

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 17, 2013

L. Huang
A. Clemm
Cisco Systems
A. Bierman
YumaWorks
October 14, 2012

YANG Data Model for Access Control List Configuration
draft-huang-netmod-acl-01.txt

Abstract

This document defines a YANG data model for the configuration of Access Control Lists (ACLs) on a device.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Internet-Draft

yang-acl

October 2012

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Internet-Draft

yang-acl

October 2012

Table of Contents

1.	Introduction	4
2.	Definitions and Acronyms	4
3.	The Design of ACL Data Model	5
4.	acl Module	9
4.1.	Features	9
4.2.	Types	10
4.3.	Groupings	11
4.4.	Containers	12
4.4.1.	acls Container	12
4.4.2.	port-groups Container	12
4.4.3.	timerange-groups Container	13
4.4.4.	ip-address-groups Container	14
5.	acl-ip module	14
5.1.	Groupings	14
5.1.1.	IP-SOURCE-NETWORK grouping	15
5.1.2.	IP-DESTINATION-NETWORK grouping	16
5.1.3.	DSCP-OR-TOS Grouping	16
5.1.4.	IP-ACE-FILTERS Grouping	17
5.2.	augment	19
5.2.1.	global-fragments leaf	19
6.	acl-mac module	22
6.1.	MAC-SOURCE-NETWORK grouping	22
6.2.	MAC-DESTINATION-NETWORK grouping	23
6.3.	augment	23
7.	acl-arp module	23
7.1.	augment	23
8.	Data Model Structure	24
9.	ACL Examples	32
10.	ACL YANG Module	34
11.	ACL-IP YANG Module	47
12.	ACL-MAC Configuration YANG Module	61
13.	ACL-ARP Configuration YANG Module	67
14.	COMMON-TYPES YANG Module	70
15.	Security Considerations	78

16.	Open Issues	78
17.	Acknowledgements	78
18.	Normative References	78

[1.](#) Introduction

This document defines a YANG [[RFC6020](#)] data model for the configuration of Access Control Lists (ACLs).

An ACL is an ordered set of rules that is used to filter traffic on a networking device, i.e. to define "firewall rules". Each rule is represented by an Access Control Entry (ACE). An ACE consists of two parts:

Filters with a set of matching criteria that a packet must satisfy for the rule to be applied.

Actions that specifies what to do with the packet when the matching criteria is met, for example, to drop the packet.

There are different types of ACL: MAC ACL, IP ACL, and ARP ACL.

MAC ACLs - MAC ACLs are used to filter traffic using the information in the Layer 2 header of each packet. MAC ACLs are by default only applied to non-IP traffic; however, Layer 2 interfaces can be configured to apply MAC ACLs to all traffic.

IP ACLs: IP ACLs are ordered sets of rules that can use to filter traffic based on IP information in the Layer 3 header of packets. The device applies IP ACLs only to IP traffic. IP ACL can be IPv4 or IPv6.

ARP ACLs - The device applies ARP ACLs to IP traffic.

Not every device implements every type of ACL. In addition, device implementations may vary greatly in terms of the filter constructs that they support. Therefore, acl YANG Module makes extensive use of the "feature" construct which allows implementations to support those ACL configuration features that lie within their capabilities.

How ACLs are applied in device configuration to interfaces and other components is outside the scope of this model.

[2.](#) Definitions and Acronyms

ACE: Access Control Entry

ACL: Access Control List

AFI: Address Field Identifier

ARP: Address Resolution Protocol

CoS: Class of Service

DSCP: Differentiated Services Code Point

ICMP: Internet Control Message Protocol

IGMP: Internet Group Management Protocol

IP: Internet Protocol

IPv4: Internet Protocol version 4

IPv6: Internet Protocol version 6

MAC: Media Access Control

QoS: Quality of Service

TCP: Transmission Control Protocol

ToS: Type of Service

TTL: Time To Live

UDP: User Datagram Protocol

VLAN: Virtual Local Area Network

VRF: Virtual Routing and Forwarding

[3.](#) The Design of ACL Data Model

The ACL data model consists of five YANG modules. The first module, "acl", defines generic ACL aspects which are common to all ACLs regardless of their type, as well as a set of auxiliary definitions. In effect, the module can be viewed as providing a generic ACL "superclass".

Three other modules, "acl-ip", "acl-mac", and "acl-arp" , augment the "acl" module with definitions that are specific to different types of ACLs, specifically, ACLs for IP, MAC, and ARP, respectively. These specifics are for the largest part reflected in the Access Control Entries, that is, the rules which specify the filter criteria that a packet must meet for the rule to be applied, and the actions that are to be taken in case the filter matches. Keeping the modules separate provides for a more modular data model than would be the case if all types were combined into a single monolithic module.

Finally, module "common-types" defines types that are used in the ACL data model but are not really specific to ACLs. These definitions could potentially be of interest to other models as well; keeping them in a separate module allows to import these definitions independent of the support for ACLs.

The data hierarchy that is defined by the acl module is depicted in the following Figure 1, where brackets enclose list keys, "rw" means configuration, "ro" means operational state data, and "?" means optional node. Parentheses enclose choice and case nodes. The structure is a collapsed structure and does not depict all definitions; it is intended to illustrate the overall structure. A fully expanded structure can be found in Data Model Structure Section ([Section 8](#)).

```

module: acl
  +--rw acls
    +--rw acl [name]
      |   +--rw name
      |   +--rw acl-type
      |   +--rw enable-capture-global?
      |   +--rw capture-session-id-global?
      |   +--rw (enable-match-counter-choices)?
      |   +--ro match?
      |
    +--rw port-groups
      |   +--rw port-group [name]
      |       +--rw name
      |       +--rw port-group-entry
    +--rw timerange-groups
      |   +--rw timerange-group [name]
      |       +--rw name
      |       +--rw time-range
    +--rw ip-address-groups
      |   +--rw ip-address-group [name]
      |       +--rw name
      |       +--rw afi?
      |       +--rw ip-address

```

Figure 1

Data nodes in the `acl` module are contained under a single container node, "acls". This node contains a list, "acl". Each ACL is represented by an element in that list and identified by a name that serves as key to the list. Interfaces (which are not part of the model) to which an ACL is applied can then refer to the ACL using that name. Each `acl` list element has furthermore a type, as

indicated through "acl-type". The `acl-type` determines which types of ACEs can be contained in an ACL. The ACE definitions themselves are provided by the `acl-ip`, `acl-mac`, and `acl-arp` modules, which augment the `acl` definition in the `acl` module accordingly. The subsequent data nodes in the `acl` list allow to configure whether packets that match an ACL should be captured for further analysis. Finally, the list contains an object that maintains a counter of the number of ACL matches.

Auxiliar objects "port-groups", "ip-address-groups", "timerange-groups" are used to define groupings of ports and of IP-addresses as well as schedule information, respectively. They are in effect convenience objects which allow ACEs to refer to groupings and schedules by name, rather than needing to re-specify them in each ACE where they apply.

The following figure depicts how different types of ACEs are inserted into that structure. As indicated earlier, the corresponding definitions are provided in separate modules that augment the `acl` module. In the data structure, the augmenting module is indicated by the prefix of the corresponding data nodes: "acl-ip", "acl-mac", and "acl-arp", respectively. ACEs for IPv4 and for IPv6 are both defined in the same module, `acl-ip`. While it would have been possible to define each in its own separate module, it was a design decision to combine them, as they share enough commonality that a separation would have resulted in a considerable amount of definition redundancy.

The figure does not depict objects not pertinent to that structure, such as objects intended to make the definition of port groups ("port-groups"), timeranges ("time-range-groups"), and IP address groups ("ip-address-groups") reusable, as well as objects that are contained in `acl` list elements, such as "name" and "enable-capture-global".

```

module: acl
  +--rw acls
    +--rw acl [name]
      |   +--rw acl-ip:afi
      |   +--rw acl-ip:ipv6-aces
      |   |   +--rw acl-ip:ipv6-ace [name]
      |   |   +--rw acl-ip:name
      |   |   +--rw (remark-or-ipv6-case)?
      |   |   |   +--:(remark)
      |   |   |   |   +--rw acl-ip:remark
      |   |   |   +--:(ipv6-ace)
      |   |   |   |   +--rw acl-ip:filters
      |   |   |   |   +-- filter parameters

```

```

      |   |   |   |   +--rw acl-ip:actions

```



```
| |      +--:(arp-ace)
| |      |   +--rw acl-arp:filters
| |      |       +-- filter parameters
| |      |   +--rw acl-arp:actions
| |      |       +-- action parameters
| |      +-- ro acl-arp:match
```

Figure 2

As is evident from Figure 2, the same generic design pattern is reflected in every ACL type. Each ACL contains a list of ACEs, identified by a name by which ACEs in the list are ordered. Each ACE consists either of a remark or of an actual access control rule. Remarks are in effect comment lines inside an ACL that are intended for human or administrator consumption. They are included in the YANG module to maintain consistency with CLI. Access control rules, on the other hand, consist of a left hand side ("filters") that specifies a set of matching criteria and a right hand side ("actions") that specifies the action to take when matching criteria are met. An overview of the full list of filter and parameters is given in [Section 8](#).

Since the design pattern for each ACL type is the same, an alternative design to the YANG modules would have been to extend the "acl" module to include the data nodes up to the level depicted in Figure 2, as the real distinction occurs in the filter and action parameters that occur below it. In that case, however, the corresponding data nodes would have had to contend with more complex conditions. The modules defined here aim at keeping complexity of definitions within the modules as low as possible, at the price of repeating a few data nodes that provide the overall top level structure.

[4.](#) acl Module

Module "acl" is a top container module for all ACLs. It contains a container "acls" with a list "acl" of named ACLs. Modules "acl-ip", "acl-mac", and "acl-arp" augment this list with the objects that are specific to each respective type of ACL. In addition, module "acl" also defines a set of features, reusable types, and reusable groupings.

[4.1.](#) Features

When it comes to ACL implementations, a wide range of different capabilities exists across devices. For example, not every device

implements every type of ACL. Some devices may support time-based ACLs that are only in effect during specified times, others may not.

In order to accommodate this wide range of capabilities, this data model makes extensive use of the "feature" construct. The defined features allow implementations to declare which capabilities they support, and only support the corresponding portions of the data model.

[4.2.](#) Types

The definition of ACLs requires a number of new data types introduced in this data model. Table 1 depicts data types that are unique to ACLs. Table 2 depicts data types that are required by ACLs, but not specific to them, and that may hence be reused by other models. Those data types are defined in module "common-types". For details of each type, please refer to the corresponding typedef descriptions and references in the model.

Internet-Draft

yang-acl

October 2012

YANG type	base type
acl-comparator	enumeration
acl-action	enumeration
acl-remark	string
acl-type-ref	identityref
acl-ref	leafref
port-group-ref	leafref
ip-address-group-ref	leafref
time-range-Ref	leafref
weekdays	bits
acl-name-string	string

Table 1

YANG type	base type
cos	uint8
tos	uint8
precedence	uint8
tcp-flag-type	enumeration
ether-type	string
ip-protocol	uint8
igmp-code	uint8
icmp-type	uint32
icmp-code	uint32
vlan-identifier	uint16
time-to-live	uint32

Table 2

[4.3.](#) Groupings

The data model defines two groupings, ACE-COMMON and FILTER-COMMON.

- o ACE-COMMON is a collection of nodes that should be added to every ACE list entry. ACE-COMMON contains the actions container and a read-only match leaf. The actions container contains two leaves.
 - * An "action" leaf that specifies what to do with the packet when the matching criteria is met, for example, to drop the packet.
 - * A "log" leaf that indicates whether to create a log entry when an ace filter matches. (Some devices may not support a log

Huang, et al.

