NOKIA

Nokia Service Router Linux

ACL AND POLICY-BASED ROUTING GUIDE

RELEASE 22.11

3HE 18801 AAAA TQZZA
Issue 01

November 2022

Table of contents

1 About this guide

This document describes ACLs and policy-based routing for the Nokia Service Router Linux (SR Linux). Examples of commonly used commands are provided.

This document is intended for network technicians, administrators, operators, service providers, and others who need to understand how the router is configured.

Note:
This manual covers the current release and may also contain some content that will be released in later maintenance loads. See the *SR Linux Release Notes* for information on features supported in each load.

1.1 Precautionary and information messages

The following are information symbols used in the documentation.

DANGER: Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.

WARNING: Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.

Caution: Caution indicates that the described activity or situation may reduce your component or system performance.

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

1.2 Conventions

Nokia SR Linux documentation uses the following command conventions.

- **Bold** type indicates a command that the user must enter.
- Input and output examples are displayed in `Courier` text.
- An open right-angle bracket indicates a progression of menu choices or simple command sequence (often selected from a user interface). Example: **start** > **connect to**.
- A vertical bar (|) indicates a mutually exclusive argument.
- Square brackets ([ ]) indicate optional elements.

- Braces ({ }) indicate a required choice. When braces are contained within square brackets, they indicate a required choice within an optional element.
- *Italic* type indicates a variable.

Generic IP addresses are used in examples. Replace these with the appropriate IP addresses used in the system.

2 What's new

This is the first release of this document. Information in this document was previously found in the *SR Linux Configuration Basics Guide*.

The following updates were made to content that previously existed in the *SR Linux Configuration Basics Guide*.

Topic	Location
Policy-based forwarding based on DSCP and/or IP protocol	Policy-based forwarding
Interface and CPM filter MAC ACLs	MAC ACLs
ACL extensions for 7220 IXR-D5	Supported ACL actions for 7220 IXR-D systems

3 Access control lists

An Access Control List, or ACL, is an ordered set of rules that are evaluated on a packet-by-packet basis to determine whether access should be provided to a specific resource. ACLs can be used to drop unauthorized or suspicious packets from entering or leaving a routing device via specified interfaces.

An ACL is applied to a selected set of traffic contexts. A traffic context could be all the IPv4 or IPv6 packets arriving or leaving on a specific subinterface, or the out-of-band IP traffic arriving on a management interface, or all the in-band IPv4 or IPv6 packets that are locally terminating on the CPM of the router.

Each ACL rule, or entry, has a sequence ID. The ACL evaluates packets starting with the entry with the lowest sequence ID, progressing to the entry with the highest sequence ID. Evaluation stops at the first matching entry (that is, when the packet matches all of the conditions specified by the ACL entry).

IPv4/IPv6 ACLs

For IPv4 and IPv6 traffic, SR Linux supports the following types of ACLs:

- Interface filters

  An interface filter is an IPv4 or IPv6 ACL that is applied to a routed or bridged subinterface to restrict traffic allowed to enter or exit the subinterface. An interface ACL can be applied to input or output traffic for one or more subinterfaces.

  See Interface filters for more information.

- Packet capture filters

  A packet capture filter is an IPv4 or IPv6 ACL used to extract packets to the control plane for inspection by packet capture tools. When an ingress IP packet on any line card transits through the router, and it matches a rule in a packet capture filter policy, it is copied and extracted toward the CPM and delivered to a Linux virtual Ethernet interface so that it can be displayed by a packet capture utility, or encapsulated and forwarded to a remote monitoring system.

  See Packet capture filters for more information.

- CPM filters

  A CPM filter is an IPv4 or IPv6 ACL used for control plane protection. There is one CPM filter for IPv4 traffic and another CPM filter for IPv6 traffic. When an ingress IP packet is matched by a CPM filter rule, and it is a terminating packet (that is, it must be extracted to the CPM), then it is processed according to the matching CPM filter rule.

  See Control plane module (CPM) filters for more information.

- System filters (7220 IXR-D1, D2, and D3 systems only)

  A system filter ACL is an IPv4 or IPv6 ACL that is evaluated early in the ingress pipeline, at a stage before tunnel termination occurs and before interface filters are run. For VXLAN traffic, system filters can match and drop unauthorized VXLAN tunnel packets before they are decapsulated, based on information in the outer header.

  See System filters for more information.

MAC ACLs

For Layer 2 traffic, SR Linux supports the following types of MAC ACLs:

- Interface MAC filters

  An interface MAC filter is a Layer 2 ACL that is applied to a routed or bridged subinterface to restrict the Ethernet frames allowed to enter or exit from that subinterface. An interface MAC filter can match Ethernet frames carrying an IP or non-IP payload and can be applied to the input or output traffic of one or more subinterfaces.

  See Interface filters for more information.

- CPM filter MAC ACLs

  A CPM filter MAC ACL is used for control plane protection. When a CPM filter MAC ACL is created, its rules are automatically evaluated against all non-IP traffic that is extracted to the CPM as a result of a match with an extraction rule (for example, a match of a specific well-known MAC DA value). This is regardless of whether the traffic entered from of network-instance, subinterface, TD3 pipeline, and so on.

  See Control plane module (CPM) filters for more information.

3.1 ACL actions

When a packet matches an ACL entry, an action specified by the ACL entry is applied to the packet. An ACL entry has a primary action and an optional secondary action. The secondary action extends the primary action with additional packet handling operations.

For traffic transiting through the router, ACL entries support the following primary actions:

- accept – Allow the packet through to the next processing function.
- accept and log – Allow the packet through to the next processing function and send information about the accepted packet to the log application.
- drop – Discard the packet without ICMP generation.
- drop and log – Discard the packet without ICMP generation and send information about the dropped packet to the log application.

For traffic transiting through the router, the following secondary action is supported:

- log – Send information about the packet to the log application.

For traffic terminating on the CPM of the router, the preceding primary and secondary actions are supported, as well as the following secondary actions for ACL entries where accept is the primary action:

- distributed-policer – If the packet is extracted to the CPM, feed the packet to a hardware-based policer, which determines if the packet should be queued by the line card toward the CPM or dropped because a bit-per-second rate is exceeded.
- system-cpu-policer – If the packet has been extracted to the CPM, feed the packet to a software-based policer, which determines if the packet should be delivered to the CPM application or dropped because a packet-per-second rate is exceeded.

If a packet matches an ACL entry, no further evaluation is done for the packet. If the packet does not match any ACL entry, the default action is accept. To drop traffic that does not match any ACL entry, you can optionally configure an entry with the highest sequence ID in the ACL to drop all traffic. This causes traffic that does not match any of the lower-sequence ACL entries to be dropped.

The supported actions for each type of ACL differ based on the hardware platform where the ACL is configured. The following tables indicate which actions are supported for each ACL filter type for each hardware platform.

3.1.1 Supported ACL actions for 7250 IXR systems

The following table lists the supported actions for each ACL filter type on 7250 IXR systems.

Table 1: Supported actions for each ACL filter type (7250 IXR)

ACL filter type	Action	Supported?
IPv4/IPv6 Interface filter (input)	accept	Yes
	accept and log	Yes
	drop	Yes
	drop and log	Yes
IPv4/IPv6 Interface filter (output)	accept	Yes
	accept and log	Yes
	drop	Yes
	drop and log	Yes
IPv4/IPv6 CPM filter	accept	Yes
	accept and log	Yes
	drop	Yes
	drop and log	Yes
	accept and distributed-policer	Yes
	accept and distributed-policer and log	No log generated
	accept and system-cpu-policer	Yes
	accept and system-cpu-policer and log	No log generated
Packet capture filter	accept	Yes
	copy	Yes
System filter	accept	No
	drop	No
	drop and log	No

ACL filter type	Action	Supported?
Interface MAC filter (input)	accept	No
	accept and log	No
	drop	No
	drop and log	No
Interface MAC filter (output)	accept	No
	accept and log	No
	drop	No
	drop and log	No
CPM filter MAC ACL	accept	No
	accept and log	No
	drop	No
	drop and log	No
	accept and distributed-policer	No
	accept and distributed-policer and log	No
	accept and system-cpu-policer	No
	accept and system-cpu-policer and log	No

3.1.2 Supported ACL actions for 7220 IXR-D systems

The following table lists the supported actions for each ACL filter type on 7220 IXR-D1, D2, D3, and D5 systems.

