ERICSSON

Configuring ACLs

SYSTEM ADMINISTRATOR GUIDE

35/1543-CRA 119 1170/1-V1 Uen K

Contents

1 Overview

This document describes access control lists (ACLs) supported by the SmartEdge router and the tasks to configure, monitor, and administer them. This document also provides ACL configuration examples

Note: In the following descriptions, the term, controller card, applies to the Cross-Connect Route Processor Version 4 (XCRP4) Controller card, unless otherwise noted.

This document applies to both the Ericsson SmartEdge® and SM family routers. However, the software that applies to the SM family of systems is a subset of the SmartEdge OS; some of the functionality described in this document may not apply to SM family routers.

For information specific to the SM family chassis, including line cards, refer to the SM family chassis documentation.

For specific information about the differences between the SmartEdge and SM family routers, refer to the Technical Product Description *SM Family of Systems* (part number 5/221 02-CRA 119 1170/1) in the **`Product Overview`** folder of this Customer Product Information library.

1.1 IP Filtering ACLs

IP ACLs are lists of rules used to control the type of service that packets should receive. All IP ACLs are defined within a context.

1.1.1 IP ACL Applications

You can create IP ACLs to filter IPv4 and IPv6 traffic on subscriber circuits, control and administration traffic on traffic card circuits, the Ethernet management port, and in system administration.

IPv4 and IPv6 ACLs can coexist on contexts or interfaces and subscribers.

Use the following guidelines for creating filtering ACLs:

- Traffic card circuits—To filter packets in either the inbound or outbound direction on traffic card circuits, you apply an IP ACL to the interface to which the circuits are bound.
- (For IPv4 only) Ethernet untagged traffic, VLAN PVCs, and access link groups—To filter Ethernet untagged traffic, 802.1Q single-tagged or double-tagged VLAN (Q-in-Q) traffic, or access link group traffic, you apply an IP ACL to the Ethernet circuit or to the access link group; for the procedures, see *Configuring Link Aggregation*.

- Ethernet management port—To filter packets in either the inbound or outbound direction on the Ethernet management port on the active controller card, you apply an IP or IPv6 ACL to the interface to which the management port is bound. Both inbound and outbound filters are supported.

- Subscriber circuits—To filter packets in either the inbound or outbound direction for a subscriber circuit, you apply an IP ACL to the subscriber record, a named subscriber profile, or the default subscriber profile. Both inbound and outbound filters are supported.

  You must configure ACLs before bringing subscribers up.

- System administration—To filter inbound packets that are delivered to the kernel, you apply an IP ACL to the context. These ACLs are independent of the interface and circuit on which they were received.

  To ensure that all inbound packets are filtered before being delivered to the kernel, you must apply an IP ACL to each context that you have configured.

1.1.2 IP ACL Statements (Rules)

In IP ACLs, each rule defines the action, either permit or deny, to be taken for a packet if the packet satisfies the rule. A **`permit`** statement causes any packet matching the criteria to be accepted. A **`deny`** statement causes any packet matching the criteria to be dropped. A packet that does not match the criteria of the first statement is subjected to the criteria of the second statement, and so on, until the end of the IP ACL is reached; at which point, the packet is dropped due to an implicit **`deny any any`** statement at the end of every IP ACL.

Implicit statements can block valid access to a context. For example, in the local context, it could block administrator access to the Ethernet management port. To allow administrator access, add a statement to explicitly allow access from authorized sources.

Note: For IPv6, the recommended limitation is 100 rules for each ACL.

You can use the **`seq`** ***`seq-num`*** **`permit`** or **`deny`** constructs to establish a sequence order for the statements you are creating. If you use the **`permit`** or **`deny`** commands, which do not explicitly set the sequence, the system automatically assigns sequence numbers to the statements that you enter, in increments of 10. The first statement that you enter is assigned the sequence number of 10, the second is assigned the number 20, and so on. This allows room to assign intermediate sequence numbers to statements that you might want to add later. The assigned sequence numbers for the various statements are displayed in the output of the **`show configuration acl`** and **`show ip access-list`** commands.

If manually assigned sequence numbers leave no room for insertion of additional entries in the IP ACL, you can use the **`resequence ip access-list`** command (in context configuration mode) to reassign the

sequence numbers so that they are in increments of 10. Use the **no seq** ***seq-num*** construct to remove an individual statement from the IP ACL.

1.1.3 IP ACL Packet Filtering

Based on the rules specified in the ACLs associated with the packet, the SmartEdge router decides whether the packet is forwarded or dropped. Statement criteria include all Internet protocols and can be specified by the protocol numbers established in RFC 1700, `Assigned Numbers`. A subset of these options can also be specified by keyword.

The last twenty IPv4 dropped packets are counted and logged if you enable the count and log functions when you apply an IP ACL. However, IPv6 packets can be counted, but not logged.

By default, these functions are disabled because they have an impact on system performance. We recommend that you only enable them when they are required for diagnostic purposes.

The SmartEdge router uses IP ACLs to filter packets in the following order:

1 ACLs applied to interfaces for inbound traffic on traffic card circuits and the Ethernet management port

2 ACLs applied to subscriber records and profiles for inbound traffic on subscriber circuits

3 ACLs applied to contexts for administration (inbound only)

4 ACLs applied to outbound traffic on traffic card circuits and the Ethernet management port

5 ACLs applied to subscriber records and profiles for outbound traffic on subscriber circuits

1.1.4 Dynamic IP Filter ACLs Using RADIUS ADF Attribute

Dynamic IP filter ACLs allow IP ACL packet filtering to be downloaded from a Remote Authentication Dial-In User Service (RADIUS) server. A dynamic IP filter ACL consists of a set of rules, each of which is contained in an instance of RADIUS IETF attribute 242, Ascend_Data_Filter (ADF).

Use the following guidelines for using the RADIUS ADF attribute:

- IPv6 ACL ADF attribute is configured with IETF 242 using type, `ipv6`.
- ADF can be configured in a RADIUS record or in an RSE profile; the order of precedence is RADIUS record, RSE profile).
- Events such as authorization and reauthorization are supported with IPv6 ADF.