Expires April 17, 2013

[Page 11]

Internet-Draft

yang-acl

October 2012

capability. Hence support of this leaf is conditional on declaration of a corresponding feature, as indicated by use of the "if-feature" construct.)

- o FILTER-COMMON is a collection of nodes that should be added to every 'filters' container within each ACE list entry.

[4.4.](#) Containers

[4.4.1.](#) acls Container

Container "acls" contains a list "acl" of named ACLs. Each list element "acl" contains the following global leaves. The list elements are augmented with additional data nodes defined in modules "acl-arp", "acl-mac", and "acl-ip".

- o name
- o acl-type
- o enable-capture-global
- o capture-session-id-global
- o enable-match-counter-choices: The difference of these two choices

is that "enable-match-counter" indicates to collect total match statistics for all aces, whereas "enable-per-entry-match-counter" indicates to collect match statistics for each ACE.

- o match

[4.4.2.](#) port-groups Container

Container "port-groups" allows to classifying protocol port into groups. It contains a sequence of "port-group" data nodes. Each "port-group" defines a range of ports and can be referred to by name. Multiple ACEs can refer to the same port group. The following is a Netconf XML example of port-groups and how it is referred to from an ACE.

```
<src-port-group-name>
<port-group-name>port-tunnel1</port-group>
</src-port-group-name>

<port-groups>
  <port-group>
    <name>port-tunnel1</name>
    <port-group-entry>
      <name>http-proxy</name>
      <port-lower>21</port-lower>
      <port-upper> 22</port-upper>
    </port-group-entry>
  </port-group>
</port-groups>
```

[4.4.3.](#) timerange-groups Container

Container "timerange-groups" container contains a list, "timerange-group". Each of its elements defines a sequence of time ranges, "time-range". Each time-range object consists of either a remark (comments for the time range), or of an absolute time for start or end (or both) of the time range, or a periodic time for start or end or both. Object "remark" contains administrator-provided comments for the time-range that will be kept in the device. Like with port groups, the same time-range can be reused by different ACEs. The following is a Netconf XML example of a timerange group that contains a remark and a single time range.

```
<timerange-groups>
  <timerange-group>
    <name>weekday</name>
    <time-range>
      <name>10</name>
      <remark> email server maintenance</remark>
    </time-range>
    <time-range>
      <name>20</name>
      <periodic>
        <weekday>
          Monday Tuesday Wednesday Thursday Friday
        </weekday>
        <start> 21:00:00</start>
        <end> 24:00:00</end>
      </periodic>
    </time-range>
  </timerange-group>
</timerange-groups>
```

[4.4.4.](#) ip-address-groups Container

Container "ip-address-groups" contains is list "ip-address-group" of named IP address groups. Each IP address group is a sequence of pairs "ip-address" and "mask", or a pair of "host" and "host-address". Each IP address group can be referred from an ACE by name. The following is a Netconf XML example of an IP address group and how it is referred to from an ACE.

```
<ip-address-groups>
```

```

<ip-address-group>
  <name>Email-Server-IPV4</name>
  <ip-addresses>
    <ip-address>
      <name>10</name>
      <ip-address>128.107.0,0</ip-address>
      <ip-mask>255.255.0.0</ip-mask>
    </ip-address>
    <ip-address>
      <name>20</name>
      <ip-address>139.207.0.0</ip-address>
      <ip-mask>255.255.0.0</ip-mask>
    </ip-address>
  </ip-addresses>
</ip-address-group>
</ip-address-groups>

<ip-ace>
  <name>100</name>
  <afi>ipv4</afi>
  <actions>permit</actions>
  <filters>
    <ip-source-group>Email-Server-IPV4</ip-source-group>
    <ip-dest-any/>
  </filters>
</ip-ace>

```

[5.](#) acl-ip module

acl-ip is the module that defines IP-ACL. It augments acl list in acl module.

[5.1.](#) Groupings

[5.1.1.](#) IP-SOURCE-NETWORK grouping

```

IP-SOURCE-NETWORK
  +--rw (source-address-host-group)?

```



```

+--:(source-ip)
|   +--rw ip-source-address      inet:ip-address
|   +--rw ip-source-mask         inet:ip-address
+--:(ip-source-any)
|   +--rw ip-source-any          empty
+--:(source-host)
|   +--:(ip-src-host-address-or-name)
|       +--:(ip-source-host-address)
|           +--rw ip-source-host-address      inet:ip-address
|       +--:(ip-source-host-name)
|           +--rw ip-source-host-name          inet:domain-name
+--:(source-group)
|   +--rw ip-source-group?          ip-address-group-ref

```

IP-SOURCE-NETWORK is a reusable grouping. It allows five ways to specify a network: ip with mask, any network, host-name or host address, reference to a predefined ip address group. Here are valid example instances:

- o ip with mask:

```

<ip-source-address>192.168.1.0</ip-source-address>
<ip-source-mask>255.255.255.0</ip-source-mask>

```

- o any network:

```

<ip-source-any/>

```

- o host-name:

```

<ip-source-host-name>switch1</ip-source-host-name>

```

- o host-address:

```

<ip-source-host-address>192.168.1.2</ip-source-host-address>

```

- o reference to a predefined ip address group (Email-Server-IPV4 is defined in [Section 4.4.4](#)):

```
<ip-source-group>Email-Server-IPV4</ip-source-group>
```

[5.1.2.](#) IP-DESTINATION-NETWORK grouping

```
IP-DESTINATION-NETWORK
  +--rw (dest-address-host-group)?
    +--:(dest-ip)
      |   +--rw ip-dest-address      inet:ip-address
      |   +--rw ip-dest-mask?        inet:ip-address
    +--:(ip-dest-any)
      |   +--rw ip-dest-any          empty
    +--:(dest-host)
      |   +--:(ip-dest-host-address-or-name)
      |     +--:(ip-dest-host-address)
      |       +--rw ip-dest-host-address      inet:ip-address
      |     +--:(ip-dest-host-name)
      |       +--rw ip-dest-host-name        inet:domain-name
    +--:(group)
      +--rw ip-dest-group?                ip-address-group-ref
```

IP-DESTINATION-ADDRESS is a reusable grouping. Its structure is similar to IP-SOURCE-NETWORK. The reason to have both IP-SOURCE-NETWORK and IP-DESTINATION-NETWORK groupings is to allow "ip-source-address" and "ip-destination-address" leaves to appear in the same container. For example:

```
<filters>
  <ip-source-address>192.168.1.0</ip-source-address>
  <ip-source-mask>255.255.255.0</ip-source-mask>
  <ip-dest-address>any</ip-dest-address>
</filters>
```

[5.1.3.](#) DSCP-OR-TOS Grouping

DSCP-OR-TOS grouping defines a choice, "dscp-or-tos". It allows two ways to filter for a QoS packet:

- o dscp: Match packet on DSCP value.
- o tos: Match packet on TOS and precedence value.

The typedef for "tos" and "precedence" is defined in module "common-types", which could be deprecated should IETF define a separate set of definitions.

Internet-Draft

yang-acl

October 2012

[5.1.4.](#) IP-ACE-FILTERS Grouping

IP-ACE-FILTERS

```

+--rw protocol?                c-types:ip-protocol
+--acl:FILTER-COMMON
+--rw fragments?              empty
+--rw time-range?             acl:Time-Range-Ref
+-- (src-ports)?
|   +--rw (port-number-or-range)?
|   |   +--:(port-number-range)
|   |   |   +--rw src-port-lower?      inet:port-number
|   |   |   +--rw src-port-upper?     inet:port-number
|   |   +--:(port-number)
|   |   |   +--rw src-comparator        comparator
|   |   |   +--rw src-port?            inet:port-number
|   +-- : (port-group-ref)
|   |   +--src-port-group-name
+-- (des-ports)?
|   +--rw (port-number-or-range)?
|   |   +--:(port-number-range)
|   |   |   +--rw des-port-lower?      inet:port-number
|   |   |   +--rw des-port-upper?     inet:port-number
|   |   +--:(port-number)
|   |   |   +--rw des-comparator        comparator
|   |   |   +--rw des-port?            inet:port-number
|   +-- : (by-name)
|   |   +-- des-port-group-name
+--rw icmp-type?              c-types:icmp-type
+--rw icmp-code?              c-types:icmp-type
+--rw (packet-length-or-range)?
|   +--:(length)
|   |   +--rw packet-length-comparator  acl:Comparator
|   |   +--rw packet-length            uint32
|   +--:(range)
|   |   +--rw packet-length-upper       uint32
|   |   +--rw packet-length-lower       uint32
+--rw tcp-flag-value?         c-types:tcp-flag-type
+--rw tcp-flag-mask?          c-types:tcp-flag-type
+--rw tcp-flag-operation?     enumeration

```

<pre> +--rw (ttl-value-or-range)? +--:(value) +--rw ttl-comparator? +--rw ttl-value? +--:(range) +--rw ttl-value-lower? +--rw :ttl-value--upper? </pre>	<pre> acl:acl-comparator c-types:Time-to-Live c-types:Time-to-Live c-types:Time-to-Live </pre>
---	---

IP-ACE-FILTERS defines the following leaves that are used by both by IPv4 and IPv6 ACEs:

- o protocol
- o acl:FILTER-COMMON: see [Section 4.3](#)
- o fragments: When present, it matches the non-initial fragment.
- o time-range: Enable packet capture on this filter for a timerange-group by name. time-range is Time-Range-Ref type which is a leafref.
- o src-ports choice: Allows the following three ways to define a group of ports.
 - * port-number-range: Use "src-port-lower" and "src-port-upper" leaves to specify a port range. The value of "src-port-lower" has to be less than or equal the value of "src-port-upper".
 - * port-number: Use "comparator" and "src-port" leaves to specify a port range. See Comparator typedef in the model for the possible values the "comparator" leaf.
 - * port range ref: Refer to a named port group that is defined using port-groups. For example:

```
<port-group-name>port-tunnel1</port-group-name>
```

- o dest-ports choice: Analogous to "src-ports".
- o packet-length-or-range: Allows two ways to specify packet length

range.

- * case length: Use comparator and a single packet-length to specify the range.
- * case range: Use packet-length-lower and packet-length-upper to specify a range. The value of packet-length-lower must be lower than or equal to the value of packet-length-upper.

- o icmp-type
- o icmp-code
- o packet-length-or-range choice

- o tcp-flag-value: tcp-flag-value, tcp-flag-mask and tcp-flag-operation allow to match any combination of packet tcp flag values.

The following example is to match the packet
tcp flag ack=1, syn=1, and fin=0;

```
<tcp-flag-value> ack syn <tcp-flag-value>  
<tcp-flag-mask>ack syn fin</tcp-flag-mask>  
<tcp-flag-operation>match-all</tcp-flag-operation>
```

- o tcp-flag-mask
- o tcp-flag-operation
- o ttl-value-or-range

[5.2.](#) augment

The module "acl-ip" augments the definition of data node "/acl:acls/acl:acl" with additional leaves and subcomponents.

- o afi
- o ipv6-aces: It contains a list of ipv6-ace. Each ipv6-ace is

either a remark or a real access control filters. The case ipv6-ace defines the filters and actions for ipv6-ace. The ace uses filters defined in grouping IP-SOURCE-NETWORK, IP-DESTINATION-NETWORK, IP-ACE-FILTERS, DSCP-OR-TOS. In addition, it also allows filter on igmp-type and flow-label,

- o ipv4-aces: ipv4-ace has similar structure to ipv6-aces.
- o global-fragments

[5.2.1.](#) global-fragments leaf

global-fragments is an optional leaf. It has an enumeration value of not-set, permit-all, deny-all. not-set is the default value. When the global-fragments is permit-all or deny-all, it is to permit or deny the implicit ace fragment filter. Here is an example of implicit ace and how the implicit ace is affected when global-fragments is set.