Table 2: Supported actions for each ACL filter type (7220 IXR-D1, D2, D3, and D5)

ACL filter type	Action	Supported?
IPv4/IPv6 Interface filter (input)	accept	Yes
	accept and log	No log generated
	drop	Yes
	drop and log	Yes, using separate CPU queue

ACL filter type	Action	Supported?
IPv4/IPv6 Interface filter (output)	accept	Yes
	accept and log	No log generated
	drop	Yes
	drop and log	No log generated
IPv4/IPv6 CPM filter	accept	Yes
	accept and log	No log generated
	drop	Yes
	drop and log	Yes, using shared CPU queue
	accept and distributed-policer	Yes[1]
	accept and distributed-policer and log	No log generated[1, 2]
	accept and system-cpu-policer	Yes
	accept and system-cpu-policer and log	No log generated
Packet capture filter	accept	Yes
	copy	Yes
System filter[3]	accept	Yes
	drop	Yes
	drop and log	Yes
MAC Interface filter (input)	accept	Yes
	accept and log	No log generated
	drop	Yes
	drop and log	Yes, using separate CPU queue
MAC Interface filter (output)	accept	Yes
	accept and log	No log generated
	drop	Yes
	drop and log	No log generated
MAC CPM filter	accept	Yes

ACL filter type	Action	Supported?
	accept and log	No log generated
	drop	Yes
	drop and log	Yes, using shared CPU queue
	accept and distributed-policer	Yes
	accept and distributed-policer and log	No log generated[1, 2]
	accept and system-cpu-policer	Yes
	accept and system-cpu-policer and log	No log generated

Note:

- [1] Distributed-policer not supported on 7220 IXR-D5
- [2] Log not supported on 7220 IXR-D2, D3, and D5
- [3] System filter not supported on 7220 IXR-D5

3.1.3 Supported ACL actions for 7220 IXR-H systems

The following table lists the supported actions for each ACL filter type on 7220 IXR-H2 and H3 systems.

Table 3: Supported actions for each ACL filter type (7220 IXR-H2 and H3)

ACL filter type	Action	Supported?
IPv4/IPv6 Interface filter (input)	accept	Yes
	accept and log	No log generated
	drop	Yes
	drop and log	Yes, but logged packet is also processed by CPM filter
IPv4/IPv6 Interface filter (output)	accept	Yes
	accept and log	No log generated
	drop	Yes
	drop and log	No log generated
IPv4/IPv6 CPM filter	accept	Yes

ACL filter type	Action	Supported?
	accept and log	No log generated
	drop	Yes
	drop and log	Yes
	accept and distributed-policer	No policing
	accept and distributed-policer and log	No policing and no log generated
	accept and system-cpu-policer	Yes
	accept and system-cpu-policer and log	No log generated
Packet capture filter	accept	Yes
	copy	Yes
System filter	accept	No
	drop	No
	drop and log	No
Interface MAC filter (input)	accept	No
	accept and log	No
	drop	No
	drop and log	No
Interface MAC filter (output)	accept	No
	accept and log	No
	drop	No
	drop and log	No
CPM filter MAC ACL	accept	No
	accept and log	No
	drop	No
	drop and log	No
	accept and distributed-policer	No
	accept and distributed-policer and log	No
	accept and system-cpu-policer	No

ACL filter type	Action	Supported?
	accept and system-cpu-policer and log	No

3.2 ACL match conditions

You can specify the following match conditions in IPv4, IPv6, and MAC ACLs.

Match conditions for IPv4 ACLs

IPv4 ACLs analyze IPv4 packets. The following match criteria are supported by IPv4 ACLs:

- IPv4 destination prefix and prefix-length
- IPv4 destination address and address-mask
- TCP/UDP destination port (range)
- ICMP type/code
- IP protocol number
- IPv4 source prefix and prefix-length
- IPv4 source address and address-mask
- TCP/UDP source port (range)
- TCP flags: RST, SYN, and ACK
- Packet fragmentation: whether the packet is a fragment
- Packet fragmentation: whether the packet is a first-fragment (`fragment-offset=0` and `more-fragments=1`)
- IP DSCP

Match conditions for IPv6 ACLs

IPv6 ACLs analyze IPv6 packets. The following match criteria are supported by IPv6 ACLs:

- IPv6 destination prefix and prefix-length
- IPv6 destination address and address-mask
- TCP/UDP destination port (range)
- ICMPv6 type/code
- IPv6 next-header value. This is the value in the very first next-header field, in the fixed header.
- IPv6 source prefix and prefix-length
- IPv6 source address and address-mask
- TCP/UDP source port (range)
- TCP flags: RST, SYN, and ACK
- IP DSCP

Match conditions for MAC ACLs

MAC ACLs analyze Ethernet frames. The following match criteria are supported by MAC ACLs:

- MAC destination address with configurable address mask
- MAC source address match with configurable address mask
- outermost VLAN ID (single VLAN ID value or one contiguous VLAN ID range)
- Ethertype number after the last 802.1Q VLAN tag (if any), specified as a number (0x0600-0xFFFF) or a well-known name

3.3 Interface filters

An interface filter is an IPv4/IPv6 or MAC ACL that restricts traffic allowed to enter or exit a subinterface.

IPv4/IPv6 interface filters

IPv4 and IPv6 ACLs can be applied to a subinterface to restrict IP traffic entering or exiting that subinterface, as follows:

- Input IPv4 ACLs – When an IPv4 filter is used as an input ACL on a subinterface that carries IPv4 traffic, the rules apply to native IPv4 packets (ethertype 0x0800) that enter the subinterface and would normally terminate locally (control/management plane packets) or transit through the router.

  The rules also apply to MPLS-encapsulated IPv4 packets (ethertype 0x8847) that are terminating or transit.
- Input IPv6 ACLs – When an IPv6 filter is used as an input ACL on a subinterface that carries IPv6 traffic, the rules apply to native IPv6 packets (ethertype 0x86DD) that enter the subinterface and would normally terminate locally (control/management plane packets) or transit through the router.

  The rules also apply to MPLS-encapsulated IPv6 packets (ethertype 0x8847) that are terminating or transit.
- Output IPv4 ACLs – When an IPv4 filter is used as an output ACL on a subinterface that carries IPv4 traffic, the rules apply to native IPv4 packets (ethertype 0x0800) that exit the subinterface, including packets that originated locally (control/management plane packets) and packets that transited through the router.

  The rules do not apply to MPLS-encapsulated IPv4 packets (ethertype 0x8847) exiting the subinterface.
- Output IPv6 ACLs – When an IPv6 filter is used as an output ACL on a subinterface that carries IPv6 traffic, the rules apply to native IPv6 packets (ethertype 0x86DD) that exit the subinterface, including packets that originated locally (control/management plane packets) and packets that transited through the router.

  The rules do not apply to MPLS-encapsulated IPv6 packets (ethertype 0x8847) exiting the subinterface.

MAC interface filters

A MAC interface filter is a Layer 2 ACL that can be applied to a routed or bridged subinterface to restrict the Ethernet frames allowed to enter or exit that subinterface. An interface MAC ACL can match Ethernet frames carrying an IP or non-IP payload.

For a specific direction of traffic (input or output), a routed or bridged subinterface of an Ethernet port or LAG can have either a MAC interface ACL or an IPv4/IPv6 interface ACL applied, but not both at the same time. This restriction also applies to subinterfaces of the `mgmt0` management interface.

Table 4: MAC ACL filter rules

Subinterface type	MAC ACL traffic type	Rules
Routed	Input	Rules apply to all Ethernet frames matched by the subinterface encapsulation, even if they contain an IP or MPLS-IP payload.
	Output	Rules apply to all Ethernet frames egressing the subinterface except VXLAN-encapsulated packets and MPLS-encapsulated packets.
Bridged	Input	Rules apply to all Ethernet frames matched by the subinterface encapsulation, even if they contain an IP or MPLS-IP payload, and regardless of whether they are switched or forwarded to the IRB. However, if the IRB also has an input IPv4/IPv6 ACL applied to it, the IRB action takes priority over the bridged subinterface MAC ACL action.
	Output	Rules apply to all Ethernet frames egressing the subinterface except frames routed from the IRB subinterface.

MAC ACLs are not supported on IRB subinterfaces or loopback or `system0` subinterfaces.