- We recommend limiting each ADF to 12 rules per protocol, per direction.
- ADF is logically concatenated after Filter-Id; it is looked up after the filter ID in processing.
- The same RADIUS Vendor-Specific Attribute (VSA) is applied for both IPv4 and IPv6 ACLs.

For more information about RADIUS VSA 242, see the *Other Supported VSAs* section in *RADIUS Attributes*.

1.1.5 Applying IPv6 Filter ACLs Using RADIUS Filter-Id Attribute

Use the **`in_v6:acl_in_v6`** command or the **`out_v6:acl_out_v6`** command when applying an IPv6 filter ACL through the RADIUS IETF attribute 11, Filter-Id. When applying an IPv6 filter ACL, use the following rules and limitations:

Note: The RADIUS attribute 11 has the same limitation as the corresponding IPv4 filter ID.

- IPv6 ACL attribute is configured with filter ID IETF 11, using the type `ipv6`.
- At least one and up to 10 IPv6 filtering ACLs should be configured.
- ACLs are separated by a semicolon in the string.
- One ACL name cannot exceed 39 characters.
- The string cannot exceed 253 characters (ACL name and Filter-ID direction and protocol).
- All access-list rules are logically concatenated in configured order.
- The filter ID can be configured in a RADIUS record or through the CLI in a subscriber record or profile, or an RSE profile; the order of precedence is RADIUS, RSE profile, subscriber record, profile, default profile.
- All events such as COA, authorization, reauthorization, and policy refresh are supported with IPv6 filter IDs.

1.1.6 IP ACL Configuration Guidelines

The following filtering rules apply to static and dynamic IPv4 and IPv6 ACLs:

- You can apply IP ACLs to interfaces, contexts, and subscriber records. Administrative access control is context-specific. To ensure that all inbound packets are filtered, you must apply an IP ACL to each configured context.
- Each IP ACL has an implicit **`deny any any`** statement added at the end. If a packet does not match any explicit filter statement in the list, it is dropped.

Unlike the explicit statements in the ACL, this implicit final statement is not displayed in the output of the **show configuration acl** or **show ip access-list** command (in any mode).

This implicit statement could block valid access to a context. For example, in the local context, it could block administrator access to the Ethernet management port. To allow administrator access, add a statement to explicitly allow access from authorized sources. For example, you could add a **permit ipv6 any any** or **permit ipv6** ***src src-wildcard dest dest-wildcard*** statement.

- (IPv6 only) For IP ACLs applied in contexts with the **ipv6 admin-access-group** command, the access list has an implicit rule enabling IPv6 Neighbor Discovery. The implicit rule is added after all explicit rules and before implicit deny rule. This rule can be overwritten by explicit permit statements for Neighbor Discovery packets.

  Note: The implicit rule enabling IPv6 Neighbor Discovery is added automatically by the OS but not shown in **show** command output or the configuration.

- (IPv4 and IPv6) If you apply an IP ACL to a multibind interface, it does not affect the IP traffic on the subscriber sessions that are bound to that interface; the ACL is applied only to the IP traffic on circuits that are statically bound to the interface using the **bind interface** command (in the circuit's configuration mode).

- (IPv4 and IPv6) If a non-existent IP ACL is applied to an interface, then all packets are forwarded with no filtering.

- (IPv4 and IPv6) If a non-existing IP ACL is applied to a single stack subscriber, then the subscriber session will not come up; this restriction also applies if a non-existent ACL is applied to a Remote Authentication Dial-In User Service (RADIUS) attribute. For dual stack subscribers, only a stack corresponding to an ACL protocol will not come up.

1.2 Policy ACLs

A policy ACL is a list of rules, each of which refers to a class that is defined in the policy (Policy ACLs can be used with IPv4 or IPv6 traffic). A policy ACL, unlike an IP ACL, does not define the action for each rule; instead, the action for each class is determined by the policy to which the policy ACL is applied. All policy ACLs are defined within a context.

If a packet does not match any classifying rule, it is considered to belong to the policy-based class.

IPv6 policy ACLs have the following restrictions:

- Packets dropped by IPv6 ACL are not logged.

- The number of rules in an IPv6 ACL is limited to 100 for Administrative ACLs and ACLs on PPA2 cards.
- Packets dropped by IPv6 ACL do not generate an ICMP Administrative Prohibited message.
- The **`ipv6 access-list`** command does not support the **`length`** parameter in rules (both filter and policy IPv6 ACLs).
- The IPv6 address mask is configured only as a prefix and is always contiguous.
- The **`modify condition`** command is available only for IPv4 ACLs.

For IPv6 filter and policy access lists, you can configure the following parameters in rules:

- IPv6 source and destination IP address with a prefix; such as `2001:a:b::1/48`.
- Source and destination ports for TCP and UDP protocol.
- ICMP type and code for ICMP protocol and IGMP type for IGMP protocol.
- The `protocol` parameter (with values of 1-255; 0 is a wildcard) excludes the following extension headers: `ahp`, `destination`, `esp`, `fragment`, `hop-by-hop`, `routing`, `none`).
- Traffic class instead of TOS and DSCP.
- Policy access-list rules have only the **`permit`** option and must use the same names for classes as in IPV4 access-list rules.

1.2.1 Policy ACL Applications

You can apply a policy ACL to class-based forwarding, Network Address Translation (NAT), or quality of service (QoS) policies to classify your policy ACL in different classes. When applied to a class-based policy, a policy ACL allows different actions to be applied to different classes of packets.

Note: If a non-existent policy ACL is applied to a forward policy, NAT policy, QoS metering policy, or QoS policing policy, it is ignored and packets are forwarded according to a policy action with no classification.

This does not apply to subscribers. It applies only when a policy ACL is applied to a static circuit.

For information about forward policies, see *Configuring Forward Policies*. For information about NAT policies, see *Configuring NAT Policies*. For information about QoS policing and metering policies, see *Configuring Rate-Limiting and Class-Limiting*.