Example 1: The acl configuration from the management interface with global-fragments is absent.

YANG instance of this cli configuration:

```
<acls>
  <acl>
    <name>fragment_test1</name>
    <afi>ipv4</afi>
    <acl-type>ip-acl</acl-type>
    <ip-aces>
      <name>10</name>
      <actions>
        <action>permit</action>
      </actions>
      <filters>
        <ip-source-address>192.168.5.0</ip-source-address>
        <ip-source-mask>255.255.255.0</ip-source-mask>
        <ip-dest-address>any</ip-dest-address>
      </filters>
    </ip-aces>
    <ip-aces>
      <name>20</name>
```

```

    <actions>
      <action>permit</action>
    </actions>
    <filters>
      <ip-source-address>189.168.0.0</ip-source-address>
      <ip-source-mask>255.255.0.0</ip-source-mask>
      <ip-dest-address>any</ip-dest-address>
      <fragments/>
    </filters>
  </ip-aces>
</acl>
</acls>

```

By taking all the tags out, the above yang can be express in a summary of cli format like the following:

```

fragment_test1 ip-acl ipv4
10 permit ip 192.168.5.0 255.255.255.0 any
20 permit ip 189.168.0.0 255.255.0.0 any fragment.

```

The acl configuration together with implicit ace in the device will be:

```

fragment_test1 ip-acl ipv4
10 permit ip 192.168.5.0 255.255.255.0 any
11 permit ip 192.168.5.0 255.255.255.0 any fragment
20 permit ip 189.168.0.0 255.255.0.0 any fragment.
100 deny any any
110 deny any any fragment

```

Notice three lines of configuration. 11, 100 and 110, are implicit.

Example 2: The acl configuration from the management interface with global-fragments

```

<acls>
  <acl>
    <name>fragment_test2</name>
    <acl-type>ip-acl</acl-type>
    <global-fragments>deny-all</global-fragments>
    <afi>ipv4</afi>
    <ip-aces>
      <name>10</name>
      <actions>
        <action>permit</action>
      </actions>
      <filters>
        <ip-source-address>192.168.5.0</ip-source-address>
        <ip-source-mask>255.255.255.0</ip-source-mask>
        <ip-dest-address>any</ip-dest-address>
      </filters>
    </ip-aces>
    <ip-aces>
      <name>20</name>
      <actions>
        <action>permit</action>
      </actions>
      <filters>
        <ip-source-address>189.168.0.0</ip-source-address>
        <ip-source-mask>255.255.0.0</ip-source-mask>
        <ip-dest-address>any</ip-dest-address>
        <fragments/>
      </filters>
    </ip-aces>
  </acl>
</acls>

```

The acl configuration in the device with implicit aces. The deny-all void "11 permit ip 1.1.1.1/16 any fragment" ace in previous example.

By taking all the tags out, the above yang can be express in a summary of cli format like the following:

```
fragment_test2 ip-acl ipv4 deny-all
```



```
10 permit ip 192.168.5.0 255.255.255.0 any
20 permit ip 189.168.0.0 255.255.0.0 any fragment.
```

The acl configuration together with implicit ace in the device will be:

```
fragment_test2 ip-acl ipv4
10 permit ip 192.168.5.0 255.255.255.0 any
20 permit ip 189.168.0.0 255.255.0.0 any fragment.
100 deny any any
110 deny any any fragment
```

[6.](#) acl-mac module

[6.1.](#) MAC-SOURCE-NETWORK grouping

```
MAC-SOURCE-NETWORK
+--rw (source-network)?
+--:(source-mac)
|   +--rw source-address          yang:mac-address
|   +--rw source-address-mask     yang:mac-address
+--:(source-any)
|   +--rw source-any              empty
+--:(source-host)
|   +--rw acl-mac:source-host-name inet:host
```

MAC-SOURCE-ADDRESS is a reusable grouping. It allows to express the three kinds network.

any network: use source-any to express any network.

```
<mac-source-kind>any</mac-source-kind>
```

single host network.

```
<source-host-name>my-host</source-host-name>
```

host address with a mask.

```
<source-address>0180.c200.000</source-address>
<source-address-mask>0000.0000.0000</source-address-mask>
```

[6.2.](#) MAC-DESTINATION-NETWORK grouping

```
MAC-DESTINATION-NETWORK
  +--rw (dest-network)?
    +--:(address)
      |   +--rw dest-address          yang:mac-address
      |   +--rw dest-address-mask     yang:mac-address
    +--:(dest-any)
      |   +--rw dest-any              empty
    +--:(host)
      +--rw acl-mac:dest-host-name    inet:host
```

MAC-DESTINATION-ADDRESS is a reusable grouping similar to MAC-SOURCE-ADDRESS. The reason to have both MAC-SOURCE-ADDRESS and MAC-DESTINATION-ADDRESS grouping is to allow source-address and destination-address leaves appear in the same container. For example:

```
<filters>
  <source-address>0180.c200.000</source-address>
  <source-address-mask>0000.0000.0000</source-address-mask>
  <dest-any/>
</filters>
```

[6.3.](#) augment

The module "acl-mac" augments the definition of data node "/acl:acls/acl:acl" with additional leaves and subcomponents. acl-mac has similar structure as acl-ipv4 and acl-ipv6 except the filters are different. mac-ace has filters defined in grouping MAC-SOURCE-NETWORK, MAC-DESTINATION-NETWORK, acl:FILTER-COMMON, ethertype-mask, cos, time-range, and vlan.

[7.](#) acl-arp module

[7.1.](#) augment

The module "acl-arp" augments the definition of data node "/acl:acls/acl:acl" with additional leaves and subcomponents.

Internet-Draft

yang-acl

October 2012

```

augment "/acl:acls/acl:acl"
  +--rw acl-arp:arp-aces
    +--rw acl-arp:arp-ace [name]
      +--rw acl-arp:name          acl:acl-name-string
      +--rw (remark-or-arp-ace)?
        +--:(remark)
          | +--rw acl-arp:remark?    acl:acl-remark
        +--:(arp-ace)
          +--rw filters
            | +--rw direction?          enumeration
            | +--acl-ip:IP-SOURCE-NETWORK
            | +--acl-ip:IP-DESTINATION-NETWORK
            | +--acl-mac:MAC-SOURCE-NETWORK
            | +--acl-mac:MAC-DESTINATION-NETWORK
            | +--acl:FILTER-COMMON
          +acl:ACE-COMMON

```

8. Data Model Structure

The combined data model for ACL configuration is structured as follows. "acl" defines the generic components of an acl system. "acl-ip", "acl-mac", "acl-arp" augment the "acl" module with additional data nodes that are needed for ip, mac, and arp acl respectively.

```

module: acl
  +--rw acls
    +--rw acl [name]
      | +--rw name
      | +--rw acl-type
      | +--rw enable-capture-global?
      | +--rw capture-session-id-global?
      | +--rw (enable-match-counter-choices)?
      | | +--:(match)
      | | | +--rw enable-match-counter?
      | | +--:(per-entry-match)
      | | | +--rw enable-per-entry-match-counter?
      | +--ro match?
      | +--rw acl-ip:afi?
      | +--rw acl-ip:ipv6-aces
      | | +--rw acl-ip:ipv6-ace [name]

```

```

| | +--rw acl-ip:name          acl:acl-name-string
| | +--rw (remark-or-ipv6-case)?
| |   +--:(remark)
| |     | +--rw acl-ip:remark?    acl:acl-remark
| |     +--:(ipv6-ace)
| |       +--rw acl-ip:filters

```

```

| | | | +--rw (source-address-host-group)
| | | | | +--:(source-ip)
| | | | | | +--rw acl-ip:ip-source-address
| | | | | | +--rw acl-ip:ip-source-mask
| | | | | +--:(ip-source-any)
| | | | | | +--rw acl-ip:ip-source-any?
| | | | | +--:(source-host)
| | | | | | +--rw (ip-src-address-or-name)
| | | | | | | +--:(ip-source-host-address)
| | | | | | | | +--rw acl-ip:ip-source-host-address?
| | | | | | | | +--:(ip-source-host-name)
| | | | | | | | +--rw acl-ip:ip-source-host-name?
| | | | | +--:(source-group)
| | | | | | +--rw acl-ip:ip-source-group?
| | | | +--rw (dest-address-host-group)
| | | | | +--:(dest-ip)
| | | | | | +--rw acl-ip:ip-dest-address
| | | | | | +--rw acl-ip:ip-dest-mask
| | | | | +--:(ip-dest-any)
| | | | | | +--rw acl-ip:ip-dest-any?
| | | | | +--:(dest-host)
| | | | | | +--rw (ip-dest-address-or-name)
| | | | | | | +--:(ip-dest-host-address)
| | | | | | | | +--rw acl-ip:ip-dest-host-address?
| | | | | | | | +--:(ip-dest-host-name)
| | | | | | | | +--rw acl-ip:ip-dest-host-name?
| | | | | +--:(dest-group)
| | | | | | +--rw acl-ip:ip-dest-group?
| | | | +--rw acl-ip:protocol?
| | | | +--rw acl-ip:enable-capture?
| | | | +--rw acl-ip:capture-session-id?
| | | | +--rw acl-ip:fragments?
| | | | +--rw acl-ip:time-range?
| | | | +--rw (src-ports)?
| | | | | +--:(port-number-range)