3.3.1 Creating an IPv4 ACL

Procedure

The following is an example IPv4 ACL that has one entry.

Example

This example creates an IPv4 ACL named `ip_tcp`. Within the `ip_tcp` ACL, an entry with sequence ID 1000 is configured. The action is specified as `accept`, with logging set to `true`.

The filter matches packets with IP destination address 100.1.3.1/32; for TCP traffic, if the source address 100.1.5.1/32, destination port 6789, and source port 6722 matches the filter, the traffic stream is accepted.

```
--{ * candidate shared default }--[  ]--
# info acl
 ipv4-filter ip_tcp {
        entry 1000 {
            description Match_IP_Address_TCP_Protocol_Ports
            action {
                accept {
                    log true
                }
            }
            match {
                destination-address 100.1.3.1/32
                protocol tcp
                source-address 100.1.5.1/32
                destination-port {
                    value 6789
                }
                source-port {
                    value 6722
                }
            }
        }
    }
```

Example

The following is an example of an entry added to the `ip_tcp` ACL that causes all traffic to be dropped. Because it has the highest sequence ID, traffic that matches any of the lower-sequenced ACL entries would have been accepted before being evaluated by this entry. Only traffic that did not match any of the other ACL entries would be dropped by this entry.

```
--{ * candidate shared default }--[  ]--
# info acl
       ipv4-filter ip_tcp {
           entry 65535 {
               action {
                   drop {
                     log true
                   }
               }
           }
        }
```

Note that the `drop` action with logging set to `true` is not supported on 7220 IXR-D1, D2, D3, and D5 systems when it is attached as an egress filter.

3.3.2 Creating an IPv6 ACL

Procedure

The following is an example IPv6 ACL that has one entry.

Example

This example creates an IPv6 ACL named `ipv6_tcp`. Within the `ipv6_tcp` ACL, an entry with sequence ID 100 is configured. The action is specified as `accept`, with logging set to `true`.

The filter matches packets with IPv6 destination address 2001:10:1:3::1/120; for TCP traffic, if the source address 2001:10:1:5::1/120, destination port 6789, and source port 6722 matches the filter, the traffic stream is accepted.

```
--{ * candidate shared default }--[  ]--
```

```
# info acl
ipv6-filter ipv6_tcp {
        entry 100 {
            description Match_Dest_Address_TCP_Src_Address_DP_SP
            action {
                accept {
                    log true
                }
            }
            match {
                destination-address 2001:10:1:3::1/120
                next-header tcp
                source-address 2001:10:1:5::1/120
                destination-port {
                    value 6789
                }
                source-port {
                    value 6722
                }
            }
        }
    }
```

3.3.3 Creating a MAC ACL

Procedure

To create a MAC ACL specify statements consisting of match criteria (see Match conditions for MAC ACLs) and actions (see Supported ACL actions for 7220 IXR-D systems).

Example: Drop traffic from a source MAC

This example creates a MAC ACL named `mac01`. Within the `mac01` ACL, an entry with sequence ID 100 is configured. The filter matches Ethernet frames with a source MAC address of AA:BB:19:DD:EE:FF. The action is specified as `drop`, with logging set to `true`.

```
--{ * candidate shared default }--[  ]--
# info acl
    acl {
        mac-filter mac01 {
            entry 100 {
                action {
                    drop {
                        log true
                    }
                }
                match {
                    source-mac {
                        address AA:BB:19:DD:EE:FF
                    }
                }
            }
        }
    }
```

Example: Accept traffic using MAC address mask

The following example uses an address mask to specify the source MAC address. The address mask, entered in MAC address format such as ff:ff:ff:ff:00:00, indicates the bit values that must be matched exactly (FF value in the mask) and the bit values that can be 0 or 1 (`00` value in the mask).

```
--{ * candidate shared default }--[  ]--
# info acl
    acl {
        mac-filter mac01 {
            entry 200 {
                action {
                    accept {
                    }
                }
                match {
                    source-mac {
                        address AA:BB:19:DD:EE:FF
                        mask FF:FF:FF:FF:00:00
                    }
                }
            }
        }
    }
```

Example: Drop traffic of specific Ethertype

The following example drops Ethernet frames with ARP Ethertype (`0x0806`).

```
--{ * candidate shared default }--[  ]--
# info acl
    acl {
        mac-filter mac01 {
            entry 300 {
                action {
                    drop {
                        log true
                    }
                }
                match {
                    ethertype 0x0806
                }
            }
        }
    }
```

Example: Drop traffic with specific outermost VLAN ID

You can configure a specific VLAN ID or range of VLAN IDs as match criteria. The match is based on the outermost VLAN ID of the Ethernet frame. The VLAN ID can be specified as an integer from 0 to 4095 or **none**. Specifying 0 as the VLAN ID matches priority-tagged frames; specifying **none** matches untagged frames.

The following example drops Ethernet frames with VLAN ID 4095.

```
--{ * candidate shared default }--[  ]--
# info acl
    acl {
        mac-filter mac01 {
            entry 400 {
                action {
                    drop {
                        log true
                    }
                }
                match {
                    vlan {
                        outermost-vlan-id {
                            value 4095
                        }
                    }
                }
            }
        }
    }
```

The following example configures a range of VLAN IDs, 100 to 999 inclusive, as match criteria.

```
--{ * candidate shared default }--[  ]--
# info acl
    acl {
        mac-filter mac01 {
            entry 500 {
                action {
                    drop {
                        log true
                    }
                }
                match {
                    vlan {
                        outermost-vlan-id {
                           range {
                                start 100
                                end 999
                            }
                        }
                    }
                }
            }
        }
    }
```

Note the following when specifying VLAN IDs as match criteria:

- When used in an ingress ACL applied to a routed or bridged subinterface, the VLAN ID is matched before the subinterface-defining VLAN tag (of a single-tagged subinterface) has been removed.
- When used in an egress ACL applied to a routed subinterface, the VLAN ID is matched after the subinterface-defining VLAN tag (of a single-tagged subinterface) has been added.
- When used in an egress ACL applied to a bridged subinterface the VLAN ID is matched before the subinterface defining VLAN tag (of a single-tagged subinterface) has been added.

3.3.4 Attaching an ACL to a subinterface

Procedure

After an ACL is configured, you can attach it to a subinterface so that traffic entering or exiting the subinterface is subject to the rules defined in the ACL. For an interface ACL to have an effect on the system, it must be explicitly applied to the input and, or output traffic of at least one subinterface.

Example: Attach IPv4/IPv6 ACLs to a subinterface

The following example applies IPv4 and IPv6 ACLs to inbound traffic on a subinterface:

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/16 subinterface 1 acl
interface ethernet-1/16 {
    subinterface 1 {
        acl {
            input {
                ipv4-filter ip_tcp
                ipv6-filter ipv6_tcp
            }
        }
    }
}
```

Example: Attach a MAC ACL to a subinterface

The following example applies a MAC ACL to outbound traffic on a subinterface. To apply a MAC ACL to traffic in the outbound direction on any subinterface, you must set the **acl egress-mac-filtering** command to **true**.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1 subinterface 1 acl
    interface ethernet-1/1 {
        subinterface 1 {
            acl {
                output {
                    mac-filter mac01
                }
            }
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# info acl
    acl {
        egress-mac-filtering true
    }
```

3.3.5 Attaching an ACL to the management interface

Procedure

To modulate traffic for the management interface, navigate to the subinterface of interface `mgmt0`. Under the `acl` context, attach the IPv4 or IPv6 ACL for input/output traffic.

Example:

```
--{ * candidate shared default }--[  ]--
# interface mgmt0
--{ * candidate shared default }--[ interface mgmt0 ]—
# subinterface 0
--{ * candidate shared default }--[ interface mgmt0 subinterface 0 ]--
# acl
--{ * candidate shared default }--[ interface mgmt0 subinterface 0 acl ]—
# input ipv4-filter ip_tcp
```

```
--{ * candidate shared default }--[ interface mgmt0 subinterface 0 acl ]—
# input ipv6-filter ipv6_tcp
```

Example

To verify the configuration for the management interface ACL:

```
--{ * candidate shared default }--[  ]--
# info interface mgmt0
   interface mgmt0     {
     admin-state enable
        subinterface 0 {
            admin-state enable
            ipv4 {
               dhcp-client true
            }
            ipv6 {
               dhcp-client true
            }
            acl {
               input {
                   ipv4-filter ip_tcp
                   ipv6-filter ipv6_tcp
               }
            }
        }
   }
```

3.3.6 Detaching an ACL from an interface

Procedure

To detach an ACL from an interface, enter the subinterface context and delete the ACL from the configuration.