1.2.2 Policy ACL Statements

A policy ACL uses **permit** statements to define how packets are assigned to classes. A packet that does not match the criteria of the first statement is subjected to the criteria of the second statement, and so on, until the end of the policy ACL is reached; at which point, the packet is assigned to the default class.

You can use the optional **seq** ***seq-num*** construct with any **permit** statement to establish a sequence number for the statement. If you do not use the **seq** ***seq-num*** construct, the system automatically assigns sequence numbers to the statements that you enter, in increments of 10. The first statement you enter is assigned the sequence number of 10, the second is assigned the number 20, and so on. This allows room to assign intermediate sequence numbers to statements that you add later. The assigned sequence numbers for the various statements are displayed in the output of the **show configuration acl**, **show configuration policy**, and **show policy access-list** commands.

If manually assigned sequence numbers leave no room for insertion of additional entries in the policy ACL, you can use the **resequence policy access-list** command (in context configuration mode) to reassign the sequence numbers. The **no seq** ***seq-num*** construct removes an individual statement from the policy ACL.

1.2.3 Policy ACL Packet Filtering

Statement criteria for filtering includes all Internet protocols, which can be specified by the protocol numbers established in RFC 1700, *Assigned Numbers*. A subset of these options can also be specified by keyword. Based on classification, a class-based policy defines the type of action to be performed on the packets in a particular class. All packets that match the criteria can be counted by the statement if you enable the count when you apply a policy ACL. By default, the counting of packets is disabled because this function has an impact on system performance. We recommend that you enable counting only when required for diagnostic purposes.

1.2.4 Dynamic Policy ACL RADIUS VSA 164 Guidelines

A Dynamic Policy Filter (DPF), also known as a Dynamic Policy ACL, allows a class-based policy to be governed by a policy ACL that is downloaded from a RADIUS server. A dynamic policy ACL consists of a set of classification rules, each of which is contained in a RADIUS VSA 164 instance. The same VSA is used for IPv4 and IPv6 policy ACLs.

All rules in all dynamic policy ACLs are downloaded in a single RADIUS message. Class-based policies configured with dynamic ACLs are referred to as RADIUS-guided policies. Traditional policy ACLs and class-based policies are referred to as static policy ACLs and static policies, respectively.

The following guidelines govern the use of RADIUS VSA 164 for policy ACLs:

- Dynamic policy ACLs are defined on a RADIUS server and downloaded using one or more instances of VSA 164.
- Each downloaded VSA 164 instance contains one classification rule.
- A VSA 164 instance can be configured only in a RADIUS record or in an RSE profile.
- All VSA 164 instances that have the same service (forward, NAT, or QoS) and the same direction are considered to be rules of a dynamic policy ACL for that service.
- The rules in a dynamic policy ACL are sequenced by the order in which VSA 164 instances appear in a subscriber record.

Note: For more information about vendor VSAs provided by Ericsson AB, see the *Vendor VSAs Provided by Ericsson AB* section in *RADIUS Attributes*.

Note: There is no IPv6 VSA 164 for NAT.

1.2.5 Dynamic QoS Parameter With RADIUS-Guided Policies

The Dynamic QoS Parameter (DQP, RADIUS VSA 196) attribute, `ipv4-fwd-in-access-group` or `ipv6-fwd-in-access-group`, provides a mechanism to combine multiple IPv6 policy access lists into an access group to be applied to a subscriber session's inbound RADIUS-guided (RG) forward policy.

All DQP parameters supported for IPv4 ACL are supported for IPv6 ACL. All existing limitations that applied to IPv4 DQP are applied to IPv6 DQP. This includes configuring a QoS or forward policy as an RG policy in order to work with the corresponding Dynamic Policy Filter (DPF) (VSA 164) to bring up subscribers. For example, when both a forward DQP and a forward DPF are applied to one subscriber, then the total rules number in the look-up is a summary of the number of DPF rules and the number of access-list rules in the referred DQP.

The **`fwd-in-access-group`** (for IPv4) and **`ipv6-fwd-in-access-group`** (for IPv6) commands can be applied on the same subscriber simultaneously or one at a time.

1.2.5.1 DQP ipv6-fwd-in-access-group Used With a Forward Policy

You can use the DQP**`ipv6-fwd-in-access-group`** command with a forward policy with the following rules and limitations:

- At least one, but no more than 10 policy ACLs should be configured with DQP.

- ACLs are separated by a semicolon in the string.
- A string cannot exceed 247 characters (ACL name and DQP parameter name).
- One ACL name cannot exceed 39 characters.
- All access-list rules are logically concatenated in configured order.
- DQP admission control should be protocol dependent, so you must verify the corresponding access-group configuration under the policy.
- Forward policies support redirect to local (HTTP redirect) and drop modes for IPv6 traffic and should be applied in the IN direction.
- When DQP is applied with DPF, then DPF rules are logically concatenated to the end of all access-lists in DQP.
- Time-range conditions configured in policy ACLs used in DQP will not take effect.
- Time-range conditions will not take effect when configured in an access-group that consists of multiple access-lists.
- DQP and filter ACLs may consist of one or more access-lists.
- Both IPv4 and IPv6 traffic can be classified into the same class, with per-class rate limits enforced on the aggregate of both kinds of traffic.
- QoS or forward policies with IPv6 policy ACLs can be applied to both static and subscriber circuits.

1.2.5.2 RG Policy Guidelines

Configuration guidelines for RG policies include:

- RG policies are only applied to subscribers.
- RG policies can have ACLs configured on them or use RADIUS-configured ACLs (DPF or DQP `ipv6-fwd-in-access-group`).
- A corresponding forward policy must be configured with an RG, without the access-group name and with at least one class.
- The policy must be an RG policy if DPF or DQP forward access groups are configured.
- You cannot configure the DQP attribute `ipv6-fwd-in-access-group` if an RG forward policy has a static ACL configured.
- The DPF attribute is applied as a secondary lookup.

1.3 ACL Counters and Clearing Counters

Circuit IPv6 ACL drop counters are persistent during circuit life per protocol. They are applied without configuration.