```

```
| | | | | +---rw acl-ip:src-port-lower
| | | | | +---rw acl-ip:src-port-upper
| | | | | +---:(port-number)
| | | | | +---rw acl-ip:src-comparator
| | | | | +---rw acl-ip:src-port
| | | | | +---:(port-group-ref)
| | | | | +---rw acl-ip:src-port-group-name
| | | | | +---rw (dest-ports)?
| | | | | +---:(port-number-range)
| | | | | +---rw acl-ip:des-port-lower
| | | | | +---rw acl-ip:des-port-upper
| | | | | +---:(port-number)
| | | | | +---rw acl-ip:des-comparator
```

```
| | +--rw acl-ip:des-port
| | +--:(port-group-ref)
| | +--rw acl-ip:des-port-group-name
+--rw acl-ip:icmp-type?
+--rw acl-ip:icmp-code?
+--rw (packet-length-or-range)?
| | +--:(length)
| | | +--rw acl-ip:packet-length-comparator
| | | +--rw acl-ip:packet-length
| | +--:(range)
| | +--rw acl-ip:packet-length-upper
| | +--rw acl-ip:packet-length-lower
+--rw acl-ip:tcp-flag-value?
+--rw acl-ip:tcp-flag-mask?
+--rw acl-ip:tcp-flag-operation?
+--rw (ttl-value-or-range)?
| | +--:(value)
| | | +--rw acl-ip:ttl-comparator?
| | | +--rw acl-ip:ttl-value?
| | +--:(range)
| | +--rw acl-ip:ttl-value-lower?
| | +--rw acl-ip:ttl-value-upper?
+--rw (dscp-or-tos)?
| | +--:(dscp)
| | | +--rw acl-ip:dscp?
| | +--:(tos)
| | +--rw acl-ip:tos?
| | +--rw acl-ip:precedence?
```

```
| | | +--rw acl-ip:igmp-type?
| | | +--rw acl-ip:flow-label?
| | | +--rw acl-ip:actions
| | | | +--rw acl-ip:action
| | | | +--rw acl-ip:log?
| | | +--ro acl-ip:match?
+--rw acl-ip:ipv4-aces
|   +--rw acl-ip:ipv4-ace [name]
|     +--rw acl-ip:name          acl:acl-name-string
|     +--rw (remark-or-ipv4-ace)?
|       |--:(remark)
|       | +--rw acl-ip:remark?    acl:acl-remark
|       +--:(ipv4-ace)
|         +--rw acl-ip:filters
|           +--rw (source-address-host-group)
|             | +--:(source-ip)
|             | | +--rw acl-ip:ip-source-address
|             | | +--rw acl-ip:ip-source-mask
|             | +--:(ip-source-any)
|             | +--rw acl-ip:ip-source-any?
```

```
| | | +--:(source-host)
| | |   |--rw (ip-src-address-or-name)
| | |     |--:(ip-source-host-address)
| | |       | |--rw acl-ip:ip-source-host-address?
| | |         |--:(ip-source-host-name)
| | |           |--rw acl-ip:ip-source-host-name?
| | |   +--:(source-group)
| | |     |--rw acl-ip:ip-source-group?
|--rw (dest-address-host-group)
| | |   |--:(dest-ip)
| | |     |--rw acl-ip:ip-dest-address
| | |     |--rw acl-ip:ip-dest-mask
| | |   +--:(ip-dest-any)
| | |     |--rw acl-ip:ip-dest-any?
| | |   +--:(dest-host)
| | |     |--rw (ip-dest-address-or-name)
| | |       |--:(ip-dest-host-address)
| | |         | |--rw acl-ip:ip-dest-host-address?
| | |           |--:(ip-dest-host-name)
| | |             |--rw acl-ip:ip-dest-host-name?
| | |   +--:(dest-group)
```

```
| | | | +---rw acl-ip:ip-dest-group?
| | | | +---rw acl-ip:protocol?
| | | | +---rw acl-ip:enable-capture?
| | | | +---rw acl-ip:capture-session-id?
| | | | +---rw acl-ip:fragments?
| | | | +---rw acl-ip:time-range?
| | | | +---rw (src-ports)?
| | | | | +---:(port-number-range)
| | | | | | +---rw acl-ip:src-port-lower
| | | | | | +---rw acl-ip:src-port-upper
| | | | | +---:(port-number)
| | | | | | +---rw acl-ip:src-comparator
| | | | | | +---rw acl-ip:src-port
| | | | | +---:(port-group-ref)
| | | | | +---rw acl-ip:src-port-group-name
+---rw (dest-ports)?
| | | | +---:(port-number-range)
| | | | | +---rw acl-ip:des-port-lower
| | | | | +---rw acl-ip:des-port-upper
| | | | +---:(port-number)
| | | | | +---rw acl-ip:des-comparator
| | | | | +---rw acl-ip:des-port
| | | | +---:(port-group-ref)
| | | | +---rw acl-ip:des-port-group-name
+---rw acl-ip:icmp-type?
+---rw acl-ip:icmp-code?
+---rw (packet-length-or-range)?
```

```
| | | | +--:(length)
| | | | | +-rw acl-ip:packet-length-comparator
| | | | | +-rw acl-ip:packet-length
| | | | +--:(range)
| | | | | +-rw acl-ip:packet-length-upper
| | | | | +-rw acl-ip:packet-length-lower
| | | +-rw acl-ip:tcp-flag-value?
| | | +-rw acl-ip:tcp-flag-mask?
| | | +-rw acl-ip:tcp-flag-operation?
| | | +-rw (ttl-value-or-range)?
| | | | +--:(value)
| | | | | +-rw acl-ip:ttl-comparator?
| | | | | +-rw acl-ip:ttl-value?
| | | | +--:(range)
```

```

| | | | +---rw acl-ip:ttl-value-lower?
| | | | +---rw acl-ip:ttl-value--upper?
| | | | +---rw (dscp-or-tos)?
| | | | | +---:(dscp)
| | | | | | +---rw acl-ip:dscp?
| | | | | +---:(tos)
| | | | | | +---rw acl-ip:tos?
| | | | | | +---rw acl-ip:precedence?
| | | | +---rw acl-ip:actions
| | | | | +---rw acl-ip:action      acl:acl-action
| | | | | +---rw acl-ip:log?       empty
| | | | +---ro acl-ip:match?        yang:counter64
+---rw acl-ip:global-fragments?     enumeration
+---rw acl-mac:mac-aces
|   +---rw acl-mac:mac-ace [name]
|     +---rw acl-mac:name          acl:acl-name-string
|     +---rw (remark-or-mac-ace)?
|       +---:(remark)
|       | +---rw acl-mac:remark?    acl:acl-remark
|       +---:(mac-ace)
|         +---rw acl-mac:filters
|           +---rw (source-network)
|             +---:(source-mac)
|               | +---rw acl-mac:source-address
|               | +---rw acl-mac:source-address-mask
|               +---:(source-any)
|                 | +---rw acl-mac:source-any?
|                 +---:(source-host)
|                   +---rw (src-address-or-name)
|                     +---:(source-host-address)
|                       | +---rw acl-mac:source-host-address?
|                       +---:(source-host-name)
|                         +---rw acl-mac:source-host-name?
+---rw (dest-network)

```

```
| | | | +--:(dest-mac)
| | | | | +--rw acl-mac:dest-address
| | | | | +--rw acl-mac:dest-address-mask
| | | | +--:(dest-any)
| | | | | +--rw acl-mac:dest-any?
| | | | +--:(dest-host)
| | | | | +--rw (dest-address-or-name)
```



```

| | | | +--rw acl-arp:ip-dest-mask
| | | | +---:(ip-dest-any)
| | | | | +--rw acl-arp:ip-dest-any?
| | | | | +---:(dest-host)
| | | | | | +--rw (ip-dest-address-or-name)
| | | | | | | +---:(ip-dest-host-address)
| | | | | | | | +--rw acl-arp:ip-dest-host-address?
| | | | | | | | +---:(ip-dest-host-name)
| | | | | | | | +--rw acl-arp:ip-dest-host-name?
| | | | | +---:(dest-group)
| | | | | | +--rw acl-arp:ip-dest-group?
| | | | +--rw (source-network)
| | | | | +---:(source-mac)
| | | | | | +--rw acl-arp:source-address
| | | | | | +--rw acl-arp:source-address-mask
| | | | | +---:(source-any)
| | | | | | +--rw acl-arp:source-any?
| | | | | +---:(source-host)
| | | | | | +--rw (src-address-or-name)
| | | | | | | +---:(source-host-address)
| | | | | | | | +--rw acl-arp:source-host-address?
| | | | | | | | +---:(source-host-name)
| | | | | | | | +--rw acl-arp:source-host-name?
| | | | +--rw (dest-network)
| | | | | +---:(dest-mac)
| | | | | | +--rw acl-arp:dest-address
| | | | | | +--rw acl-arp:dest-address-mask
| | | | | +---:(dest-any)
| | | | | | +--rw acl-arp:dest-any?
| | | | | +---:(dest-host)
| | | | | | +--rw (dest-address-or-name)
| | | | | | | +---:(dest-host-address)
| | | | | | | | +--rw acl-arp:dest-host-address?
| | | | | | | | +---:(dest-host-name)
| | | | | | | | +--rw acl-arp:dest-host-name?
| | | | | +--rw acl-arp:enable-capture?
| | | | | +--rw acl-arp:capture-session-id?
| | | +--rw acl-arp:actions
| | | | +--rw acl-arp:action
| | | | +--rw acl-arp:log?
| | +--ro acl-arp:match?
+--rw port-groups
| +--rw port-group [name]
| | +--rw name
| | +--rw port-group-entry [name]
| | | +--rw name
| | | +--rw (port-number-or-range)?
| | | | +---:(port-number-range)

```

Internet-Draft

yang-acl

October 2012

```
|          |  +---rw port-lower
|          |  +---rw port-upper
|          +---:(port-number)
|          +---rw comparator
|          +---rw port
+---rw timerange-groups
|  +---rw timerange-group [name]
|    +---rw name
|    +---rw time-range [name]
|      +---rw name
|      +---rw remark?
|      +---rw (range-type)?
|        +---:(absolute)
|          |  +---rw absolute
|            |  +---rw start?
|            |  +---rw end?
|            +---:(periodic)
|              +---rw periodic
|                +---rw weekdays?
|                +---rw start?
|                +---rw end?
+---rw ip-address-groups
  +---rw ip-address-group [name]
    +---rw name
    +---rw afi?
    +---rw ip-address [name]
      +---rw name
      +---rw (ip-network-kind)
        +---:(ip)
          |  +---rw ip-address?
          |  +---rw ip-mask
        +---:(ip-any)
          |  +---rw ip-any?
        +---:(host)
          +---rw (address-or-name)
            +---:(ip-host-address)
              |  +---rw ip-host-address?
            +---:(ip-host-name)
              +---rw ip-host-name?
```

module: acl-ip

module: acl-mac

module: acl-arp

Figure 3

[9.](#) ACL Examples

Here is an acl configuration example.

Requirement: Denies TELNET traffic from 14.3.6.234 bound for host 6.5.4.1 from leaving. Denies all TFTP traffic bound for TFTP servers. Permits all other IP traffic.

In order to achieve the requirement, an name access control list is needed. In the acl, we need three aces. The acl and aces can be described in CLI: as the following:

```
access-list ip iacl

deny tcp 14.3.6.234 0.0.0.0 host 6.5.4.1 eq 23
deny udp any any eq tftp
permit ip any any
```

Here is the example acl configuration xml:

```
<rpc message-id="101"
  xmlns:nc="urn:cisco:params:xml:ns:yang:acl:1.0"
  xmlns:acl-ip="urn:cisco:params:xml:ns:yang:acl-ip"
  // replace with IANA namespace when assigned
<edit-config>
  <target>
    <running/>
  </target>
  <config>
    <top xmlns="http://example.com/schema/1.2/config">

      <acls>
        <acl >
          <name>sample-ip-acl</name>
          <acl-type>ip-acl</acl-type>
          <enable-match-counter>false</enable-match-counter>
```

```

<acl-ip:afi>ipv4</acl-ip:afi>
<acl-ip:ipv4-aces>

  <acl-ip:ipv4-ace>
    <acl-ip:name>ace10</acl-ip:name>
    <acl-ip:filters>
      <acl-ip:protocol>6</acl-ip:protocol>
      <acl-ip:ip-source-address>
        14.3.6.234
      </acl-ip:ip-source-address>
      <acl-ip:ip-source-mask>0.0.0.0</acl-ip:ip-source-mask>
      <acl-ip:ip-dest-host-address>

```

```

        6.5.4.1
      </acl-ip:ip-dest-host-address>
      <acl-ip:des-comparator>eq</acl-ip:des-comparator>
      <acl-ip:des-port>23</acl-ip:des-port>
    </acl-ip:filters>
    <acl-ip:actions>
      <acl-ip:action>deny</acl-ip:action>
    </acl-ip:actions>
  </acl-ip:ipv4-ace>

  <acl-ip:ipv4-ace>
    <acl-ip:name>ace20</acl-ip:name>
    <acl-ip:filters>
      <acl-ip:protocol>17</acl-ip:protocol>
      <acl-ip:ip-source-any/>
      <acl-ip:ip-dest-any/>
      <acl-ip:des-comparator>eq</acl-ip:des-comparator>
      <acl-ip:des-port>69</acl-ip:des-port>
    </acl-ip:filters>
    <acl-ip:actions>
      <acl-ip:action>deny</acl-ip:action>
    </acl-ip:actions>
  </acl-ip:ipv4-ace>

  <acl-ip:ipv4-ace>
    <acl-ip:name>ace30</acl-ip:name>
    <acl-ip:filters>
      <acl-ip:ip-source-any/>
      <acl-ip:ip-dest-any/>

```

```

        </acl-ip:filters>
        <acl-ip:actions>
            <acl-ip:action>permit</acl-ip:action>
        </acl-ip:actions>
    </acl-ip:ipv4-ace>
</acl-ip:ipv4-aces>

</acl>
</acls>

</top>
</config>
</edit-config>
</rpc>

```

[10](#). ACL YANG Module

This module imports type definitions from [\[RFC6021\]](#).

```

<CODE BEGINS> file "acl@2012-10-12.yang"
module acl {
    namespace "urn:cisco:params:xml:ns:yang:acl";
    // replace with IANA namespace when assigned
    prefix acl;

    import ietf-inet-types {
        prefix "inet";
    }

    import ietf-yang-types {
        prefix "yang";
    }

    organization
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

    contact

```

"WG Web: <http://tools.ietf.org/wg/netmod/>
WG List: netmod@ietf.org

WG Chair: David Kessens
david.kessens@nsn.com

WG Chair: Juergen Schoenwaelder
j.schoenwaelder@jacobs-university.de

Editor: Lisa Huang
yihuan@cisco.com

Editor: Alexander Clemm
alex@cisco.com

Editor: Andy Bierman
andy@yumaworks.com";

description

"This YANG module defines a component that describing the configuration of Access Control Lists (ACLs).

