuration mode. |
| **Step 4** | **ipv6 traffic-filter** *access-list-name* {**in**\| **out**}<br>**Example:**<br>`Device(config-if)# ipv6 traffic-filter outbound out` | Applies the specified IPv6 access list to the interface specified in the previous step. |

# Controlling Access to a vty

## Creating an IPv6 ACL to Provide Access Class Filtering

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 access-list** *access-list-name*
4. Do one of the following:
    - **permit protocol** {*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*]] [**dest-option-type** [*doh-number* | *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*
    - **deny** *protocol* {*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*]] [**dest-option-type** [*doh-number* | *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] [**undetermined-transport**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ipv6 access-list** *access-list-name*<br>**Example:**<br>`Device(config)# ipv6 access-list cisco` | Defines an IPv6 ACL, and enters IPv6 access list configuration mode. |
| **Step 4** | Do one of the following:<br>• **permit protocol** {*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*]] [**dest-option-type** [*doh-number* \| *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** | Specifies permit or deny conditions for an IPv6 ACL. |

| | Command or Action | Purpose |
|---|---|---|
| | [*mh-number* \| *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*<br>- **deny** *protocol* {*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*]] [**dest-option-type** [*doh-number* \| *doh-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* \| *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] [**undetermined-transport**<br>**Example:**<br>`Device(config-ipv6-acl)# permit ipv6 host 2001:DB8:0:4::32 any`<br>**Example:**<br>`Device(config-ipv6-acl)# deny ipv6 host 2001:DB8:0:6::6 any` | |

## Applying an IPv6 ACL to the Virtual Terminal Line

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **line** [**aux**\| **console**\| **tty**\| **vty**] *line-number*[*ending-line-number*]
4. **ipv6 access-class** *ipv6-access-list-name* {**in**\| **out**}

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>- Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **line** [**aux**\| **console**\| **tty**\| **vty**] *line-number*[*ending-line-number*]<br>**Example:** | Identifies a specific line for configuration and enters line configuration mode.<br>- In this example, the **vty** keyword is used to specify the virtual terminal lines for remote console access. |

| | Command or Action | Purpose |
|---|---|---|
| | `Device(config)# line vty 0 4` | |
| **Step 4** | **ipv6 access-class** *ipv6-access-list-name* {**in**\| **out**}<br>**Example:**<br>`Device(config-line)# ipv6 access-class cisco in` | Filters incoming and outgoing connections to and from the device based on an IPv6 ACL. |

# Configuration Examples for IPv6 Access Control Lists

## Example: Verifying IPv6 ACL Configuration

In this example, the **show ipv6 access-list** command is used to verify that IPv6 ACLs are configured correctly:

```
Device> show ipv6 access-list

IPv6 access list inbound
    permit tcp any any eq bgp (8 matches) sequence 10
    permit tcp any any eq telnet  (15 matches) sequence 20
    permit udp any any  sequence 30

IPv6 access list Virtual-Access2.1#427819008151 (per-user)
    permit tcp host 2001:DB8:1::32 eq bgp host 2001:DB8:2::32 eq 11000 sequence 1
    permit tcp host 2001:DB8:1::32 eq telnet host 2001:DB8:2::32 eq 11001 sequence 2
```

## Example: Creating and Applying an IPv6 ACL

The following example shows how to restrict HTTP access to certain hours during the day and log any activity outside of the permitted hours:

```
Device# configure terminal
Device(config)# time-range lunchtime
Device(config-time-range)# periodic weekdays 12:00 to 13:00
Device(config-time-range)# exit
Device(config)# ipv6 access-list INBOUND
Device(config-ipv6-acl)# permit tcp any any eq www time-range lunchtime
Device(config-ipv6-acl)# deny tcp any any eq www log-input
Device(config-ipv6-acl)# permit tcp 2001:DB8::/32 any
Device(config-ipv6-acl)# permit udp 2001:DB8::/32 any
Device(config-ipv6-acl)# end
```

## Example: Controlling Access to a vty

In the following example, incoming connections to the virtual terminal lines 0 to 4 are filtered based on the IPv6 access list named acl1:

```
ipv6 access-list acl1
```

```
 permit ipv6 host 2001:DB8:0:4::2/32 any
!
line vty 0 4
 ipv6 access-class acl1 in
```

# Additional References

### Related Documents

| Related Topic | Document Title |
|---|---|
| IP access list commands | *Cisco IOS Security Command Reference* |
| Configuring IP access lists | *Creating an IP Access List and Applying It to an Interface* |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for IPv6 Access Control Lists

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 43: Feature Information for IPv6 Access Control Lists*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IPv6 Access Lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on the Cisco cBR Series Converged Broadband Routers. |

CHAPTER 22

# IPv6 Template ACL

When user profiles are configured using vendor-specific attribute (VSA) Cisco AV-pairs, similar per-user IPv6 ACLs may be replaced by a single template ACL. That is, one ACL represents many similar ACLs. By using IPv6 template ACLs, you can increase the total number of per-user ACLs while minimizing the memory and Ternary Content Addressable Memory (TCAM) resources needed to support the ACLs.