Example:

The following commands detach an ACL from a subinterface:

```
--{ * candidate shared default }--[  ]--
# interface ethernet-1/16
--{ * candidate shared default }--[ interface ethernet-1/16 ]—
# subinterface 1
--{ * candidate shared default }--[ interface ethernet-1/16 subinterface 1 ]—
# delete acl input ipv4-filter
--{ * candidate shared default }--[ interface ethernet-1/16 subinterface 1 ]--
```

Example

Use the **info interface** command to verify that the ACL is no longer part of the subinterface configuration. For example:

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/16
interface ethernet-1/16 {
    description dut1-dut-2
    subinterface 1 {
        ipv4 {
            address 100.1.5.2/24 {
            }
            arp {
                neighbor 100.1.5.1 {
                    link-layer-address 00:00:64:01:05:05
                }
            }
        }
        ipv6 {
            address 2001:10:1:5::2/120 {
            }
            neighbor-discovery {
                neighbor 2001:10:1:5::1 {
                    link-layer-address 00:00:64:01:05:05
                }
            }
        }
        acl {
            input {
                ipv6-filter ipv6_tcp
            }
        }
    }
}
```

3.3.7 Detaching an ACL from the management interface

Procedure

The following example detaches an ACL from the management interface:

Example:

```
--{ * candidate shared default }--[  ]--
# interface mgmt0
--{ * candidate shared default }--[ interface mgmt0 ]—
# subinterface 0
--{ * candidate shared default }--[ interface mgmt0 subinterface 0 ]--
# delete acl input ipv4-filter
```

Example

To verify that the ACL was detached from the management interface:

```
--{ * candidate shared default }--[  ]--
# info interface mgmt0
   interface mgmt0     {
     admin-state enable
        subinterface 0 {
            admin-state enable
            ipv4 {
               dhcp-client true
            }
            ipv6 {
               dhcp-client true
            }
            acl {
               input {
                  ipv6-filter ipv6_tcp
               }
            }
        }
    }
```

3.3.8 Modifying ACLs

Procedure

You can add entries to an ACL, delete entries from an ACL, and delete the entire ACL from the configuration.

Example:

To add an entry to an ACL, enter the context for the ACL, then add the entry. For example, the following commands add an entry to IPv4 ACL `ip_tcp`:

```
--{ * candidate shared default }--[  ]--
# acl ipv4-filter ip_tcp
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# entry 65535
--{ candidate shared default }--[ acl ipv4-filter ip_tcp entry 65535 ]--
# action drop
--{ candidate shared default }--[ acl ipv4-filter ip_tcp entry 65535 action drop ]--
# log true
--{ candidate shared default }--[ acl ipv4-filter ip_tcp entry 65535 action drop ]--
```

Example

To delete an entry in an ACL, use the **delete** command under the context for the ACL and specify the sequence ID of the entry to be deleted. For example, the following commands delete the entry in IPv4 ACL `ip_tcp` with sequence ID 65535:

```
--{ candidate shared default }--[  ]--
# acl ipv4-filter ip_tcp
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# delete entry 65535
```

Example

To delete the entire ACL, use the **delete** command under the `acl` context. For example, the following commands delete the `ip_tcp` ACL:

```
--{ * candidate shared default }--[  ]--
# acl
--{ candidate shared default }--[ acl ]--
# delete ipv4-filter ip_tcp
```

3.3.9 Resequencing ACL entries

Procedure

To aid in managing complex ACLs that have many entries, you can resequence the ACL entries to set a sequence ID number for the first entry and a constant increment for the sequence ID for subsequent entries.

For example, if you have an ACL with three entries, sequence IDs 123, 124, and 301, you can resequence the entries so that the initial entry has sequence ID 100, and the other two entries have sequence ID 110 and 120.

Example:

The following is an example of an ACL with three entries:

```
--{ candidate shared default }--[ acl ]—
# info ipv4-filter ip_tcp
 ipv4-filter ip_tcp {
        entry 123 {
            action {
                drop {
                    log true
                }
            }
            match {
                destination-address 1.1.2.1/24
            }
        }
        entry 124 {
            action {
                accept {
                    log true
                }
            }
            match {
                destination-address 1.1.3.1/24
            }
        }
        entry 301 {
            action {
                drop {
                }
            }
            match {
                destination-address 1.1.4.1/24
            }
        }
```

To resequence the entries in the ACL so that the first entry has sequence ID 100, and the next two entries are incremented by 10, enter the context for the ACL, issue the **tools resequence** command, then specify the initial sequence ID and the increment for the subsequent entries. For example:

```
--{ candidate shared default }--[  ]--
# acl ipv4-filter ip_tcp
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# tools resequence start 100 increment 10
```

After you enter the command, the ACL entries are renumbered. For example:

```
--{ candidate shared default }--[ acl ]—
# info ipv4-filter ip_tcp
 ipv4-filter ip_tcp {
        entry 100 {
            action {
                drop {
                    log true
                }
            }
            match {
                destination-address 1.1.2.1/24
            }
        }
        entry 110 {
            action {
                accept {
                    log true
                }
            }
            match {
                destination-address 1.1.3.1/24
            }
        }
        entry 120 {
            action {
                drop {
                }
            }
            match {
                destination-address 1.1.4.1/24
            }
        }
```

The **resequence** command is only available inside an ACL configuration context, and it only applies to the entries of the ACL associated with that context.

3.4 Packet capture filters

For troubleshooting purposes, the SR Linux supports ACL policies called packet capture filters. When an ingress IP packet on any line card transits through the router, and it matches a rule in a capture-filter policy, it is copied and extracted toward the CPM (using the capture-filter extraction queue) and delivered to a Linux virtual Ethernet interface, so that it can be displayed by **tcpdump** (or similar packet capture utility), or encapsulated and forwarded to a remote monitoring system.

Similarly, when an ingress IP packet on any line card terminates locally, and it matches a rule of a capture-filter policy, it is extracted toward the CPM (using the normal protocol-based extraction queue), and a header field indicates to the CPM to replicate it (after running the CPM-filter rules) toward the Linux virtual Ethernet interface.

There is one capture-filter for IPv4 traffic and another capture-filter for IPv6 traffic. The default IPv4 capture-filter policy copies no IPv4 packets and the default IPv6 capture-filter copies no IPv6 packets.

The entries for each capture-filter are installed on every line card. On the line card, the entries are evaluated after the input subinterface ACLs and before the CPM-filter ACLs. On the CPM, the entries in the capture-filter policy are evaluated after the CPM-filter entries.

When a capture-filter ACL is created, its rules are evaluated against all transit and terminating IPv4 or IPv6 traffic that is arriving on any subinterface of the router, regardless of where that traffic entered in terms of network-instance, subinterface, linecard, pipeline, and so on. Note that capture-filter ACL rules cannot override interface filter or system-filter ACL drop outcomes; packets dropped by interface filter ACLs or a system filter ACL cannot be mirrored to the control plane.

Each capture-filter entry has a set of zero or more match conditions, and one of two possible actions: accept and copy. The match conditions are the same as the other filter types. The accept action passes the matching packet to the next stage of processing, without creating a copy. The copy action creates a copy of the matching packet, extracts it toward the CPM and delivers it to the designated virtual Ethernet interface.

3.5 Control plane module (CPM) filters

For control plane protection, SR Linux supports ACL policies called CPM filters. You can configure one CPM filter that applies to IPv4 traffic, one that applies to IPv6 traffic, and one that applies to non-IP traffic. When ingress traffic is matched by a CPM filter rule, and it is a terminating packet (that is, it must be extracted to the CPM), then it is processed according to the matching CPM filter rule.

The entries for each CPM filter are installed on every line card. They are evaluated after the input subinterface ACLs and after the capture-filter ACLs. CPM filter rules have no effect on locally originating traffic or transit traffic, and they have no interaction with output subinterface ACLs.

When a CPM filter ACL is created, its rules are evaluated against all IPv4 or IPv6 traffic that is locally terminating on the router, regardless of where that traffic entered in terms of network-instance, subinterface, linecard, pipeline, and so on.

On 7250 IXR systems, for traffic terminating on the CPM of the router, the following secondary actions are supported for ACL entries where accept is the primary action:

- distributed-policer – If the packet is extracted to the CPM, feed the packet to a hardware-based policer, which determines if the packet should be queued by the line card toward the CPM or dropped because a bit-per-second rate is exceeded.
- system-cpu-policer – If the packet has been extracted to the CPM, feed the packet to a software-based policer, which determines if the packet should be delivered to the CPM application or dropped because a packet-per-second rate is exceeded.

On 7220 IXR-D1, D2, D3, and D5 systems, CPM filter ACLs support the following actions:

- accept – Allow the packet through to the next processing function.
- accept and distributed-policer – The packet is allowed through to the next processing function and rate limited by a policer instance implemented by the 7220 IXR-D2 and D3.
- accept and system-cpu-policer – The packet is allowed through to the next processing function and rate limited by a policer instance implemented by XDP-CPM.
- drop – Discard the packet without ICMP generation.
- drop and log – Discard the packet without ICMP generation and send information about the dropped packet to the log application.

The system-cpu-policer and distributed-policer actions police terminating traffic to ensure that the rate does not exceed a safe limit.

The system-cpu-policer action applies an aggregate rate limit, regardless of ingress line card, while the distributed-policer action applies a rate limit to the extracted traffic from each core (7250 IXR) or complex (7220 IXR) associated with the ingress port. You can have both types of policer actions in the same CPM filter entry, or only one of them.

CPM filter rules that apply a system-cpu-policer or distributed-policer action do not directly specify the policer parameters; they refer to a generically defined policer. This allows different CPM filter entries, even across multiple ACLs, to use the same policer. Optionally, each policer can be configured as entry-specific, which means a different policer instance is used by each referring filter entry, even if they are part of the same ACL.

3.5.1 Creating CPM filters

Procedure

To create CPM filters for IPv4 traffic, IPv6 traffic, or non-IP traffic, you specify statements consisting of match criteria and actions. CPM-bound traffic that matches the match criteria is processed according to the specified action.

Example: Creating an IPv4 CPM filter

The following example creates a CPM filter for IPv4 traffic. IPv4 traffic extracted to the CPM that matches the rule in the filter is processed according to the action configured in the rule. In this example, the matching CPM-bound IPv4 traffic is accepted and rate-limited according to the limit specified by the `sp1` system-cpu-policer.