An ACL is an ordered set of rules and actions used to filter traffic. Each set of rules and actions is represented as an Access Control Entries (ACE). Each ACE is evaluated sequentially. When the rule matches then action for that

rule is applied to the packet.

There are three types of ACL.

IP ACLs - IP ACLs are ordered sets of rules that can use to filter traffic based on IP information in the Layer 3 header of packets.

The device applies IP ACLs only to IP traffic. IP ACL can be IPv4 or IPv6.

MAC ACLs - MAC ACLs are used to filter traffic using the information in the Layer 2 header of each packet.

MAC ACLs are by default only applied to non-IP traffic; however, Layer 2 interfaces can be configured to apply MAC ACLs to all traffic.

ARP ACLs - The device applies ARP ACLs to IP traffic.

This module should be used with `acl-ip`, `acl-arp`, or `acl-mac` depends on what feature the device supports.

This YANG module also includes auxiliary definitions that are needed in conjunction with configuration of ACLs, such as reusable containers and references for ports and IP.

Terms and Acronyms

ACE (`ace`): Access Control Entry

ACL (`acl`): Access Control List

AFI (`afi`): Authority and Format Identifier (Address Field Identifier)

ARP (`arp`): Address Resolution Protocol

IP (`ip`): Internet Protocol

IPv4 (`ipv4`): Internet Protocol Version 4

IPv6 (`ipv6`): Internet Protocol Version 6

MAC: Media Access Control

TCP (`tcp`): Transmission Control Protocol

TTL (`ttl`): Time to Live

VLAN (`vlan`): Virtual Local Area Network
";

reference

"Access List Commands on Cisco IOS XR Software,
Cisco Nexus 7000 Series NX-OS Security Configuration Guide,
Catalyst 6500 Release 12.2SX Software Configuration Guide,
ACL TCP Flags Filtering";

revision 2012-10-12 {
description "Initial revision. ";


```

}

/* Features */

feature capture-session-id {
    if-feature packet-capture;
    description
        "The ability to configure ACL capture in order to
        selectively monitor traffic on an interface or VLAN.
        When the capture option for an ACL rule
        is enabled, packets that match this rule are
        either forwarded or dropped based on the specified permit
        or deny action and may also be copied to an alternate
        destination port for further analysis.
        An ACL rule with the capture option can be applied
        as follows:
            On a VLAN
            In the ingress direction on all interfaces
            In the egress direction on all Layer 3 interfaces
        The statistics data for the capture-session are capture
        in the device where the ACL rule applied to.";
}

feature host-by-name {
    description
        "The capability to reference a host by DNS name.";
}

feature ip-address-groups {
    description
        "The ability to define named groups for lists of
        ip addresses. ";
}

feature logging {
    description
        "The ability to log messages upon the matching of ACLs.";
}

feature match-counter {

```

```

        "The ability to maintain global or local match statistics
        for each ACL rules.";
    }

    feature packet-capture {
        description "The ability to capture packets that
        match the filter.";
    }

    feature packet-length {
        description "The ability to filter packets by packet length";
    }

    feature port-groups {
        description
            "The ability to define named groups for lists of ports. ";
    }

    /* Identities */

    identity acl-type {
        description "Base acl type for all ACL type identifiers.";
    }

    /* Types */

    typedef acl-comparator {
        description "A data type used to express comparator string";
        type enumeration {
            enum "eq" {
                value 0;
                description "match only equal to any giving number.";
            }
            enum "gt" {
                value 1;
                description
                    "match only greater than any giving number.";
            }
            enum "lt" {
                value 2;
                description
                    "match only lower than any giving number.";
            }
            enum "neq" {
                value 3;
            }
        }
    }

```

```
        description
            "match only not equal to any giving number";
    }
}

typedef acl-action {
    description "An enumeration data type to express acl
        action when match.";
    type enumeration {
        enum deny {
            description "Apply deny action to the traffic";
        }
        enum permit {
            description "Apply permit action to the traffic";
        }
    }
}

typedef acl-remark {
    type string {
        length "0..100";
    }
    description
        "A remark is a comment that can be
        associated with an ACE in order to make
        the access list easier for the network
        administrator to understand.
        It is retained to facilitate
        co-existence with CLI.";
}

typedef acl-type-ref {
    description
        "This type is used to refer to an Access Control List
        (ACL) type";
    type identityref {
        base "acl-type";
    }
}

typedef acl-ref {
    description "This type refers to an ACL.";
    type leafref {
```

```
        path "/acl:acls/acl:acl/acl:name";
    }
```

Internet-Draft

yang-acl

October 2012

```
    }

    typedef port-group-ref {
        description
            "This type is used to refer to a Portgroup object.";
        type leafref {
            path "/acls/port-groups/port-group/name";
        }
    }

    typedef ip-address-group-ref {
        description
            "This type is used to refer to a time range object.";
        type leafref {
            path "/acls/ip-address-groups/ip-address-group/name";
        }
    }

    typedef time-range-ref {
        description
            "This type is used to refer to a time range object.";
        type leafref {
            path "/acls/timerange-groups/timerange-group/name";
        }
    }

    typedef weekdays {
        type bits {
            bit Sunday {
                position 0;
            }
            bit Monday {
                position 1;
            }
            bit Tuesday {
                position 2;
            }
        }
    }
```

```

    bit Wednesday {
        position 3;
    }
    bit Thursday {
        position 4;
    }
    bit Friday {
        position 5;
    }

```

```

    bit Saturday {
        position 6;
    }
}

typedef acl-name-string {
    type string {
        length "1 .. 64";
    }
}

/* Groupings */

grouping ACE-COMMON {
    description
        "A collection of nodes that should be added to
        every ACE list entry";

    container actions {
        leaf action {
            type acl:acl-action;
            mandatory true;
            description "Permit/deny action.";
        }

        leaf log {
            if-feature acl:logging;
            type empty;
            description
                "Causes an informational logging message about the
                packet that matches the entry to be sent to the

```

```

        console.";
    }
}

leaf match {
    if-feature acl:match-counter;
    config false;
    type yang:counter64;
    description
        "The total packet that have matched for the
        particular ACE";
}

}

grouping FILTER-COMMON {
    description

```

```

    "A collection of nodes that should be added to
    every 'filters' container within each
    ACE list entry";

leaf enable-capture {
    if-feature acl:packet-capture;
    type boolean;
    description
        "Enable packet capture on this filter
        for this session.";
}

leaf capture-session-id {
    if-feature acl:capture-session-id;
    when "../enable-capture = 'true'";
    type uint32 {
        range "1..48";
    }
    description
        "Enable packet capture on this filter
        for this session id.";
}

}

/* Data Nodes */

```

```

container acls {
  description
    "This is the top container that contains a list of
    named ACL and reusable acl object groups.";
  list acl {
    key name;
    leaf name {
      description "ACL/access group name.";
      type acl-name-string;
    }

    leaf acl-type {
      type acl-type-ref;
      description "Type of ACL";
      mandatory true;
    }
    leaf enable-capture-global {
      if-feature packet-capture;
      type boolean;
      description "Enable packet capture on this filter
        for this session. Session ID range is 1 to 48";
      default "false";
    }
  }
}

```

```

}
leaf capture-session-id-global {
  if-feature capture-session-id;
  when "../enable-capture-global = 'true'";
  type uint32 {
    range "1..48";
  }
  description "Enable packet capture on this filter
    for this session. Session ID range is 1 to 48";
}
choice enable-match-counter-choices {
  if-feature match-counter;
  case match {
    leaf enable-match-counter {
      type boolean;
      description
        "Enable to collect statistics for the ACL";
      default false;
    }
  }
}

```

```

    }
  }
  case per-entry-match {
    leaf enable-per-entry-match-counter {
      type boolean;
      description "Enable to collect match
        statistics for each ACL entry(ACE).";
      default false;
    }
  }
}

leaf match {
  if-feature match-counter;
  config false;
  type yang:counter64;
  description
    "The total packet that have matched for the
    particular access list";
}

}

container port-groups {
  if-feature port-groups;
  list port-group {
    key "name";
    leaf name {
      type acl-name-string;
    }
  }
}

```

```

list port-group-entry {
  key "name";
  ordered-by user;
  leaf name {
    type acl-name-string;
  }
  //unique "comparator port-number
  //port-lower port-upper";

  choice port-number-or-range {
    case port-number-range {

```



```

        description
        "Port group includes all ports between
        port-lower and port-upper (including those)";
    leaf port-lower {
        type inet:port-number;
        description "Lower Port number.";
        mandatory true;
    }
    leaf port-upper {
        type inet:port-number;
        description "Upper Port number.";
        mandatory true;
        must "../port-lower <= ../port-upper";
    }
}
case port-number {
    description
    "Port group includes all ports that are greater
    than, greater or equal, less than, less or
    equal, or not equal the port, per the
    indicated comparator.
    It is possible for the port group to be empty
    (for example, in case a port group that
    is less than the minimum port number is
    specified).";
    leaf comparator {
        type acl-comparator;
        mandatory true;
    }
    leaf port {
        type inet:port-number;
        description "Port number.";
        mandatory true;
    }
}
} // choice port-number-or-range
} // list port-group-entry

```

```

    } // list port-group
} // container port-groups

container timerange-groups {

```

```

description "Define time range entries to restrict
the access. The time range is identified by a name
and then referenced by a function, so that those
time restrictions are imposed on the function itself.";
list timerange-group {
    key "name";
    leaf name {
        type acl-name-string;
    }
    list time-range {
        key "name";
        ordered-by user;
        leaf name {
            type acl-name-string;
        }

        leaf remark {
            type acl-remark;
        }

        choice range-type {
            // absolute or periodic time range
            container absolute {
                description
                "Absolute time and date that
                the associated function starts
                going into effect.";
                leaf start {
                    type yang:date-and-time;
                    description
                    "Absolute start time and date";
                }
                leaf end {
                    type yang:date-and-time;
                    description "Absolute end time and date";
                }
            }
            container periodic {
                description
                "To specify a periodic time and date.";
                leaf weekdays {
                    type weekdays;
                }
                leaf start {

```

```
        type yang:timestamp;
        description "Start time";
    }
    leaf end {
        type yang:timestamp;
        description "End time";
    }
}
} // choice range-type
} // list time-range
} // list timerange-group
} // container timerange-groups

container ip-address-groups {
    if-feature ip-address-groups;
    description
        "This contains a list of named ip address group. Each
        group defines a range of address and mask pair.";
    list ip-address-group {
        key "name";
        leaf name {
            type acl-name-string;
        }
        leaf afi {
            default "ipv4";
            type inet:ip-version;
            description "Address Field Identifier (AFI).";
        }
        list ip-address {
            key "name";
            ordered-by user;
            leaf name {
                type acl-name-string;
            }
            //unique "ip-address ip-mask";
            //unique "ip-host-address";

            grouping IP-HOST {
                description
                    "Choice within a case not allowed so need
                    this grouping.";
                choice address-or-name {
                    mandatory true;
                    leaf ip-host-address {
                        type inet:ip-address;
                    }
                    leaf ip-host-name {
```

if-feature acl:host-by-name;