The IPv6 Template ACL feature can create templates using the following ACL fields:

- IPv6 source and destination addresses
- TCP and UDP, including all associated ports (0 through 65535)
- ICMP neighbor discovery advertisements and solicitations
- IPv6 DSCP with specified DSCP values

ACL names are dynamically generated by this feature; for example:

- 6Temp_#152875854573--Example of a dynamically generated template name for a template ACL parent
- Virtual-Access2.32135#152875854573--Example of a child ACL or an ACL that has not yet been made part of a template.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

# Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 44: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information About IPv6 ACL—Template ACL

## IPv6 Template ACL

When user profiles are configured using vendor-specific attribute (VSA) Cisco AV-pairs, similar per-user IPv6 ACLs may be replaced by a single template ACL. That is, one ACL represents many similar ACLs. By using IPv6 template ACLs, you can increase the total number of per-user ACLs while minimizing the memory and Ternary Content Addressable Memory (TCAM) resources needed to support the ACLs.

The IPv6 Template ACL feature can create templates using the following ACL fields:

- IPv6 source and destination addresses
- TCP and UDP, including all associated ports (0 through 65535)
- ICMP neighbor discovery advertisements and solicitations
- IPv6 DSCP with specified DSCP values

ACL names are dynamically generated by this feature; for example:

- 6Temp_#152875854573--Example of a dynamically generated template name for a template ACL parent
- Virtual-Access2.32135#152875854573--Example of a child ACL or an ACL that has not yet been made part of a template.

# How to Enable IPv6 ACL—Template ACL

## Enabling IPv6 Template Processing

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **access-list template** [*number-of-rules*]
4. **exit**
5. **show access-list template** {**summary** | *aclname* | **exceed** *number* | **tree**}

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Router> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **configure terminal**<br>**Example:**<br>`Router# configure terminal` | Enters global configuration mode. |
| **Step 3** | **access-list template** [*number-of-rules*]<br>**Example:**<br>`Router(config)# access-list template 50` | Enables template ACL processing.<br>- The example in this task specifies that ACLs with 50 or fewer rules will be considered for template ACL status.<br>- The *number-of-rules* argument default is 100. |
| **Step 4** | **exit**<br>**Example:**<br>`Router(config)# exit` | Exits global configuration mode and places the router in privileged EXEC mode. |
| **Step 5** | **show access-list template** {**summary** \| *aclname* \| **exceed** *number* \| **tree**}<br>**Example:**<br>`Router# show access-list template summary` | Displays information about ACL templates. |

# Configuration Examples for IPv6 ACL—Template ACL

## Example: IPv6 Template ACL Processing

In this example, the contents of ACL1 and ACL2 are the same, but the names are different:

```
ipv6 access-list extended ACL1 (PeerIP: 2001:1::1/64)
permit igmp any                  2003:1::1/64
permit icmp 2002:5::B/64         any
permit udp  any                  host 2004:1::5
permit udp  any                  host 2002:2BC::a
permit icmp host 2001:BC::7      host 2003:3::7
ipv6 access-list extended ACL2 (PeerIP: 2007:2::7/64)
permit igmp any                  2003:1::1/64
permit icmp 2002:5::B/64         any
permit udp  any                  host 2004:1::5
permit udp  any                  host 2002:2BC::a
permit icmp host 2001:BC::7      host 2003:3::7
```

The template for these ACLs is as follows:

```
ipv6 access-list extended Template_1
permit igmp any                  2003:1::1/64
permit icmp 2002:5::B/64         any
permit udp  any                  host 2004:1::5
permit udp  any              host 2002:2BC::a
permit icmp host 2001:BC::7  host 2003:3::7
```

# Additional References

### Related Documents

| Related Topic | Document Title |
|---|---|
| IPv6 addressing and connectivity | *IPv6 Configuration Guide* |
| IPv6 commands | *Cisco IOS IPv6 Command Reference* |
| Cisco IOS IPv6 features | *Cisco IOS IPv6 Feature Mapping* |