```
--{ * candidate shared default }--[  ]--
# info acl cpm-filter
    acl {
        cpm-filter {
            ipv4-filter {
                entry 1000 {
                    action {
                        accept {
                            rate-limit {
                                system-cpu-policer 1000
                            }
                        }
                    }
                    match {
                        protocol tcp
                        destination-ip {
                            address 100.1.3.1
                            mask 255.255.255.255
                        }
                        source-ip {
                            address 100.1.5.1
                            mask 255.255.255.255
                        }
                        destination-port {
                            value 6789
                        }
                        source-port {
                            value 6722
                        }
                    }
                }
            }
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# info acl policers system-cpu-policer sp1
    acl {
        policers {
            system-cpu-policer sp1 {
                peak-packet-rate 1000000
                max-packet-burst 16
            }
        }
    }
```

Example: Creating a MAC CPM filter

The following example creates a MAC CPM filter. Non-IP traffic extracted to the CPM that matches the filter rule is processed according to the configured action. In this example, the matching CPM-bound Ethernet frames are accepted and rate-limited according to the limit specified by the `dp1` distributed-cpu-policer setting.

```
--{ * candidate shared default }--[  ]--
# info acl cpm-filter
    acl {
        cpm-filter {
            mac-filter {
                entry 100 {
                    action {
                        accept {
                            rate-limit {
                                distributed-policer dp1
                            }
                        }
                    }
                    match {
                        source-mac {
                            address AA:BB:19:DD:EE:FF
                            mask FF:FF:FF:FF:00:00
                        }
                    }
                }
            }
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# info acl policers policer dp1
    acl {
        policers {
            policer dp1 {
                peak-rate 500000
                max-burst 100000
            }
        }
    }
```

3.6 System filters

A system filter ACL is an IPv4 or IPv6 ACL that evaluates ingress traffic before all other ACL rules. If an IP packet is dropped by a system filter rule, it is the final disposition of the packet; neither a capture-filter copy/accept action, nor an ingress interface ACL accept action, nor a CPM-filter accept action can override the drop action of a system filter.

At most one system filter can be defined for IPv4 traffic, and at most one system filter can be defined for IPv6 traffic. System filter ACLs are supported on 7220 IXR-D1, D2, and D3 systems only. They can be applied only at ingress, not egress.

When a system-filter ACL is created, its rules are automatically installed everywhere, meaning they are evaluated against all transit and terminating IPv4 or IPv6 traffic arriving on any subinterface of the router, regardless of where that traffic entered in terms of network-instance, subinterface, pipeline, and so on.

A system filter is the only type of filter that can match the outer header of tunneled packets. For VXLAN traffic, this allows you to configure a system filter that matches and drops unauthorized VXLAN tunnel packets before they are decapsulated. The system filter matches the outer header of tunneled packets; they do not filter the payload of VXLAN tunnels.

3.6.1 Creating a system filter

Procedure

The following is an example of a system filter ACL that filters IPv4 traffic. When the system filter is applied, it evaluates all transit and terminating IPv4 arriving on any subinterface of the router. The system filter ACL evaluates the traffic before any other ACL filters. System filter ACLs can be configured on 7220 IXR-D1, D2, and D3 systems only.

Example:

```
--{ * candidate shared default }--[  ]--
# info acl system-filter
 acl
    system-filter {
        ipv4-filter {
            entry 44 {
                action {
                    drop {
                        log true
                    }
                }
                match {
                    source-ip {
                        address 100.1.5.1
                        mask 0.0.0.255
                    }
                }
            }
        }
    }
```

3.7 Configuring logging for ACLs

You can configure the SR Linux to log information about packets that match an ACL entry in the system log.

You can set thresholds for ACL or TCAM resource usage. When utilization of a specified resource reaches the threshold in either the rising or falling direction, it can trigger a log message.

3.7.1 Enabling syslog for the ACL subsystem

Procedure

If you set the **log** parameter to **true** for the accept or drop action in an ACL entry, information about packets that match the ACL entry is recorded in the system log. You can specify settings for the log file for the ACL subsystem, including the location of the log file, maximum log file size, and the number of log files to keep.

Example:

For example, the following configuration specifies that the log file for the ACL subsystem be stored in the file `dut1_file`, located in the `/opt/srlinux/bin/logs/srbase` directory. The log file can be a maximum of 1 Mb. When the log file reaches this size, it is renamed using `dut1_file` as its base name. The five most recent log files are kept.

```
--{ candidate shared default }--[  ]--
# system
--{ candidate shared default }--[ system ]--
# info logging file dut1_file
logging {
        file dut1_file {
            directory /opt/srlinux/bin/logs/srbase/
            rotate 5
            size 1M
            subsystem acl {
            }
        }
}
```

Ensure that write permission is set for the specified directory path.

3.7.1.1 Syslog entry examples

The following are examples of syslog entries for ACLs.

IPv4 Accept:

```
acl||I Type: Ingress IPv4 Filter: testing Sequence Id: 100 Action: Accept Interface: ethernet-
1/16:1 Packet length: 56 IP Source: 100.1.5.1 Destination: 100.1.3.1 Protocol: 6 TCP Source
 port: 6722 Destination Port: 6789 Flags: SYN
```

IPv4 Drop:

```
acl||I Type: Ingress IPv4 Filter: test Sequence Id: 65535 Action: Drop Interface: ethernet-
1/16:1 Packet length: 44 IP Source: 100.3.2.3 Destination: 100.1.3.1 Protocol: 17 UDP Source
 port: 6722 Destination Port: 6789
```

IPv6 Accept:

```
acl||I Type: Ingress IPv6 Filter: tests Sequence Id: 1000 Action: Accept Interface: ethernet-
1/16:1 Packet length: 76 IP Source: 2001:10:1:5::1 Destination: 2001:10:1:3::1 Protocol: 6 TCP
 Source port: 6722 Destination Port: 6789 Flags: SYN
```

3.7.2 Logging ACL resource usage

Procedure

You can set thresholds for ACL resource usage. When utilization of a specified ACL resource, such as input IPv4 filter instances, reaches the threshold in either the rising or falling direction, it can trigger a log message.

Example:

The following example sets thresholds for resource usage by input IPv4 filter instances. If the resource usage percentage falls below the `falling-threshold-log` value, a log message of priority `notice` is generated. If the resource usage percentage falls below the `rising-threshold-log` value, a log message of priority `warning` is generated.