The best way to verify traffic is to enable rule counters. Use them if you understand traffic patterns and performance is not an issue.

Use the following guidelines for IPv6 ACL counters:

- IPv6 ACL counters impact traffic performance.
- IPv6 ACL counters are cleared with any configuration update.
- To clear IPv6 filtering ACL counters at other times, use the **clear access-group ipv6** ***service-type*** **circuit {in | out}** command; for example, to clear IPv6 policy ACL counters, use the **clear access-group ipv6 ipv6-policy** command.
- IPv6 ACL counters should be applied to subscribers before subscribers are brought up.
- IPv6 ACL rule counters cannot be applied if service accounting is configured.

2 Configuration and Operations Tasks

Note: In this section, the command syntax in the task tables displays only the root command; for the complete command syntax, see *Command List*.

To configure ACLs, perform the tasks described in the following sections:

2.1 IPv4 Filtering ACLs

The following sections describe how to configure an IPv4 ACL and how to apply an IP ACL for IPv4 traffic.

2.1.1 Configuring an IPv4 ACL

To configure an IPv4 ACL, perform the following tasks; enter all commands in access control list configuration mode, unless otherwise noted:

1. Create or select an ACL and enter access control list configuration mode to add options and rules using the *ip access-list* command in context configuration mode.

2. Optional. Associate a description with an IP ACL using the *description (ACL)* command.

3. Create an ACL statement using **`permit`** conditions, using the *permit* or *seq* command.

   To explicitly set its order in the list, use the **`seq permit`** construct for each statement.

4. Create an ACL statement using **`deny`** conditions using the *deny* or *seq* command.

   To explicitly set its order in the list, use the **`seq deny`** construct for each statement.

5. Optional. Create an ACL condition using a unique ID and access ACL condition configuration mode with the *condition* command.

   Enter the following commands, as appropriate:

   - Optional. Configure absolute time condition statements using the *absolute* command in ACL condition configuration mode.

     An absolute time ACL statement redefines an ACL rule's action for only one specific time interval.

- Optional. Configure periodic time condition statements using the *periodic* command in ACL condition configuration mode.

  A periodic time ACL statement redefines the ACL rule action for a recurring time interval.

6. Optional. Re-sequence statements in an IP ACL using the *resequence ip access-list* command in context configuration mode.

2.1.2 Applying an IP ACL for IPv4 Traffic

To apply an IPv4 ACL to packets associated with a context, circuit, access link group, or interface, or a subscriber record, named profile, or default profile, perform the following tasks, as appropriate.

1. Apply an IP ACL to an interface or to a subscriber record, named profile, or default profile with RADIUS IETF attribute 11 using the *ip access-group)* command in interface or subscriber configuration mode.

2. Apply an IP ACL to a context, using the *admin-access-group* command in context configuration mode.

3. (For IPv4 traffic only) For the procedures to apply an IP ACL to packets associated with Ethernet untagged circuits, 802.1Q single or double VLAN PVCs, or an access link-group, see *Configuring ATM, Ethernet, and POS PortsConfiguring Circuits*, or *Configuring Link Aggregation*.

2.2 IPv6 Filtering ACLs

The following sections describe how to create and apply a filtering ACL for IPv6 traffic and subscribers.

You can apply an IPv6 filtering ACL to packets associated with a context or an interface or to a subscriber record, named profile, or default profile.

2.2.1 Configuring an IPv6 ACL

To configure an IPv6 ACL, perform the following tasks; enter all commands in access control list configuration mode, unless otherwise noted.

1. Create or select an ACL and enter access control list configuration mode using the *ipv6 access-list* command in context configuration mode.

   For IPv6, the recommended limit is 100 rules for each IPv6 ACL.

   The *ipv6 access-list* command does not support the length parameter in rules (for both filter and policy ACLs).

2. Optional. Associate a description with an IP ACL using the *description (ACL)* command.

3. Optional. Create an ACL statement using **`permit`** conditions using the *permit* or *seq* command.

   To explicitly set its order in the list, use the **`seq permit`** construct for each statement.

4. Optional. Create an ACL statement using **`deny`** conditions using the *deny* or *seq* command.

   To explicitly set its order in the list, use the **`seq deny`** construct for each statement.

5. Optional. Create an ACL condition using a unique ID and access ACL condition configuration mode using the *condition* command.

   The **`condition`** command is not supported on IPv6 ACLs applied on the Ethernet Management port; conditions are ignored on that port.

6. After entering the previous command, you can enter the optional *absolute* or *periodic* commands in ACL condition configuration mode.

   An absolute time ACL statement redefines an ACL rule action for only one specific time interval and a periodic time ACL statement redefines the ACL rule action for a recurring time interval.

7. Optional. Re-sequence statements in an IP ACL using the *resequence ip access-list* command in context configuration mode.

2.2.2 Applying an IP ACL for IPv6 Traffic

To apply an IPv6 ACL to packets associated with an interface or context, or a subscriber, perform the appropriate task:

1. To filter control and administrative traffic, apply an administrative IPv6 ACL to a context, use the *ipv6 admin-access-group* command in context configuration mode.

2. To apply an IPv6 ACL to an interface bound to a port, used to filter control and administrative traffic, use the *ipv6 access-group (interface)* command in interface configuration mode.

3. To apply an IPv6 ACL to a subscriber record, named profile, or default profile, use the *ipv6 access-group (subscriber)* command in subscriber configuration mode.

2.3 Enabling IPv4 ACL Counters or Logging for a Subscriber

Warning!

Enabling ACL counters can impact SmartEdge router performance. Use them carefully.

To enable ACL filtering counters or logging for a subscriber through the subscriber record, the default subscriber profile, or a named subscriber profile, perform the following tasks, as appropriate:

1. Enable IPv4 ACL counters or logging for a subscriber record, the default subscriber profile, or a named subscriber profile using the *access-list count ip policy* or *access list log ip* command in subscriber configuration mode.

This command is not applied to currently active subscribers.