Internet-Draft

yang-acl

October 2012

```
        type inet:domain-name;
    }
}

choice ip-network-kind {
    mandatory true;

    case ip {
        leaf ip-address {
            type inet:ip-address;
        }
        leaf ip-mask {
            type inet:ip-prefix;
            mandatory true;
        }
    }
    leaf ip-any {
        type empty;
        description "To express Any network or address.
            Use the any keyword as an abbreviation
            for an address and a mask of 0.0.0.0
            255.255.255.255. For example:
            0.0.0.0/255.255.255.255 means 'any'";
    }
    case host {
        description
            "Use the host address combination as an
            abbreviation for an address and wildcard
            of address 0.0.0.0";

        uses IP-HOST;
    }
    // case group not allowed here!
}

} // list ip-address
} // list ip-address-group
} // container ip-address-groups
} // container acls
```

```
}  
<CODE ENDS>
```

11. ACL-IP YANG Module

This module imports type definitions from [[RFC6021](#)] and common-types yang defined with acl model.

```
<CODE BEGINS> file "acl-ip@2012-10-12.yang"  
module acl-ip {  
    namespace "urn:cisco:params:xml:ns:yang:acl-ip";  
    // replace with IANA namespace when assigned  
    prefix acl-ip;  
  
    import acl {  
        prefix acl;  
    }  
    import ietf-inet-types {  
        prefix "inet";  
    }  
    import common-types {  
        prefix "c-types";  
    }  
  
    organization  
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";  
  
    contact  
        "WG Web: http://tools.ietf.org/wg/netmod/  
        WG List: netmod@ietf.org  
  
        WG Chair: David Kessens  
        david.kessens@nsn.com  
  
        WG Chair: Juergen Schoenwaelder  
        j.schoenwaelder@jacobs-university.de
```

Editor: Lisa Huang
yihuan@cisco.com

Editor: Alexander Clemm
alex@cisco.com

Editor: Andy Bierman
andy@yumaworks.com";

description

"This YANG module augments the 'acl' module with configuration and operational data for IPv4 and IPv6 access control list.

An ACL is an ordered set of rules and actions used to filter

Huang, et al.

Expires April 17, 2013

[Page 47]

Internet-Draft

yang-acl

October 2012

traffic.

Each set of rules and actions is represented as an Access Control Entries (ACE). Each ACE is evaluated sequentially. When the rule matches then action for that rule is applied to the packet.

IP ACLs are ordered sets of rules that can use to filter traffic based on IP information in the Layer 3 header of packets.

The device applies IP ACLs only to IP traffic. IP ACL can be IPv4 or IPv6.

Terms and Acronyms

ACE (ace): Access Control Entry

ACL (acl): Access Control List

AFI (afi): Authority and Format Identifier (Address Field Identifier)

DSCP (dscp): Differentiated Services Code Point

ICMP (icmp): Internet Control Message Protocol

IGMP (igmp): Internet Group Management Protocol

IP (ip): Internet Protocol

IPv4 (ipv4):Internet Protocol Version 4

IPv6 (ipv6): Internet Protocol Version 6

QoS: Quality of Service

TCP (tcp): Transmission Control Protocol

ToS (tos): Type of Service

TTL (ttl): Time to Live

UDP (udp): User Datagram Protocol

VLAN (vlan): Virtual Local Area Network

VRF(vrf) : Virtual Routing and Forwarding

";
reference

"Access List Commands on Cisco IOS XR Software,

Huang, et al.

Expires April 17, 2013

[Page 48]

Internet-Draft

yang-acl

October 2012

```
Cisco Nexus 7000 Series NX-OS Security Configuration Guide,  
Catalyst 6500 Release 12.2SX Software Configuration Guide,  
ACL TCP Flags Filtering";  
  
revision 2012-10-12 {  
    description "Initial revision. ";  
}  
  
/* Features */  
  
feature time-to-live {  
    description "The ability to filter packets based on their  
    time-to-live (TTL) value (0 to 255)";  
    reference "ACL Support for Filtering on TTL Value";  
}  
  
feature flow-label {  
    description  
        "The ability to filter packets based on flow lable.
```

```

        The 20-bit Flow Label field in the IPv6 header
        is used by a source to label packets
        of a flow. This is an IPv6 ACEs option.";
        reference "RFC 3697 IPv6 Flow Label Specification";
    }

```

```

/* Identities */

```

```

identity ip-acl {
    base "acl:acl-type";
    description "layer 3 ACL type";
}

```

```

/* Groupings */

```

```

grouping IP-SOURCE-NETWORK {
    description "Reusable IP address and mask pair.";

    grouping IP-SOURCE-HOST {
        description
            "Choice within a case not allowed so need
            this grouping.";
        choice ip-src-address-or-name {
            mandatory true;
            leaf ip-source-host-address {
                type inet:ip-address;
            }
            leaf ip-source-host-name {

```

```

        if-feature acl:host-by-name;
        type inet:domain-name;
    }
}

choice source-address-host-group {
    mandatory true;
    case source-ip {
        description "Used with address and mask couple
            to express network.";

```



```

        leaf ip-source-address {
            type inet:ip-address;
            mandatory true;
        }
        leaf ip-source-mask {
            type inet:ip-address;
            mandatory true;
        }
    }
    leaf ip-source-any {
        type empty;
        description "To express Any network or address.
            Use the any keyword as an abbreviation
            for an address and a mask of 0.0.0.0
            255.255.255.255. For example:
            0.0.0.0/255.255.255.255 means 'any'";
    }
    case source-host {
        description "Used with host address to express a
            single host
            Use the host address(or name)
            combination is the same as an address
            and mask of address 0.0.0.0.
            For example: '10.1.1.2/0.0.0.0' is the same
            as 'host 10.1.1.2'";
        uses IP-SOURCE-HOST;
    }
    case source-group {
        if-feature acl:ip-address-groups;
        leaf ip-source-group {
            type acl:ip-address-group-ref;
        }
    }
}

```

```

grouping IP-DESTINATION-NETWORK {
    description
        "Reusable IP address and mask pair for destination.";

    grouping IP-DESTINATION-HOST {

```

```

description
  "Choice within a case not allowed so need
  this grouping.";
choice ip-dest-address-or-name {
  mandatory true;
  leaf ip-dest-host-address {
    type inet:ip-address;
  }
  leaf ip-dest-host-name {
    if-feature acl:host-by-name;
    type inet:domain-name;
  }
}
}

choice dest-address-host-group {
  mandatory true;
  case dest-ip {
    description "Used with address and mask couple
    to express network.";
    leaf ip-dest-address {
      type inet:ip-address;
      mandatory true;
    }
    leaf ip-dest-mask {
      type inet:ip-address;
      mandatory true;
    }
  }
  leaf ip-dest-any {
    type empty;
    description "To express Any network or address.
    Use the any keyword as an abbreviation
    for an address and a mask of 0.0.0.0
    255.255.255.255. For example:
    0.0.0.0/255.255.255.255 means 'any'";
  }
  case dest-host {
    description "Used with host address to express a
    single host
    Use the host address(or name)
    combination is the same as an address
    and mask of address 0.0.0.0.

```

For example: '10.1.1.2/0.0.0.0' is the same as 'host 10.1.1.2';

```
        uses IP-DESTINATION-HOST;
    }
    case dest-group {
        if-feature acl:ip-address-groups;
        description "Use the group keyword and group name
            to refer to a pre-defined address object group
            which is a list of address and mask.";

        leaf ip-dest-group {
            type acl:ip-address-group-ref;
        }
    }
}

grouping DSCP-OR-TOS {
    choice dscp-or-tos {
        leaf dscp {
            type inet:dscp;
            description
                "Match packets with given dscp value";
        }

        case tos {
            leaf tos {
                type c-types:tos;
                description
                    "Match packets with given TOS value";
            }
            leaf precedence {
                when "boolean(..tos)" ;
                type c-types:precedence;
                description
                    "Match packets with given precedence value";
            }
        }
    }
}

grouping IP-ACE-FILTERS {
    leaf protocol {
        type c-types:ip-protocol;
        description "IP protocol number.";
    }
}
```

Internet-Draft

yang-acl

October 2012

```
uses acl:FILTER-COMMON;

leaf fragments {
    type empty;
    description "Check non-initial fragments";
}

leaf time-range {
    type acl:time-range-ref;
    description
        "Refer a time range object by
        name (Max Size 64).";
}

choice src-ports {
    when "protocol = '6' or protocol = '17' or " +
        "protocol = '132'";

    description
        "Apply only when the protocol is TCP,
        UDP or SCTP.";

    case port-number-range {
        description
            "Port group includes all ports between port-lower
            and port-upper (including those)";
        leaf src-port-lower {
            type inet:port-number;
            description "Lower Port number.";
            mandatory true;
        }
        leaf src-port-upper {
            type inet:port-number;
            description "Upper Port number.";
            mandatory true;
            must "../src-port-lower <= ../src-port-upper";
        }
    }
}

case port-number {
    description
        "Port group includes all ports that are greater
        than, greater or equal, less than, less or equal,
```

or not equal the port, per the indicated comparator. It is possible for the port group to be empty (for example, in case a port group that is less than the minimum port number is specified).";
leaf src-comparator {

```
        type acl:acl-comparator;
        mandatory true;
    }
    leaf src-port {
        type inet:port-number;
        description "Port number.";
        mandatory true;
    }
}
case port-group-ref {
    if-feature acl:port-groups;
    leaf src-port-group-name {
        type acl:port-group-ref;
        mandatory true;
        description
            "Reference a port group by the Port
            Group name.";
    }
}
} // choice src-ports

choice dest-ports {
    when "protocol = '6' or protocol = '17' or " +
        "protocol = '132'";
    description
        "Apply only when the protocol is TCP,
        UDP or SCTP.";

    case port-number-range {
        description "Port group includes all ports between
            port-lower and port-upper (including those)";
        leaf des-port-lower {
            type inet:port-number;
            description "Lower Port number.";
            mandatory true;
        }
    }
}
```

```

    }
    leaf des-port-upper {
        type inet:port-number;
        description "Upper Port number.";
        mandatory true;
        must "../des-port-lower <= ../des-port-upper";
    }
}
case port-number {
    description "Port group includes all ports that
        are greater than, greater or equal, less than,
        less or equal, or not equal the port, per the
        indicated comparator. It is possible for the

```

```

        port group to be empty (for example, in case a
        port group that is less than the minimum port
        number is specified).";
    leaf des-comparator {
        type acl:acl-comparator;
        mandatory true;
    }
    leaf des-port {
        type inet:port-number;
        description "Port number.";
        mandatory true;
    }
}
case port-group-ref {
    if-feature acl:port-groups;
    leaf des-port-group-name {
        type acl:port-group-ref;
        mandatory true;
        description
            "Reference a port group by the Port Group name.";
    }
}
} // choice dest-ports

leaf icmp-type {
    when "../protocol = '1'";
    type c-types:icmp-type;
    description

```

```

        "ICMP message type number.
        Apply only when the protocol is icmp";
    }

    leaf icmp-code {
        when "boolean(..icmp-type) ";
        type c-types:icmp-code;
        description
            "ICMP subtype for a given icmp type.";
    }

    choice packet-length-or-range {
        if-feature acl:packet-length;
        case length {
            leaf packet-length-comparator {
                type acl:acl-comparator;
                description
                    "Operant that compare the packet
                    length. Operands are lt (less than),
                    gt (greater than), eq (equal), and neq

```

```

            (not equal).";
            mandatory true;
        }
        leaf packet-length {
            type uint32 {
                range "20..9210";
            }
            description
                "Packet length value for
                operation gt, eq, etc, other
                than range";
            //TODO need to find out why package is
            // less than 9210
            mandatory true;
        }
    }
    case range {
        description
            "Packet operator 'range' takes
            both lower and upper value.";
    }