### Standards and RFCs

| Standard/RFC | Title |
|---|---|
| RFCs for IPv6 | *IPv6 RFCs* |

### MIBs

To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL:

http://www.cisco.com/go/mibs

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

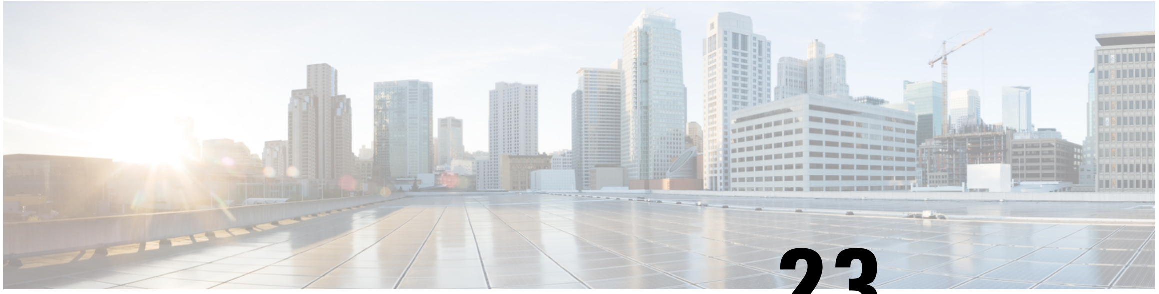
# Feature Information for IPv6 Template ACL

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 45: Feature Information for IPv6 Template ACL*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IPv6 Access Lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |



CHAPTER 23

# IPv6 ACL Extensions for Hop by Hop Filtering

The IPv6 ACL Extensions for Hop by Hop Filtering feature allows you to control IPv6 traffic that might contain hop-by-hop extension headers. You can configure an access control list (ACL) to deny all hop-by-hop traffic or to selectively permit traffic based on protocol.

### Finding Feature Information

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

### Contents

## Hardware Compatibility Matrix for the Cisco cBR Series Routers

**Note** The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

*Table 46: Hardware Compatibility Matrix for the Cisco cBR Series Routers*

| Cisco CMTS Platform | Processor Engine | Interface Cards |
|---|---|---|
| Cisco cBR-8 Converged Broadband Router | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 Supervisor:<br>• PID—CBR-SUP-250G<br>• PID—CBR-CCAP-SUP-160G | **Cisco IOS-XE Release 16.5.1 and Later Releases**<br>Cisco cBR-8 CCAP Line Cards:<br>• PID—CBR-LC-8D30-16U30<br>• PID—CBR-LC-8D31-16U30<br>• PID—CBR-RF-PIC<br>• PID—CBR-RF-PROT-PIC<br>• PID—CBR-CCAP-LC-40G<br>• PID—CBR-CCAP-LC-40G-R<br>• PID—CBR-CCAP-LC-G2-R<br>• PID—CBR-SUP-8X10G-PIC<br>• PID—CBR-2X100G-PIC<br>Digital PICs:<br>• PID—CBR-DPIC-8X10G<br>• PID—CBR-DPIC-2X100G<br>Cisco cBR-8 Downstream PHY Module:<br>• PID—CBR-D31-DS-MOD<br>Cisco cBR-8 Upstream PHY Modules:<br>• PID—CBR-D31-US-MOD |

# Information About IPv6 ACL Extensions for Hop by Hop Filtering

## ACLs and Traffic Forwarding

IPv6 access control lists (ACLs) determine what traffic is blocked and what traffic is forwarded at device interfaces. ACLs allow filtering based on source and destination addresses, inbound and outbound to a specific interface. Use the **ipv6 access-list** command to define an IPv6 ACL, and the **deny** and **permit** commands to configure its conditions.

The IPv6 ACL Extensions for Hop by Hop Filtering feature implements RFC 2460 to support traffic filtering in any upper-layer protocol type.