2.4 Enabling IPv6 ACL Counters

Warning!

Enabling IPv6 ACL and forwarding related counters can impact SmartEdge router performance. Use them carefully.

You can enable IPv6 ACL counters in two ways:

- Filtering ACL Counters—To enable IPv6 filtering ACL counters for a subscriber through the subscriber record, the default subscriber profile, or a named subscriber profile, enter the *access-list count ipv6 ipv6-policy* command in subscriber configuration mode. For both IPv4 and IPv6, this command is not applied to active subscribers. The following example enables ACL counters for a default subscriber policy:

```
[local]Redback(config-ctx)#subscriber default
[local]Redback(config-sub)#qos policy metering pol23
[local]Redback(config-sub)#access-list count ipv6-policy
```

- Policy ACL Rule Counters—To enable IPv6 policy ACL rule counters for policing, metering, and forward policy classification, add the **acl-counters** keyword to the policy configuration. The following example enables counters for a QoS metering policy at the port level:

```
[local]Redback(config)#port ethernet 4/1
[local]Redback(config-port)#qos policy meterin
g pol23 ipv6 acl-counters
```

Note: You enter **show** commands at the level where the counters were configured to view the counters. For example, for subscriber circuits, enter the **show** command specifying the subscriber circuit. For ACL counters enabled on a port, specify the port in the **show** command. For example, if you enabled ACL counters on the QoS policy on a port, entering the **show circuit counters** ***slot/port*** **detail** command shows how many packets passed through the policy processing on that port.

2.5 Modify IPv4 ACL Conditions in Real Time

To modify the action for a condition referenced by an IPv4 ACL, in real time, without requiring the reconfiguration of the ACL condition statements, enter the *modify ip access-list* command in exec mode.

This function is not supported with IPv6 conditions.

2.6 IPv4 Policy ACLs

The following sections describe how to configure, apply, and modify an IPv4 policy ACL.

2.6.1 Configuring an IPv4 Policy ACL

To configure a static policy ACL, perform the following tasks; enter all commands in access control list configuration mode, unless otherwise noted.

1. Create or select a policy ACL and enter access control list configuration mode using the *policy access-list* command in context configuration mode.

2. Optional. Associate a description with a policy ACL using the*description (ACL)* command.

3. Create policy ACL statements to allow packets that meet the specified criteria, using the *permit* command.

   Enter this command multiple times to specify multiple classes.

4. Optional. Create a policy ACL condition using a unique ID and access ACL condition configuration mode using the *condition* command.

5. Enter the following commands in ACL condition configuration mode.

   You can create up to seven conditions in a policy ACL.

- Optional. Configure absolute time condition statements using the *absolute* command.

  An absolute time ACL condition statement applies an ACL rule for only one specific time interval.

- Optional. Configure periodic time condition statements using the *periodic* command.

  A periodic time ACL statement applies an ACL rule for a recurring time interval.

6. Optional. Re-sequence statements in a policy ACL using the *resequence policy access-list* command in context configuration mode.

2.6.2 Applying an IPv4 Policy ACL

To apply a policy ACL to packets associated with a forward policy, a NAT policy, or QoS metering or policing policy, and complete the configuration of the policy, perform the tasks described in *Configuring Forward Policies*, *Configuring NAT Policies*, and *Configuring Rate-Limiting and Class-Limiting* respectively.

2.6.3 Modifying IPv4 Policy ACL Conditions in Real Time

To modify the action for a class name referenced by a policy ACL in real time, without requiring the reconfiguration of the ACL condition statements, enter the *modify policy access-list* command in exec mode.

2.7 IPv6 Policy ACLs

The following sections describe how to configure, apply, and modify an IPv6 policy ACL with an optional HTTP redirect profile.

2.7.1 Configuring an IPv6 Policy ACL

To configure an IPv6 policy ACL, perform the following tasks; enter all commands in access control list configuration mode, unless otherwise noted.

1. Create or select an IPv6 policy ACL and enter access control list configuration mode using the *ipv6 policy access-list* command in context configuration mode.

2. Optional. Associate a description with an IPv6 policy ACL using the *description (ACL)* command

3. Create IPv6 policy ACL statements to allow packets that meet the specified criteria using the *permit* command.

   Enter this command multiple times to specify multiple classes.

4. Optional. Create an IPv6 policy ACL condition using a unique ID and access ACL condition configuration mode using the *condition* command

   After you enter this command, you can enter the following optional commands. You can create up to 12 conditions in an IPv6 policy ACL.

   a *absolute*

      An absolute time ACL condition statement applies an IPv6 ACL rule for only one specific time interval.

   b *periodic*

      A periodic time ACL statement applies an IPv6 ACL rule for a recurring time interval.

5. Optional. Re-sequence statements in an IPv6 policy ACL using the *resequence ipv6 policy access-list* command in context configuration mode.

2.7.2 Applying an IPv6 Policy ACL

To apply an IPv6 policy ACL to packets associated with a forward policy or QoS metering or policing policy, and complete the configuration of the policy, enter the *ipv6 access-group (policy)* command in forward policy, metering policy, or policing policy configuration mode. For more information, see *Configuring Forward Policies* and *Configuring Rate-Limiting and Class-Limiting*.

2.8 Operations Tasks

To monitor, troubleshoot, and administer IP and policy ACLs, perform the tasks described in Table 1. Enter the **`clear`** and **`debug`** commands in exec mode; enter the **`show`** commands in any mode.