```

```

        leaf packet-length-upper {
            type uint32 {
                range "20..9210";
            }
            mandatory true;
            description "Upper Packet length";
        }

        leaf packet-length-lower {
            type uint32 {
                range "20..9210";
            }
            must "number(..../packet-length-lower) <= " +
                "number(..../packet-length-upper)";
            mandatory true;
            description "Lower packet length";
        }
    }

    leaf tcp-flag-value {
        type c-types:tcp-flag-type ;
        description "TCP flag bits that needs to be checked";
    }

    leaf tcp-flag-mask {
        when "boolean(..../tcp-flag-value)" ;
    }

```

```

        type c-types:tcp-flag-type ;
        description "TCP flag bit that needs to be checked";
    }

    leaf tcp-flag-operation {
        when "boolean(..../tcp-flag-value)" ;
        description
            "TCP flag Match option.
            A match occurs if the TCP
            datagram has certain TCP flags
            set or not set. You use the
            match-any keyword to allow a match
            to occur if any of the specified
            TCP flags are present, or you can

```



```

        use the match-all keyword to allow
        a match to occur only if all of
        the specified TCP flags are
        present. You must follow the
        match-any and match-all keywords
        with the + or - keyword and the
        flag-name argument to match on
        one or more TCP flags. ";
    default match-any;
    type enumeration {
        enum match-any {
            description "match any";
        }
        enum match-all {
            description "match all";
        }
    }
}

choice ttl-value-or-range {
    if-feature time-to-live;
    case value {
        leaf ttl-comparator {
            type acl:acl-comparator;

            description
                "Compares the TTL value in the packet
                to the TTL value specified in this
                ACE statement. Operands are lt (less
                than), gt (greater than), and eq
                (equal), neq (not equal).";
        }
        leaf ttl-value {
            type c-types:time-to-live;
        }
    }
}

```

```

    }
}
case range {
    leaf ttl-value-lower {
        type c-types:time-to-live;
        description "Lower ttl number.";
    }
}

```

```

        leaf ttl-value--upper {
            type c-types:time-to-live;
            description "Upper ttl number.";
        }
    }
}

/* Data Nodes */

augment "/acl:acls/acl:acl" {
    when "acl:acl-type = 'ip-acl'";

    leaf afi {
        type inet:ip-version ;
        default "ipv4";
    }

    container ipv6-aces {
        when "../afi = 'ipv6'";

        description
            " The ip-aces container contains a list of ip-ace.
            Each ip-ace is made of a unique ID, an optional
            remark (comment), and a filter. The filter
            requires a mandatory action (permit/deny) and one or
            more options such as source-address with mask,ttl etc";

        list ipv6-ace {
            key "name";
            ordered-by user;
            description "Layer 3 Access Control Element (ACE)";

            leaf name {
                type acl:acl-name-string;
                description "Unique ACE identifier.";
            }

            choice remark-or-ipv6-case {
                leaf remark {

```

```

        type acl:acl-remark;
        // mandatory true;
    }
    case ipv6-ace {
        container filters {

            uses IP-SOURCE-NETWORK;
            uses IP-DESTINATION-NETWORK;
            uses IP-ACE-FILTERS;
            uses DSCP-OR-TOS;

            leaf igmp-type {
                when "../protocol = '2' ";
                type c-types:igmp-code;
                description
                    "IGMP message type (0 to 15) for
                     filtering IGMP packets. Apply only
                     when the protocol is igmp in ipv4";
            }

            leaf flow-label {
                if-feature flow-label;
                when "../protocol = '17'";
                type uint64 {
                    range "0..1048575";
                }
                description
                    "Flow label value. Apply only when
                     the protocol is UDP in ipv6.";
                reference
                    "RFC3697 IPv6 Flow Label Specification";
            }
        } // container filters

        uses acl:ACE-COMMON;
    } // case ipv6-ace
} // choice remark-or-ipv6-ace
} // list ipv6-ace
} // container ipv6-aces

container ipv4-aces {
    when "../afi = 'ipv4' " ;

    description
        "The ip-aces container contains a list of ip-ace.
         Each ip-ace is made of a unique ID, an optional
         remark (comment), and a filter. The filter requires a
         mandatory action (permit/deny) and one or more options

```

Internet-Draft

yang-acl

October 2012

```
        such as source-address with mask,ttl etc";

list ipv4-ace {
    key "name";
    ordered-by user;
    description "Layer 3 Access Control Element (ACE)";

    leaf name {
        type acl:acl-name-string;
        description "Unique ACE identifier";
    }

    choice remark-or-ipv4-ace {
        leaf remark {
            type acl:acl-remark;
            // mandatory true;
        }
        case ipv4-ace {
            container filters {
                uses IP-SOURCE-NETWORK;
                uses IP-DESTINATION-NETWORK;
                uses IP-ACE-FILTERS;
                uses DSCP-OR-TOS;
            }
            uses acl:ACE-COMMON;
        } // case ipv4-ace
    } // choice remark-or-ipv4-ace
} // list ipv4-ace
} // container ipv4-aces

leaf global-fragments {
    default "not-set";
    type enumeration {
        enum not-set;
        enum permit-all {
            description "Allow all fragments";
        }
        enum deny-all {
            description "Drop all fragments";
        }
    }
}
description
    "Optimizes fragment handling for noninitial fragments.
```

When this leaf is set to 'permit-all', noninitial fragments will be permitted unless explicitly denied. When this leaf is set to 'deny-all', noninitial fragments will be denied unless explicitly permitted. ";

Internet-Draft

yang-acl

October 2012

```
    }  
  }  
  
}  
  
</CODE ENDS>
```

[12.](#) ACL-MAC Configuration YANG Module

This module imports type definitions from common-types YANG defined in this model.

<CODE BEGINS> file "acl-mac@2012-10-12.yang"

```
module acl-mac {  
  namespace "urn:cisco:params:xml:ns:yang:acl-mac";  
  // replace with IANA namespace when assigned  
  prefix acl-mac;  
  
  import acl { prefix acl; }  
  
  import common-types {  
    prefix "c-types";  
  }  
  
  import ietf-inet-types {  
    prefix "inet";  
  }  
  
  import ietf-yang-types {  
    prefix "yang";  
  }  
  
  organization  
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
```

contact

"WG Web: <http://tools.ietf.org/wg/netmod/>
WG List: netmod@ietf.org

WG Chair: David Kessens
david.kessens@nsn.com

WG Chair: Juergen Schoenwaelder
j.schoenwaelder@jacobs-university.de

Editor: Lisa Huang
yihuan@cisco.com

Huang, et al.

Expires April 17, 2013

[Page 61]

Internet-Draft

yang-acl

October 2012

Editor: Alexander Clemm
alex@cisco.com

Editor: Andy Bierman
andy@yumaworks.com";

description

"This YANG module augments the 'acl' module with configuration and operational data for MAC access control list

An ACL is an ordered set of rules and actions used to filter traffic.

Each set of rules and actions is represented as an Access Control Entries (ACE). Each ACE is evaluated sequentially. When the rule matches then action for that rule is applied to the packet.

MAC ACLs - MAC ACLs are used to filter traffic using the information in the Layer 2 header of each packet.

MAC ACLs are by default only applied to non-IP traffic; however, Layer 2 interfaces can be configured to apply MAC ACLs to all traffic.

Terms and Acronyms

ACE (ace): Access Control Entry

ACL (acl): Access Control List

AFI (afi): Authority and Format Identifier (Address Field

Identifier)

CoS (cos): Class of Service

MAC: Media Access Control

TTL (ttl): Time to Live

VLAN (vlan): Virtual Local Area Network

VRF(vrf) : Virtual Routing and Forwarding
";

reference

"Access List Commands on Cisco IOS XR Software,
Cisco Nexus 7000 Series NX-OS Security Configuration Guide,
Catalyst 6500 Release 12.2SX Software Configuration Guide";

revision 2012-10-12 {
description "Initial revision. ";

Huang, et al.

Expires April 17, 2013

[Page 62]

Internet-Draft

yang-acl

October 2012

}

/* Features */

feature ethertype-mask {
description
"The ability to filter packets based on ether-type mask
in hex 0x0-0xFFFF."
}

/* Identities */

identity mac-acl {
base acl:acl-type;
description "layer 2 ACL type";
}

/* Groupings */

grouping MAC-SOURCE-NETWORK {
description "MAC address and mask pair for source.";

```

grouping MAC-SOURCE-HOST {
  description
    "Choice within a case not allowed so need
    this grouping.";
  choice src-address-or-name {
    mandatory true;
    leaf source-host-address {
      type inet:ip-address;
      description
        "Use the host address combination as an
        abbreviation for an address and wildcard
        of address 0.0.0.0";
    }
    leaf source-host-name {
      if-feature acl:host-by-name;
      type inet:domain-name;
    }
  }
}

```

```

choice source-network {
  mandatory true;
  case source-mac {
    description
      "Used with address and mask couple to
      express network.";
  }
}

```

```

leaf source-address {
  type yang:mac-address;
  mandatory true;
  description "A source MAC address.";
}
leaf source-address-mask {
  type yang:mac-address;
  mandatory true;
  description "A source MAC address mask.";
}
leaf source-any {
  type empty;
  description "To express Any network or address";
}

```



```

        case source-host {
            description
                "Use the host address combination as an
                abbreviation for an address and wildcard
                of address 0.0.0.0";
            uses MAC-SOURCE-HOST;
        }
    }
}

grouping MAC-DESTINATION-NETWORK {
    description
        "MAC address and mask pair for destination.";

    grouping MAC-DESTINATION-HOST {
        description
            "Choice within a case not allowed so need
            this grouping.";
        choice dest-address-or-name {
            mandatory true;
            leaf dest-host-address {
                type inet:ip-address;
                description
                    "Use the host address combination as an
                    abbreviation for an address and wildcard
                    of address 0.0.0.0";
            }
            leaf dest-host-name {
                if-feature acl:host-by-name;
                type inet:domain-name;
            }
        }
    }
}

```

```

choice dest-network {
    mandatory true;
    case dest-mac {
        description
            "Used with address and mask couple to
            express network.";
        leaf dest-address {
            type yang:mac-address;

```

```

        mandatory true;
        description "A source MAC address.";
    }
    leaf dest-address-mask {
        type yang:mac-address;
        mandatory true;
        description "A source MAC address mask.";
    }
}
leaf dest-any {
    type empty;
    description "To express Any network or address";
}
case dest-host {
    description
        "Use the host address combination as an
        abbreviation for an address and wildcard
        of address 0.0.0.0";
    uses MAC-DESTINATION-HOST;
}
}
}

/* Layer 2 ACL */

augment "/acl:acls/acl:acl" {
    when "acl:acl-type = 'mac-acl'";
    description
        "Layer 2 Access Control Entry (ACE). The mac-aces
        container contains a list of mac-ace. Each mac-ace is
        comprised of a name, an optional remark
        and a rule.
        A rule is referred to as 'packet-filter', although it
        contains both a filter and an action.
        The packet-filter requires a mandatory action (permit/deny)
        and one or more options such as source-address with mask,
        ethertype, vlan etc.";
    container mac-aces {
        list mac-ace {
            key name;

```

```

leaf name {
    type acl:acl-name-string;
    description "Unique ACE identifier";
}

choice remark-or-mac-ace {
    leaf remark {
        type acl:acl-remark;
        // mandatory true;
    }
    case mac-ace {
        container filters {
            uses MAC-SOURCE-NETWORK;
            uses MAC-DESTINATION-NETWORK;

            leaf ethertype {
                type c-types:ether-type;
                description "ether-type (also known as
                    protocol) in hex 0x0-0xffff";
            }

            leaf ethertype-mask {
                if-feature ethertype-mask;
                when "boolean(..ethertype)";
                type c-types:ether-type;
                default "0x0000";
                description
                    "Ether-type mask in hex 0x0-0xFFFF.
                    0x0 is exactly match of the Ethertype..";
            }
        }

        leaf cos {
            type c-types:cos;
            description "CoS value <0-7>";
        }

        leaf time-range {
            type acl:time-range-ref;
            description
                "Enable packet capture on this
                filter for a specify time range
                by name.";
        }

        leaf vlan {
            type c-types:vlan-identifier;

```

```
        description "VLAN number";
    }

    uses acl:FILTER-COMMON;

} // container filters

    uses acl:ACE-COMMON;

    } // case mac-ace
    } // choice remark-or-ace
    } // list mac-ace
    } // container mac-aces
} // augment

}
```

</CODE ENDS>

13. ACL-ARP Configuration YANG Module

<CODE BEGINS> file "acl-arp@2012-10-12.yang"

```
module acl-arp {
    namespace "urn:cisco:params:xml:ns:yang:acl-arp";
    // replace with IANA namespace when assigned
    prefix acl-arp;

    import acl { prefix acl; }
    import acl-ip { prefix acl-ip; }
    import acl-mac { prefix acl-mac; }

    organization
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

    contact
        "WG Web: http://tools.ietf.org/wg/netmod/
        WG List: netmod@ietf.org

        WG Chair: David Kessens
        david.kessens@nsn.com

        WG Chair: Juergen Schoenwaelder
        j.schoenwaelder@jacobs-university.de
```

Editor: Lisa Huang
yihuan@cisco.com

Huang, et al.