# How to Configure IPv6 ACL Extensions for Hop by Hop Filtering

## Configuring IPv6 ACL Extensions for Hop by Hop Filtering

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 access-list** *access-list-name*
4. **permit** *protocol* {*source-ipv6-prefix/prefix-length* | **any** | **host** *source-ipv6-address* | **auth**} [*operator* [*port-number*]] {*destination-ipv6-prefix/prefix-length* | **any** | **host** *destination-ipv6-address* | **auth**} [*operator* [*port-number*]] [**dest-option-type** [*header-number* | *header-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**hbh**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**reflect** *name* [**timeout** *value*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*]
5. **deny** *protocol* {*source-ipv6-prefix/prefix-length* | **any** | **host** *source-ipv6-address* | **auth**} [*operator* [*port-number*]] {*destination-ipv6-prefix/prefix-length* | **any** | **host** *destination-ipv6-address* | **auth**} [*operator* [*port-number*]] [**dest-option-type** [*header-number* | *header-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**hbh**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* | *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] [**undetermined-transport**]
6. **end**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ipv6 access-list** *access-list-name*<br>**Example:**<br>`Device(config)# ipv6 access-list hbh-acl` | Defines an IPv6 ACL and enters IPv6 access list configuration mode. |
| **Step 4** | **permit** *protocol* {*source-ipv6-prefix/prefix-length* \| **any** \| **host** *source-ipv6-address* \| **auth**} [*operator* [*port-number*]] {*destination-ipv6-prefix/prefix-length* \| **any** \| **host** *destination-ipv6-address* \| **auth**} [*operator* [*port-number*]] [**dest-option-type** [*header-number* \| *header-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**hbh**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* \| *mh-type*]] [**reflect** *name* [**timeout** *value*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] | Sets permit conditions for the IPv6 ACL. |

| | Command or Action | Purpose |
|---|---|---|
| | **Example:**<br>`Device(config-ipv6-acl)# permit icmp any any dest-option-type` | |
| **Step 5** | **deny** *protocol* {*source-ipv6-prefix/prefix-length* \| **any** \| **host** *source-ipv6-address* \| **auth**} [*operator* [*port-number*]] {*destination-ipv6-prefix/prefix-length* \| **any** \| **host** *destination-ipv6-address* \| **auth**} [*operator* [*port-number*]] [**dest-option-type** [*header-number* \| *header-type*]] [**dscp** *value*] [**flow-label** *value*] [**fragments**] [**hbh**] [**log**] [**log-input**] [**mobility**] [**mobility-type** [*mh-number* \| *mh-type*]] [**routing**] [**routing-type** *routing-number*] [**sequence** *value*] [**time-range** *name*] [**undetermined-transport**]<br>**Example:**<br>`Device(config-ipv6-acl)# deny icmp any any dest-option-type` | Sets deny conditions for the IPv6 ACL. |
| **Step 6** | **end**<br>**Example:**<br>`Device (config-ipv6-acl)# end` | Returns to privileged EXEC configuration mode. |

# Configuration Example for IPv6 ACL Extensions for Hop by Hop Filtering

## Example: IPv6 ACL Extensions for Hop by Hop Filtering

```
Device(config)# ipv6 access-list hbh_acl
Device(config-ipv6-acl)# permit tcp any any hbh
Device(config-ipv6-acl)# permit tcp any any
Device(config-ipv6-acl)# permit udp any any
Device(config-ipv6-acl)# permit udp any any hbh
Device(config-ipv6-acl)# permit hbh any any
Device(config-ipv6-acl)# permit any any
Device(config-ipv6-acl)# hardware statistics
Device(config-ipv6-acl)# exit

! Assign an IP address and add the ACL on the interface.

Device(config)# interface TenGigabitEthernet4/1/0
Device(config-if)# ipv6 address 1001::1/64
Device(config-if)# ipv6 traffic-filter hbh_acl in
Device(config-if)# exit
Device(config)# exit
Device# clear counters
Clear "show interface" counters on all interfaces [confirm]
Device#
```

```
! Verify the configurations.

Device# show running-config interface TenGigabitEthernet4/1/0

Building configuration...

Current configuration : 114 bytes
!
interface TenGigabitEthernet4/1/0
no switchport
ipv6 address 1001::1/64
ipv6 traffic-filter hbh_acl
end
```

# Additional References

### Related Documents

| Related Topic | Document Title |
|---|---|
| IPv6 addressing and connectivity | *IPv6 Configuration Guide* |
| IPv6 commands | *Cisco IOS IPv6 Command Reference* |
| Cisco IOS IPv6 features | *Cisco IOS IPv6 Feature Mapping* |

### Standards and RFCs

| Standard/RFC | Title |
|---|---|
| RFCs for IPv6 | *IPv6 RFCs* |

### MIBs

To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL:

http://www.cisco.com/go/mibs

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for IPv6 ACL Extensions for Hop by Hop Filtering

Use Cisco Feature Navigator to find information about the platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to the www.cisco.com/go/cfn link. An account on the Cisco.com page is not required.

**Note** The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

*Table 47: Feature Information for IPv6 ACL Extensions for Hop by Hop Filtering*

| Feature Name | Releases | Feature Information |
|---|---|---|
| IPv6 access lists | Cisco IOS XE Fuji 16.7.1 | This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers. |