Table 1 ACL Operations Tasks

Task	Root Command	Notes
Clear counters for filtering ACLs.	*clear access-group*	
Enable the generation of debug messages for IP classifier processes.	*debug cls*	
Enable the generation of aaa (authentication, authorization, and accounting) debug messages and counters for IPv6 ACL.	*debug aaa*	

Table 1 ACL Operations Tasks

Task	Root Command	Notes
Enable the generation of qos (quality of service) debug messages and counters for IPv6 ACL.	*debug qos*	
Enable the generation of debug messages for IP ACL events (IPv4).	*debug ip access-list*	
Enable the generation of debug messages for IP ACL events (IPv6).	*debug ipv6 access-list*	
Enable the generation of debug messages for policy ACLs.	*debug policy access-list*	
Display information about all ACLs and the entities to which the ACLs are applied.	*show access-group*	
Display information about all ipv6 ACLs and the entities to which the ACLs are applied. Includes keywords for forwarding, filtering, and QoS.	*show access-group ipv6*	This command is context specific; enter it in the context where the access-group is configured.
Display information about the quality of service (qos) configuration.	*show configuration qos*	
Display the IP ACL configuration.	*show configuration acl*	
Display the policy ACL configuration.	*show configuration policy (ACL)*	
Display the status of configured IPv4 ACLs.	*show ip access-list*	
Display the status of configured IPv6 ACLs.	*show ipv6 access-list*	
Display the status of configured policy ACLs.	*show policy access-list*	

Table 1 ACL Operations Tasks

Task	Root Command	Notes
Verify configured policies (with ACLs) were downloaded and applied on line cards.	*show qos policy policing*	The output shows on which line cards the policy was applied.
Verify that ACLs and their rules for a subscriber were configured correctly.	*show subscriber active*	The output can confirm that DQP and DPF were applied.

3 Configuration Examples

This section provides examples to configure, add, and re-sequence ACL statements, configure absolute and periodic time condition statements, configure an IP ACL, and configure a policy ACL with a forward policy, NAT policy, or QoS policing policy.

3.1 Configure an ACL Statement

The following example configures a policy ACL to prioritize **web** and voice-over-IP (**VoIP**) traffic.

In this example, the order of the condition statements is assigned automatically by the SmartEdge OS. To determine the list order, use the **`seq permit`** construct for each one.

```
[local]Redback(config-ctx)#policy access-list QoSACL-1
[local]Redback(config-access-list)#permit tcp any any eq 80 class Web
[local]Redback(config-access-list)#permit udp any any eq 1000 class VOIP
[local]Redback(config-access-list)#permit any any class default
```

The following example uses a policy ACL to define classes of traffic to be mirrored:

```
[local]Redback(config-ctx)#policy access-list PBR_ACL
[local]Redback(config-access-list)#seq 10 permit tcp any eq www any class WEB
[local]Redback(config-access-list)#seq 20 permit tcp any any eq www class WEB
[local]Redback(config-access-list)#seq 30 permit udp any class UDP
[local]Redback(config-access-list)#seq 40 permit ip any class IP
```

The following example specifies that all IP traffic to destination host **10.25.1.1** is to be denied, and all other traffic on subnet 10.25.1/24 is to be permitted:

```
[local]Redback(config-ctx)#ip access-list protect201
[local]Redback(config-access-list)#deny ip any host 10.25.1.1
[local]Redback(config-access-list)#permit ip any 10.25.1.0 0.0.0.255
```

3.2 Add an ACL Statement

The following example shows how to use the **seq** keyword to modify the existing **tc1** ACL, adding a statement between the statements with sequence numbers 20 and 30:

```
[local]Redback#configure
[local]Redback(config)#context local
[local]Redback(config-ctx)#ip access-list tc1
[local]Redback(config-access-list)#seq 25 deny tcp 10.10.10.4 0.0.0.0 any eq 80
```

The output of the **show configuration acl** command now includes the new statement, with sequence number **25:**

```
!
  ip access-list tc1
    description This is a sample access control list
    seq 10 deny ip host 10.10.10.2 host 10.10.20.2
    seq 20 deny tcp host 10.10.10.3 any eq www
    seq 25 deny tcp host 10.10.10.4 any eq www
    seq 30 deny udp host 10.10.10.3 any
    seq 40 deny ip host 10.10.10.4 any
    seq 50 deny ip host 10.10.10.5 any
    seq 60 permit ip any any
```

3.3 Resequence ACL Statements

The following example displays the current sequencing of an IP ACL:

```
[local]Redback#show configuration acl

Building configuration...
!
  ip access-list tc1
    description This is a sample access control list
    seq 10 deny ip host 10.10.10.2 host 10.10.20.2
    seq 20 deny tcp host 10.10.10.5 any eq telnet
    seq 25 deny tcp host 10.10.10.4 any eq www
    seq 30 deny udp host 10.10.10.3 any
    seq 50 deny ip host 10.10.10.5 any
    seq 60 permit ip any any
```

The following example re-sequences the statements in the IP ACL to increments of 10 and displays the new sequence of statements:

```
[local]Redback(config)#context local
[local]Redback(config-ctx)#resequence ip access-list tc1

[local]Redback#show configuration

Building configuration...
Current configuration:
context local
  ip access-list tc1
    description This is a sample access control list
    seq 10 deny ip host 10.10.10.2 host 10.10.20.2
    seq 20 deny tcp host 10.10.10.5 any eq telnet
    seq 30 deny tcp host 10.10.10.4 any eq www
    seq 40 deny udp host 10.10.10.3 any
    seq 50 deny ip host 10.10.10.5 any
    seq 60 permit ip any any
```

3.4 Configure an Absolute Time Condition Statement

The following example configuration creates an absolute time ACL condition statement for ACL condition **342**, which is defined in the IP ACL, **ip-acl-1**. The absolute time ACL condition applies a **deny** action to all IP ACL statements that

reference the ACL condition for the time interval beginning on December 15, 2003 at 9:00 p.m. (**21:00**) and ending on the same day at 11:00 p.m. (**23:00**):

```
[local]Redback(config-ctx)#ip access-list ip-acl-1
[local]Redback(config-access-list)#condition 342 time-range
[local]Redback(config-acl-condition)#absolute start 2003:12:15:21:00 end 2003:12:15:23:00 deny
```

3.5 Configure a Periodic Time Condition Statement

The following example configuration creates a periodic ACL condition statement for the ACL condition **101**, which is referenced by the IP ACL, **ip-acl-2**, such that all packets traveling between 9 a.m. and 5 p.m. (9:00 to 17:00 in 24-hour format) on weekdays are permitted:

```
[local]Redback(config-ctx)#ip access-list ip-acl-2
[local]Redback(config-access-list)#condition 101 time-range
[local]Redback(config-acl-condition)#periodic weekdays 9:00 to 17:00 permit
```