```
--{ * candidate shared default }--[  ]--
# info platform resource-monitoring
    platform {
        resource-monitoring {
            acl {
                resource input-ipv4-filter-instances {
                    rising-threshold-log 90
                    falling-threshold-log 90
                }
            }
        }
    }
```

3.7.3 Logging TCAM resource usage

Procedure

You can set thresholds for Ternary Content Addressable Memory (TCAM) resource usage. When utilization of a specified TCAM resource, such as TCAM used by IPv4 CPM filters, reaches the threshold in either the rising or falling direction, it can trigger a log message.

Example:

The following example sets thresholds for TCAM resource usage by IPv4 CPM filters. If the resource usage percentage falls below the `falling-threshold-log` value, a log message of priority `notice` is generated. If the resource usage percentage falls below the `rising-threshold-log` value, a log message of priority `warning` is generated.

```
--{ * candidate shared default }--[  ]--
# info platform resource-monitoring
    platform {
        resource-monitoring {
            tcam {
                resource cpm-capture-ipv4 {
                    rising-threshold-log 90
                    falling-threshold-log 90
                }
            }
        }
    }
```

3.8 Collecting and displaying ACL statistics

The SR Linux can collect statistics for packets matching an ACL and display statistics for the matched packets. You can display the amount of system resources (TCAM) used by each type of ACL on each line card. ACL statistics can also be displayed using **show** commands.

3.8.1 Collecting ACL statistics

Procedure

You can configure an ACL to collect statistics for packets matching the ACL. Statistics can be collected for packets that match each ACL entry, as well as for matching input/output traffic per subinterface.

Example:

The following example configures the ACL to record the number of matching packets for each entry:

```
--{ candidate shared default }--[ acl ]--
# ipv4-filter ip_tcp
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# statistics-per-entry true
```

Example

By default, if two or more subinterfaces on the same line card reference the same ACL for filtering the same direction of traffic, they use a shared instance of the same ACL in hardware. This means that per-entry statistics (including the number of matched packets and the time stamp of the last matching packet), if enabled, reflect the aggregate of the data gathered for the multiple subinterfaces.

To collect per-entry, per-subinterface statistics, instead of the aggregate of the subinterfaces where the ACL is applied, you can configure an ACL to operate in subinterface-specific mode.

If you change an ACL from subinterface-specific mode to shared mode, or the other way around, during the transition from one mode to the next, traffic continues to be subject to the previous mode until the system resources (TCAM) entries are programmed for the new mode.

The following example configures the ACL to collect statistics for matching packets inbound and outbound on each subinterface:

```
--{ candidate shared default }--[ acl ]--
# ipv4-filter ip_tcp
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# subinterface-specific input-and-output
```

You can configure the following values for the **subinterface-specific** parameter:

- **disabled** (the default) – All subinterfaces on a single line card that reference the ACL as an input ACL use a shared filter instance, and all subinterfaces on a single line card that reference the ACL as an output ACL use a shared filter instance.
- **input-only** – All subinterfaces on a single line card that reference the ACL as an output ACL use a shared filter instance, but each subinterface that references the ACL as an input ACL uses its own separate instance of the filter.
- **output-only** – All subinterfaces on a single line card that reference the ACL as an input ACL use a shared filter instance, but each subinterface that references the ACL as an output ACL uses its own separate instance of the filter.
- **input-and-output** – Each subinterface that references the ACL as either an input ACL or an output ACL uses its own separate instance of the filter.

3.8.2 Displaying ACL statistics

Procedure

Use the **info from state** command to display the matched packet statistics and the time of the last match for the interfaces to which the ACL is attached.

Example:

In the following example, the ACL is attached to two interfaces, and statistics are collected for each subinterface:

```
--{ candidate shared default }--[ acl ipv4-filter ip_tcp ]--
# info from state
    subinterface-specific input-and-output
    statistics-per-entry true
    entry 1000 {
        description Match_IP_Address_TCP_Protocol_Ports
        action {
            accept {
                log true
            }
        }
        match {
            destination-address 100.1.3.1/32
            protocol tcp
            source-address 100.1.5.1/32
            destination-port {
                value 6789
            }
            source-port {
                value 6722
            }
        }
        statistics {
            aggregate {
                in-matched-packets 3000
                in-last-match 2019-07-16T10:53:00.1563Z
                out-matched-packets 0
            }
            per-interface {
                subinterface ethernet-1/16.1 {
                    in-matched-packets 3000
                    in-last-match 2019-07-16T10:53:00.1563Z
                }
            }
        }
    }
    entry 65535 {
        action {
            drop {
                log true
            }
        }
        statistics {
            aggregate {
                in-matched-packets 1000
                in-last-match 2019-07-16T10:53:30.1563Z
            }
            per-interface {
                subinterface ethernet-1/16.1 {
                    in-matched-packets 1000
                    in-last-match 2019-07-16T10:53:30.1563Z
                }
            }
        }
    }
```

If the match criteria changes for an ACL entry, the statistics counter does not reset to zero. To reset the statistics counter for an ACL entry to zero, use the **tools acl clear** command, as described in Clearing ACL statistics.

3.8.3 Displaying ACL resource usage

Procedure

Use the **info from state platform** command to display the amount of system resources (TCAM) used by each type of ACL on each line card.

Example:

The following example shows two different numbers for the remaining (free) TCAM entry resources that are available for input IPv4 ACLs on the line card. The free-static value refers to the available number of resources assuming no additional TCAM banks are dynamically assigned to the ACL type. The free-dynamic value refers to the available number of resources assuming all unallocated TCAM banks are dedicated to the specified ACL type.

```
--{ candidate shared default }--[  ]--
# info from state platform linecard 1 forwarding-complex 0 tcam resource if-input-ipv4
    platform {
        linecard 1 {
            forwarding-complex 0 {
                tcam {
                    resource if-input-ipv4 {
                        free-static 2046
                        free-dynamic 18430
                        reserved 2
                        programmed 2
                    }
                }
            }
        }
    }
```

Example:

The following example shows the amount of system resources allocated to input IPv4 ACLs on the line card and how much is used and free:

```
--{ candidate shared default }--[  ]--
# info from state platform linecard 1 forwarding-complex 0 acl resource input-ipv4-filter-
instances
    platform {
        linecard 1 {
            forwarding-complex 0 {
                acl {
                    resource input-ipv4-filter-instances {
                        used 1
                        free 254
                    }
                }
            }
        }
    }
```

3.8.4 Clearing ACL statistics

Procedure

To reset ACL statistics counters to zero, use the **tools acl clear** command. This command can clear statistics at the IPv4 / IPv6 / CPM filter level, ACL entry level, or for an interface or subinterface to which the ACL is attached.

Example:

The following example clears statistics for an IPv4 filter:

```
--{ candidate shared default }--[ acl ]--
# tools acl ipv4-filter tcp_ip clear
```

Example

After this command is executed, the **info from state** output for the IPv4 filter includes a timestamp indicating when the statistics were cleared. For example:

```
--{ candidate shared default }--[ acl ipv4-filter tcp_ip ]--
# info from state
    subinterface-specific output-only
    statistics-per-entry true
    entry 1000 {
        description Match_IP_Address_TCP_Protocol_Ports
        action {
            accept {
                log true
            }
        }
        match {
            destination-address 100.1.3.1/32
            protocol tcp
            source-address 100.1.5.1/32
            destination-port {
                value 6789
            }
            source-port {
                value 6722
            }
        }
        statistics {
            last-clear 2021-10-03T13:53:51.000Z
        }
    }
```

Example

The following example clears statistics for a specific entry in the IPv4 filter:

```
--{ candidate shared default }--[ acl ]--
# tools acl ipv4-filter tcp_ip entry 1000 statistics clear
```

Example

The following example clears statistics for a specified subinterface for a specified entry in the IPv4 filter:

```
--{ candidate shared default }--[ acl ]--
# tools acl ipv4-filter tcp_ip entry 1000 statistics per-interface subinterface 1 clear
```

3.8.5 Displaying ACL statistics using show commands

Procedure

You can display ACL statistics using relevant **show** commands.

Example:

To display information about all active ACLs, use the **show acl summary** command. For example:

```
--{ candidate shared default }--[  ]--
# show acl summary
--------------------------------------------------------------------------------
CPM Filter ACLs
--------------------------------------------------------------------------------
ipv4-entries: 1
ipv6-entries: 0
mac-entries : 1
--------------------------------------------------------------------------------
Capture Filter ACLs
--------------------------------------------------------------------------------
ipv4-entries: 0
ipv6-entries: 0
--------------------------------------------------------------------------------
IPv4 Filter ACLs
--------------------------------------------------------------------------------
Filter : ip_tcp
Active on : 1 subinterfaces (input) and 0 subinterfaces (output)
Entries : 2
--------------------------------------------------------------------------------
IPv6 Filter ACLs
--------------------------------------------------------------------------------
Filter : ipv6_tcp
Active on : 1 subinterfaces (input) and 0 subinterfaces (output)
Entries : 1
--------------------------------------------------------------------------------
MAC Filter ACLs
--------------------------------------------------------------------------------
Filter   : mac01
Active On: 1 subinterfaces (input) and 0 subinterfaces (output)
Entries  : 5
--------------------------------------------------------------------------------
```

Example

You can display statistics for a specific ACL, including how many times each ACL entry was matched on all subinterfaces to which the ACL was applied. For example:

```
--{ candidate shared default }--[  ]--
# show acl ipv4-filter ip_tcp
====================================================================================
Filter        : ip_tcp
SubIf-Specific: input-and-output
Entry-stats   : yes
Entries       : 2
------------------------------------------------------------------------------------
 Subinterface     Input   Output
ethernet-1/16.1   yes     no
------------------------------------------------------------------------------------
Entry 1000
  Match               : protocol=tcp, 100.1.5.1/32(6722-6722)->100.1.3.1/32(6789-
6789)
  Action              : accept
  Input Match Packets : 3000
  Input Last Match    : 18 seconds ago
  Output Match Packets: 0
  Output Last Match   : never
Entry 65535
  Match               : protocol=<undefined>, any(*)->any(*)
  Action              : drop
  Input Match Packets : 1000
  Input Last Match    : 6 minutes ago
  Output Match Packets: 0
  Output Last Match   : never
```

Example

To display per-interface statistics for packets matching each ACL entry, specify the interface name in addition to the ACL name. For example:

```
--{ candidate shared default }--[  ]--
# show acl ipv4-filter ip_tcp interface ethernet-1/16
====================================================================================
Filter        : ip_tcp
SubIf-Specific: input-and-output
Entry-stats   : yes
Entries       : 2
------------------------------------------------------------------------------------
 Subinterface     Input   Output
ethernet-1/16.1   yes     no
------------------------------------------------------------------------------------
Entry 1000
  Match               : protocol=tcp, 100.1.5.1/32(6722-6722)->100.1.3.1/32(6789-
6789)
  Action              : accept
  Input Match Packets : 3000
  Input Last Match    : 5 minutes ago
  Output Match Packets: 0
  Output Last Match   : never
Entry 65535
  Match               : protocol=<undefined>, any(*)->any(*)
  Action              : drop
  Input Match Packets : 1000
  Input Last Match    : 10 minutes ago
  Output Match Packets: 0
  Output Last Match   : never
```

Example

To display statistics for packets matching a CPM filter, specify the CPM filter type (IPv4 or IPv6). For example:

```
--{ candidate shared default }--[  ]--
```

```
# show acl cpm-filter ipv4-filter
================================================================================
Filter     : CPM IPv4-filter
Entry-stats: no
Entries    : 1
--------------------------------------------------------------------------------
Entry 1001
  Match          : protocol=<undefined>, any(*)->1.1.2.1/24(*)
  Action         : none
  Matched Packets: 0
--------------------------------------------------------------------------------
```

4 Policy-based forwarding

Policy-based forwarding (PBF) supports traffic forwarding in a network-instance based on match conditions and actions defined in a policy, as an alternative to forwarding based on entries in a routing table.

Each PBF policy is modeled as a sequence of rules, each of which has match conditions and actions. Match conditions specify values for various packet header fields. A packet matches a rule only if all the match conditions evaluate to true. Actions specify the processing to apply to each matching packet.

Each PBF policy is associated with a specific network-instance. The PBF rules only apply to the ingress IP packets on selected routed subinterfaces of the network-instance. Policy-forwarded packets are classified according to the DSCP policy that is attached to the ingress subinterface.

Match conditions for PBF policies

The following match conditions can be specified in a PBF policy:

- **dscp-set** – list of DSCP values to match for incoming packets; a packet must match one of the DSCP values defined in this list for the rule to apply
- **protocol** – protocol carried in the IP packet, specified either by name or IP protocol value
- **source-ip** – source IP address of the IP packet; for an IP-in-IP packet; this refers to the outer IP header source address

Actions for PBF policies

You can specify the following action in a PBF policy:

- **network-instance** – Look up matching packets in the network-instance referenced in the PBR policy instead of the network-instance associated with the subinterface.

4.1 Creating a PBF policy

Procedure

To create a PBF policy, configure the match conditions for the policy and the action to take for packets that meet the match conditions.

Example: Match based on IPv4 protocol value

The following example configures a PBF policy that applies to the default network-instance. On subinterfaces where this policy is applied, incoming IPv4 packets that have a value of 4 in their IP protocol field are looked up and forwarded in network-instance red.

```
--{ candidate shared default }--[  ]--
# info network-instance default policy-forwarding
    network-instance default {
        policy-forwarding {
            policy 100 {
                description "Sample PBF Policy"
                rule 1 {
                    action {
                        network-instance red
                    }
                    match {
                        ipv4 {
                            protocol 4
                        }
                    }
                }
            }
        }
    }
```

Example: Match based on DSCP values

In the following example, incoming packets matching DSCP values 0, 1, or 2 are looked up and forwarded in network-instance blue:

```
--{ * candidate shared default }--[  ]--
# info network-instance default policy-forwarding
    network-instance default {
        policy-forwarding {
            policy 101 {
                rule 1 {
                    action {
                        network-instance blue
                    }
                    match {
                        ipv4 {
                            dscp-set [
                                0
                                1
                                2
                            ]
                        }
                    }
                }
            }
        }
    }
```

Example: Match based on source IP prefix

In the following example, incoming packets whose source IP address matches prefix 10.10.0.0/16 are looked up and forwarded in network-instance green:

```
--{ candidate shared default }--[  ]--
# info network-instance default policy-forwarding
    network-instance default {
        policy-forwarding {
            policy 100 {
                rule 1 {
                    action {
                        network-instance green
                    }
                    match {
                        ipv4 {
                            source-ip {
                                prefix 10.10.0.0/16
                            }
                        }
                    }
                }
            }
        }
    }
```

4.2 Applying a PBF policy

Procedure

To activate a PBF policy, apply the policy to one or more routed subinterfaces of the network-instance configured in the policy.

Example

The following example applies a PBF policy to a subinterface in the default network-instance. The system evaluates ingress packets on the subinterface according to the match conditions in the policy and forwards the matching packets according to the action specified in the policy.