The following example configuration creates a periodic ACL condition statement for the ACL condition **342**, which is referenced by the policy ACL **policy_acl_1**, such that all packets traveling every weekday (Monday to Friday) from 9:00 p.m. to 11:00 p.m. (9:00 to 23:00 in 24-hour format) are permitted:

```
[local]Redback(config-ctx)#policy access-list policy_acl_1
[local]Redback(config-access-list)#condition 342 time-range
[local]Redback(config-acl-condition)#periodic weekdays 21:00 to 23:00 permit
```

3.6 Configure an IP ACL to Filter IPv4 Traffic

The following example configuration creates an IPv4 ACL, **tc4** with deny and permit statements. In this case, the SmartEdge OS orders the list sequence. If you wanted to determine the order of the statements, use the **`seq deny`** and **`seq permit`** commands.

```
[local]Redback(config-ctx)#ip access-list tc4
[local]Redback(config-access-list)#description This is a sample IP access control list
[local]Redback(config-access-list)#deny ip 10.10.10.2 0.0.0.0 10.10.20.2 0.0.0.0
[local]Redback(config-access-list)#deny tcp 10.10.10.3 0.0.0.0 any eq 80
[local]Redback(config-access-list)#deny udp 10.10.10.3 0.0.0.0 any
[local]Redback(config-access-list)#deny ip 10.10.10.4 0.0.0.0 any
[local]Redback(config-access-list)#deny ip 10.10.10.5 0.0.0.0 any
[local]Redback(config-access-list)#permit ip any any
```

3.7 Configure an IP ACL to Filter IPv6 Traffic

The following example configuration creates an IPv6 ACL, **tc6**.

```
[local]Redback(config-ctx)#ipv6 access-list tc6
[local]Redback(config-ipv6-access-list)#description This is a sample IPv6 access control list
[local]Redback(config-ipv6-access-list)#seq 12 deny tcp 22:1:1::2/128 any traffic-class eq df
[local]Redback(config-ipv6-access-list)#seq 20 deny udp any any range 80 81
[local]Redback(config-ipv6-access-list)#seq 20 deny tcp any 5:6:7:8:9::/120 eq ftp established
[local]Redback(config-ipv6-access-list)#seq 900 permit ipv6 any any
```

3.8 Apply IPv4 and IPv6 ACLs to the Same Interface

The following example configuration shows the two IPv4 and IPv6 ACLs in Section 3.6 on page 24and Section 3.7 on page 25 applied to the same interface in a context:

```
[local]Redback(config-access-list)#exit
[local]Redback(config-ctx)#interface oc1

[local]Redback(config-if)#ip access-group tc4 in count log
[local]Redback(config-if)#ipv6 access-group tc6 in count
```

3.9 Configure and Apply IPv6 Filtering ACLs for Subscriber Traffic

The following example creates an IPv6 filtering ACL, `acl2`, (in the `local` context) and applies it to the default subscriber profile:

```
[local]Redback(config-ctx)#ipv6 access-list acl2
[local]Redback(config-ipv6-access-list)#seq 10 permit ipv6 1:2:3:4::/48
[local]Redback(config-ipv6-access-list)#seq 20 permit ipv6 1:2:3:4::5/64
[local]Redback(config-ipv6-access-list)#exit
[local]Redback(config-ctx)#subscriber default
[local]Redback(config-sub)#ipv6 access-group acl2 in
```

3.10 Configure IPv4 and IPv6 Policy ACLs and Associate Them With a Forward Policy

The following example shows how to configure and apply IPv4 and IPv6 policy ACLs for forward policy classification:

Note: Forward policies currently only support the following actions for IPv6 packets: HTTP redirect and drop. Since HTTP redirect actions are only supported for subscriber circuits, there is no benefit to applying a forward policy to a non-subscriber circuit which carries only IPv6 traffic.

```
[local]Redback(config)#context isp

[local]Redback(config-ctx)#!

[local]Redback(config-ctx)#interface subs multiband
```

```
[local]Redback(config-if)#ip address 17.1.1.1/21
[local]Redback(config-if)#ipv6 address 2001:a:b::1/48
[local]Redback(config-if)#ip pool 17.1.1.0/24
[local]Redback(config-if)#!
[local]Redback(config-if)#aaa authentication subscriber radius
[local]Redback(config-ctx)#!
[local]Redback(config-ctx)#policy access-list find_tcp
[local]Redback(config-access-list)#seq 10 permit tcp any any class tcp
[local]Redback(config-access-list)#!
[local]Redback(config-access-list)#ipv6 policy access-list find_tcp
[local]Redback(config-ipv6-access-list)#seq 10 permit tcp any any class tcp
[local]Redback(config-ipv6-access-list)#!
[local]Redback(config-ipv6-access-list)#http-redirect profile h_redirect
!
[local]Redback(config-hr-profile)#url "http://2.2.2.2:8888"
[local]Redback(config-hr-profile)#ipv6 url "http://[2000:1:2::1]:80"
[local]Redback(config-hr-profile)# message "to be redirected to this address"
[local]Redback(config-hr-profile)#!
[local]Redback(config-hr-profile)#subscriber default
[local]Redback(config-sub)#ip address pool
[local]Redback(config-sub)#http-redirect profile h_redirect
[local]Redback(config-sub)#forward policy red_pol in
[local]Redback(config-sub)#access-list count ip ipv6 ipv6-policy policy
[local]Redback(config-sub)#!
[local]Redback(config-sub)#! ** End Context **
[local]Redback(config-sub)#!
[local]Redback(config-sub)#http-redirect server
[local]Redback(config-hr-server)#port 80 8888
[local]Redback(config-hr-server)#!
[local]Redback(config-hr-server)#forward policy red_pol
!
[local]Redback(config-policy-frwd)#ip access-group find_tcp isp
[local]Redback(config-policy-group)#ipv6 access-group find_tcp isp
[local]Redback(config-policy-group)#class tcp
[local]Redback(config-policy-group-class)#redirect destination local
[local]Redback(config-policy-group-class)#
[local]Redback(config-policy-group-class)#
[local]Redback(config-policy-group-class)#end
!
```