```
--{ candidate shared default }--[  ]--
# info network-instance default policy-forwarding
    network-instance default {
        policy-forwarding {
            interface ethernet-1/1.1 {
                apply-forwarding-policy 100
            }
        }
    }
```

5 Routing policies

The SR Linux supports policy-based routing. Policy-based routing controls the size and content of the routing tables, the routes that are advertised, and the best route to take to reach a destination.

Each routing policy has a sequence of rules (called entries or statements) and a default action. Each statement has numerical sequence identifier that determines its order relative to other statements in that policy. When a route is analyzed by a policy, it is evaluated by each statement in sequential order.

Each policy statement has zero or more match conditions and a base action (either accept or reject); the statement may also have route-modifying actions. A route matches a statement if it meets all of the specified match conditions.

The first statement that matches the route determines the actions that are applied to the route. If the route is not matched by any statements, the default action of the policy is applied. If there is no default action, then a protocol- and context-specific default action is applied.

All routes match a statement with no match conditions. When a route fulfills the match conditions of a statement, the base action of the statement is applied, along with all of its route-modifying actions.

5.1 Creating a routing policy

A routing policy consists of one or more statements that contain the following:

- Match conditions, such as protocol type, AS path length, and address family, against which routes are evaluated
- Actions, such as accept or reject, for routes that match the conditions in the statement

Each statement has a specified sequence number. If a policy has multiple statements, they are processed in sequence-number order. In addition, you can specify a default action that applies to routes that do not match any statement in the policy.

5.1.1 Specifying match conditions

Procedure

You can specify the following match conditions in a policy statement:

- Match routes by their protocol type – BGP, static, direct, host, IS-IS, OSPF, and so on.
- Match routes of any protocol by their address family – IPv4-unicast, IPv6-unicast, EVPN.
- Match routes of any protocol by their IPv4/IPv6 prefix.
- Match aggregate and BGP routes by their standard and large communities.
- Match BGP routes by their AS path length.
- Match BGP routes by their AS path encoding.
- Match BGP routes by whether the EVPN route has an Ethernet Segment identifier (ESI) that matches a member of a specified ESI set.

- Match BGP routes by whether the EVPN route has an Ethernet tag ID that matches a member of a specified Ethernet tag set.
- Match BGP routes by whether the EVPN type-2 route has a MAC address that matches a member of a specified MAC set.
- Match BGP routes by whether the originating router's IP address in an EVPN type-3 or type-4 route matches a specified IP address.
- Match BGP routes by whether the EVPN route is a specified route type (ethernet-ad, mac-ip, ethernet-segment, imet, or ip-prefix).
- Match BGP routes by whether the BGP VPN route has a specified route distinguisher value.
- Match IS-IS routes by their association with either Level1 (L1) or Level2 (L2).
- Match IS-IS routes by their route type (internal or external). An IPv4 route is internal if the prefix was signaled in TLV 128. An IPv4 route is external if the prefix was signaled in TLV 130. The encoding of TLV 236 indicates whether an IPv6 route is internal or external.
- Match IS-IS routes by their tag value.
- Match OSPF routes by their area ID.
- Match OSPF routes by their route type (intra-area, inter-area, type-1-ext, type-2-ext, type-1-nssa, type-2-nssa, summary-aggregate, or nssa-aggregate).
- Match OSPF routes by instance ID.
- Match routes based on EVPN route type.
- Match routes based on IP addresses and prefixes via prefix-sets for route types 2 and 5.
- Match routes based BGP encapsulation extended community.

If a statement has no match conditions defined, all routes evaluated by the policy are considered to be matches.

Example

The following example specifies BGP protocol as a match condition in a policy statement:

```
--{ candidate shared default }--[  ]--
# info routing-policy policy policy01
    routing-policy {
        policy policy01 {
            statement 100 {
                match {
                    protocol bgp
                }
            }
        }
    }
```

5.1.1.1 Specifying a list as a match condition

Procedure

You can specify a list of items, such as address prefixes, autonomous systems, BGP communities, and Ethernet tags as match criteria. To do this, you configure the list and include the list in a policy statement.

Example

The following example specifies a list of prefixes that can be used as a match condition in a policy statement:

```
--{ candidate shared default }--[  ]--
# info routing-policy
    routing-policy {
        prefix-set western {
            prefix 10.10.0.1/32 mask-length-range exact {
            }
            prefix 10.10.0.2/32 mask-length-range exact {
            }
            prefix 10.10.0.3/32 mask-length-range exact {
            }
            prefix 10.10.0.4/32 mask-length-range exact {
            }
        }
    }
```

5.1.2 Specifying actions

Procedure

The following are valid actions for routes that meet the match conditions in a statement:

- Accept the route.
- Reject the route.
- Modify the AS path of a BGP route by prepending additional AS numbers.
- Modify the AS path of a BGP route by deleting the current AS path and replacing it with a new AS path (a sequence of zero or more AS numbers).
- Add, delete, or replace standard and large communities attached to a BGP route.
- Modify the LOCAL_PREF attribute of a BGP route by specifying a new value.
- Modify the ORIGIN attribute of a BGP route by specifying a new value.
- Set the next-hop-self option for a BGP route.
- Set the next-hop-unchanged option for a BGP route.
- Add, delete, or replace the route tag associated with an IS-IS route.

Example

The following example specifies a policy with two statements. If a BGP route matches the first statement, the action is to specify a new value for the ORIGIN attribute of the BGP route. If the route matches the second statement, the action is to accept the route.

```
--{ candidate shared default }--[  ]--
# info routing-policy
    routing-policy {
        policy policy02 {
            statement 100 {
                match {
                    protocol bgp
                }
                action {
                    bgp {
                        origin {
                            set egp
                        }
                    }
                }
            }
            statement 101 {
                match {
                    prefix-set western
                }
                action {
                    policy-result accept
                }
            }
        }
    }
```

5.1.3 Specifying a default action

Procedure

You can optionally specify the policy action for routes that do not match the conditions defined in the policy. The default action can be set to all available action states including accept, reject, next-entry, and next-policy.

- If a default action is defined, and no matches occur with the entries in the policy, the default action is used.
- If no default action is specified, the default behavior of the protocol controls whether the routes match.

  For BGP, the default import action is to accept all routes from BGP. For internal routes, the default export action is to advertise them to BGP peers. For external routes, the default export action is not to advertise them to BGP peers.

Example

The following example defines a policy where the default action for non-matching routes is to reject them:

```
--{ candidate shared default }--[  ]--
# info routing-policy
    routing-policy {
        policy policy01 {
            default-action {
                policy-result reject
            }
        }
    }
```

5.2 Applying a routing policy

Procedure

Routing policies can be applied to routes received from other routers (imported routes), as well as routes advertised to other routers (exported routes). Routing policies can be applied at the network-instance level, peer-group level, and neighbor level.

Example

The following example specifies that BGP in the default network-instance applies `policy01` to imported routes:

```
--{ candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                import-policy policy01
                }
            }
```

Example

The following example applies `policy02` to BGP routes exported from the peers in peer-group `headquarters1`:

```
--{ candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                group headquarters1 {
                    export-policy policy02
                    }
                }
            }
        }
```

Example

The following example applies `policy02` to BGP routes exported from a specific BGP neighbor:

```
--{ candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                neighbor 192.168.11.1 {
                    export-policy policy02
                }
        }
    }
```

5.2.1 Applying a default policy to eBGP sessions

Procedure

You can specify the action to take for routes exported to or imported from eBGP peers to which no configured policy applies. This is set with the **ebgp-default-policy** command and the **export-reject-all** and **import-reject-all** parameters.

- The **export-reject-all** parameter, when set to **true**, causes all outbound routes that do not match a configured export policy to be rejected as if they had been rejected by a configured export policy. The default is **true**.
- The **import-reject-all** parameter, when set to **true**, causes all inbound routes that do not match a configured import policy to be rejected as if they had been rejected by a configured import policy. The default is **true**.

Example

The following example allows a BGP neighbor to export a default route even though the route is not subject to any configured policy:

```
--{ candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                ebgp-default-policy {
                    export-reject-all false
                    }
                neighbor 2001:db8::c11 {
                    send-default-route {
                        ipv6-unicast true
                    }
                }
        }
    }
```

5.2.2 Replacing an AS path

Procedure

You can configure a routing policy where the AS path in matching routes is replaced by a list of AS numbers specified in the policy. For routes that match the policy, the current AS path is deleted and replaced with an AS_SEQ element containing the AS numbers listed in the policy in their configured sequence.

If you configure an empty AS list in the policy, then the current AS path in a matching route is deleted, and it would then have a null AS_PATH attribute.

Example

The following is an example of a routing policy whose action is to replace the AS path in matching routes.

```
--{ candidate shared default }--[  ]--
# info routing-policy
    routing-policy {
        policy policy02 {
            statement 100 {
                match {
                    protocol bgp
                }
                action {
                    bgp {
                        as-path {
                            replace [
                                12
                                13
                                14
                            ]
                        }
                    }
                }
            }
        }
    }
```

Customer document and product support

Customer documentation
Customer documentation welcome page

Technical support
Product support portal

Documentation feedback
Customer documentation feedback