3.11 Configure and Apply IPv4 and IPv6 ACLs to Process and Classify all Traffic with the Same QoS Policy

The following example configuration shows how IPv4 and IPv6 ACLs can be applied and used to allow processing and classification of both traffic types for the same QoS policy. Web traffic that conforms to the traffic rate of 5000 kbps is marked with a Differentiated Services Code Point (DSCP) value of AF11. Web traffic exceeding that rate is dropped by default. Packets classified as VOIP are prioritized over both web and default traffic through the DSCP setting of `ef`, or expedited forwarding. Packets classified as default are set to the DSCP value of `df`, or default forwarding.

```
[local]Redback(config)#context local
[local]Redback(config-ctx)#policy access-list ipv4_access
[local]Redback(config-access-list)#seq 10 permit tcp any any eq www class web
[local]Redback(config-access-list)#seq 20 permit tcp any any class data
[local]Redback(config-access-list)#seq 30 permit udp any any eq 1000 class voip
[local]Redback(config-access-list)#ipv6 policy access-list ipv6_access
[local]Redback(config-ipv6-access-list)#seq 10 permit tcp any any eq www class web
[local]Redback(config-ipv6-access-list)#seq 20 permit tcp any any class data
[local]Redback(config-ipv6-access-list)#seq 30 permit udp any any eq 1000 class voip
[local]Redback(config-ipv6-access-list)#qos policy policing Police
[local]Redback(config-policy-policing)#ip access-group ipv4_access local
[local]Redback(config-policy-group)#ipv6 access-group ipv6_access local
[local]Redback(config-policy-group)#class web
[local]Redback(config-policy-group-class)#rate 5000 burst 7000
[local]Redback(config-policy-class-rate)#class voip
[local]Redback(config-policy-group-class)#mark dscp ef
[local]Redback(config-policy-group-class)#class data
[local]Redback(config-policy-group-class)#mark dscp df
[local]Redback(config-policy-group-class)#exit
[local]Redback(config-policy-group)#exit
[local]Redback(config)#port ethernet 4/1
[local]Redback(config-port)#qos policy policing Police
[local]Redback(config-port)#commit
```

3.12 Configure a Policy ACL Associated with a NAT Policy

The following example creates a policy ACL and applies it to a NAT policy with dynamic translations in which all packets except those classified as **CLASS3** are ignored (that is, the NAT policy is not applied to them). All source IP addresses for incoming packets classified as **CLASS3** are translated using IP addresses from the **pool_dyn** pool:

```
!Create the NAT pool
[local]Redback(config-ctx)#ip nat pool pool_dyn
[local]Redback(config-nat-pool)#address 11.11.11.0/24
[local]Redback(config-nat-pool)#exit
!Create the policy ACL
[local]Redback(config-ctx)#policy access-list NAT-ACL
[local]Redback(config-access-list)#seq 10 permit ip 10.10.10.0 0.0.0.255 class CLASS3
[local]Redback(config-access-list)#exit
!Create the NAT policy and apply the policy ACL
[local]Redback(config-ctx)#nat policy pol1
[local]Redback(config-nat-pool)#ignore
[local]Redback(config-nat-pool)#access-group NAT-ACL
[local]Redback(config-policy-group)#class CLASS3
[local]Redback(config-policy-group-class)#pool pool_dyn local
```

3.13 Configure an ACL to Enable SNMP Traps

Filter ACLs contain a set of rules, executed in specific order. Each rule defines certain criteria, such as checking for IP packets, protocol, source/destination IP addresses, and ports. Each rule also contains an *action*, either *permit* or *deny*. If the packet matches the criteria declared in the given rule, the action is executed; that is, the packet is either permitted or denied, and the processing is stopped without executing the rules that follow. If the packet does not match the criteria declared in the given rule, the processing continues on to the next rule. Table 2 describes the key concepts in the IP ACL statements:

Table 2 ACL Fields and Values

ACL Fields	Description
Sequence Number	An appropriate sequence number that positions the rule in the proper processing order in the list of rules.
Action	Permit or deny
Protocol	UDP

Table 2 ACL Fields and Values

ACL Fields	Description
Source IP	— Source IP Address: 0.0.0.0 — Wildcard: 255.255.255.255 Specifies that the source can be any IP address
Destination Port	snmptrap
Condition	eq

To configure an ACL that allows you to block all incoming UDP traffic and continue to forward SNMP traps, enter the command in context configuration mode. In the following example, for every statement with sequence numbers, 10, 20, and 30, a different ACL action is initiated. Sequence 10 *permits* UDP traffic with destination address 127.0.0.1 and the **snmptrap** destination port. In other words, it allows the packets matching these criteria, to be processed by the system. Sequence 20 *denies* UDP traffic data collection, thereby drops UDP packets that do not meet the criteria declared in the previous rule. Sequence 30 *permits* collection of any IP packets that do not match the previous rules. The *any any* construct at the end of every IP ACL refers to any source or any destination IP address:

```
[local]Redback(config-ctx)#ip access-list block-all-udp-except-internal-traps
[local]Redback(config-access-list)#seq 10 permit udp any host 127.0.0.1 eq snmptrap
[local]Redback(config-access-list)#seq 20 deny udp any any
[local]Redback(config-access-list)#seq 30 permit ip any any
```

Administrative ACLs are used to protect the SmartEdge OS from Denial of Service (DoS) and other attacks. As opposed to ACLs that are applied to IP traffic sent or received on an interface, administrative ACLs are applied to IP traffic that is received locally and processed within a SmartEdge context. For example, an administrative ACL may be used to restrict access to a limited set of applications originating from a single source subnet. To configure an administrative ACL, issue the following command:

```
[local]Redback(config-ctx)#admin-access-group block-all-udp-except-internal-traps in
```