Expires April 17, 2013

[Page 67]

Internet-Draft

yang-acl

October 2012

Editor: Alexander Clemm
alex@cisco.com

Editor: Andy Bierman
andy@yumaworks.com";

description

"This YANG module augments the 'acl' module with configuration and operational data for ARP access control list

An ACL is an ordered set of rules and actions used to filter traffic.

Each set of rules and actions is represented as an Access Control Entries (ACE). Each ACE is evaluated sequentially. When the rule matches then action for that rule is applied to the packet.

ARP ACLs - The device applies ARP ACLs to IP traffic.

Terms and Acronyms

ACE (ace): Access Control Entry

ACL (acl): Access Control List

ARP (arp): Address Resolution Protocol

IP (ip): Internet Protocol

MAC: Media Access Control

VLAN (vlan): Virtual Local Area Network

";

reference

"Access List Commands on Cisco IOS XR Software, Cisco Nexus 7000 Series NX-OS Security Configuration Guide, Catalyst 6500 Release 12.2SX Software Configuration Guide, ACL TCP Flags Filtering";

revision 2012-10-12 {

description "Initial revision. ";

```

}

/* Identities */

identity arp-acl {
    base "acl:acl-type";
    description "ARP ACL type";
}

```

```

/* Data Nodes */

augment "/acl:acls/acl:acl" {
    when "acl:acl-type = 'arp-acl'";

    description "ARP Access Control Entry (ACE).";
    container arp-aces {
        list arp-ace {
            key "name";
            ordered-by user;

            leaf name {
                type acl:acl-name-string;
            }

            choice remark-or-arp-ace {
                leaf remark {
                    type acl:acl-remark;
                    // mandatory true;
                }
                case arp-ace {
                    container filters {
                        leaf direction {
                            default "bi-direction";
                            type enumeration {
                                enum bi-direction;
                                enum request;
                                enum response;
                            }
                        }
                        description "ARP request/response.";
                    }
                }
            }
        }
    }
}

```

```

        uses acl-ip:IP-SOURCE-NETWORK;
        uses acl-ip:IP-DESTINATION-NETWORK {
            when "../direction = 'response'";
        }

        uses acl-mac:MAC-SOURCE-NETWORK;
        uses acl-mac:MAC-DESTINATION-NETWORK {
            when "../direction = 'response'";
        }

        uses acl:FILTER-COMMON;

    } // container filters

    uses acl:ACE-COMMON;

```

```

        } // case arp-ace
    } // choice remark-or-arp-ace
} // list arp-ace
} // container arp-aces
} // augment

}

</CODE ENDS>

```

14. COMMON-TYPES YANG Module

<CODE BEGINS> file "common-types@2012-10-12.yang"

```

module common-types {
    namespace "urn:cisco:params:xml:ns:yang:common-types";
    // replace with IANA namespace when assigned
    prefix c-types;

    organization
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

    contact
        "WG Web: http://tools.ietf.org/wg/netmod/

```

[WG List: netmod@ietf.org](mailto:netmod@ietf.org)

WG Chair: David Kessens
david.kessens@nsn.com

WG Chair: Juergen Schoenwaelder
j.schoenwaelder@jacobs-university.de

Editor: Lisa Huang
yihuan@cisco.com

Editor: Alexander Clemm
alex@cisco.com

Editor: Andy Bierman
andy@yumaworks.com";

description

"This module contains a collection of generally useful YANG types could be referred from multiple speciality components.

Terms and Acronyms

Huang, et al.

Expires April 17, 2013

[Page 70]

Internet-Draft

yang-acl

October 2012

CoS (cos): Class of Service

ICMP (icmp): Internet Control Message Protocol

IGMP (igmp): Internet Group Management Protocol

IP (ip): Internet Protocol

IPv4 (ipv4): Internet Protocol Version 4

IPv6 (ipv6): Internet Protocol Version 6

TCP (tcp): Transmission Control Protocol

ToS (tos): Type of Service

TTL (ttl): Time to Live


```

    UDP (udp): User Datagram Protocol

    VLAN (vlan): Virtual Local Area Network
";
revision 2012-10-12 {
    description "Initial revision. ";
}

/* Typedefs */

typedef cos {
    type uint8 {
        range "0..7";
    }
    description
        "Class of Service.
        An integer that is in the range of the layer 2 CoS values.
        This corresponds to the 802.1p and ISL CoS values.";
    reference "IEEE 802.1p";
}

typedef tos {
    type uint8 {
        range "0..15";
    }
    description
        "tos stands for Type of service .
        The tos field are five bits in the IPv4 header.
        It could specify a datagrams priority and
        request a route for low-delay, high-throughput,

```

or highly-reliable service.

Based on these TOS values, a packet would be placed in an prioritized outgoing queue, or take a route with appropriate latency, throughput, or reliability. The following are TOS field values (expressed as binary numbers):

1000	--	minimize delay
0100	--	maximize throughput
0010	--	maximize reliability

```
0001 -- minimize monetary cost
0000 -- normal service
```

```
.";
```

```
reference
```

```
"RFC 791 Internet Protocol
  Protocol Specification
RFC 1122 Requirements for Internet Hosts --
  Communication Layers
RFC 1349 Type of Service in the Internet Protocol
  Suite
RFC 2474 Definition of the Differentiated Services
  Field (DS Field)
  in the IPv4 and IPv6 Headers
RFC 3168 The Addition of Explicit Congestion
  Notification (ECN) to IP
";
```

```
}
```

```
typedef precedence {
  type uint8 {
    range "0..7";
  }
}
```

```
description
```

```
"Indicates the IP precedence.
Precedence is three bits in IP header.
```

Value	Description
000 (0)	Routine or Best Effort
001 (1)	Priority
010 (2)	Immediate
011 (3)	Flash - mainly used for Voice Signaling or for Video.
100 (4)	Flash Override
101 (5)	Critical -mainly used for Voice RTP.

```
110 (6) Internet
111 (7) Network";
```

```
reference
```

```

    "RFC 791 Internet Protocol Chapter 3.1
    Protocol Specification";
}

typedef tcp-flag-type {
    type bits {
        bit fin {
            position 0;
            description "No more data from sender";
        }
        bit syn {
            position 1;
            description "Synchronize sequence numbers";
        }
        bit rst {
            position 2;
            description "Reset the connection";
        }
        bit psh {
            position 3;
            description "Push Function";
        }
        bit ack {
            position 4;
            description "Acknowledgment field significant";
        }
        bit urg {
            position 5;
            description "Urgent Pointer field significant";
        }
    }
    description "TCP flag type";
    reference "RFC 793 TRANSMISSION CONTROL PROTOCOL";
}

typedef ether-type {
    type string {
        pattern '0x[0-9a-fA-F]{4}';
    }
    description
        "ether-type is 0x0-0xffff. The protocol number
        is a four-byte hexadecimal number prefixed with 0x.
        Valid protocol numbers are from 0x0 to 0xffff."
}

```

This list shows the EtherType values and their corresponding protocol keywords:

0x0600 xns-idp Xerox XNS IDP

0x0BAD vines-ip Banyan VINES IP

0x0baf vines-echo Banyan VINES Echo

0x6000 etype-6000 DEC unassigned, experimental

0x6001 mop-dump DEC Maintenance Operation Protocol
(MOP) Dump/Load Assistance

0x6002 mop-console DEC MOP Remote Console

0x6003 decnet-iv DEC DECnet Phase IV Route

0x6004 lat DEC Local Area Transport (LAT)

0x6005 diagnostic DEC DECnet Diagnostics

0x6007 lavc-sca DEC Local-Area VAX Cluster (LAVC), SCA

0x6008 amber DEC AMBER

0x6009 mumps DEC MUMPS

0x0800 ip Malformed, invalid, or deliberately corrupt
IP frames

0x8038 dec-spanning DEC LANBridge Management

0x8039 dsm DEC DSM/DDP

0x8040 netbios DEC PATHWORKS DECnet NETBIOS Emulation

0x8041 msdos DEC Local Area System Transport

0x8042 etype-8042 DEC unassigned

0x809B appletalk Kinetics EtherTalk (AppleTalk over
Ethernet)

0x80F3 aarp Kinetics AppleTalk Address Resolution
Protocol (AARP)

Internet-Draft

yang-acl

October 2012

```

        bpdu-snap      BPDU SNAP encapsulated packets
        ipx-arpa       IPX Advanced Research Projects Agency
                        (ARPA)
        ipx-non-arpa   IPX non arpa
        lacp           Link Aggregation Control Protocol(LACP)
                        encapsulated packets
        pagp           Port Aggregation Protocol(PAGP)
                        encapsulated packets
        vtp            VTP packets
        ";
    }

typedef ip-protocol {
    type uint8{
        range "0..255";
    }
    description
        "The Internet Protocol (IP) is the principal communications
        protocol used for relaying datagrams (also known as network
        packets) across an internetwork using the Internet Protocol
        Suite.

        IP protocol number value is 0 to 255. It is an 8 bit field
        in the packet header";
    reference
        "IANA Protocol Numbers
        RFC5237 IANA Allocation Guidelines for the Protocol Field";
}

typedef igmp-code {
    //TODO: need more work. In NxOs, range is 0..15.
    // Could not match the IGMP with 0..15
    type uint8 ;/* {
        range "0..15";
    }*/
    //IGMP v1 4 bits 0-15
    //IGMP v2 8bits. 0-
    //NXOS only support v1, but XR support v2.
    //

```

description

"Many of these IGMP types have a 'code' field. Here is the list of the types again with their assigned code fields.

Type	Name	Reference
-----	-----	-----
0x11	IGMP Membership Query	[RFC1112]

Huang, et al.

Expires April 17, 2013

[Page 75]

Internet-Draft

yang-acl

October 2012

0x12	IGMPv1 Membership Report	[RFC1112]
0x13	DVMRP	[RFCDVMRP]
0x14	PIM version 1	[PIMv1]
0x15	Cisco Trace Messages	
0x16	IGMPv2 Membership Report	[RFC2236]
0x17	IGMPv2 Leave Group	[RFC2236]
0x1e	Multicast Traceroute Response	[Fenner]
0x1f	Multicast Traceroute	[Fenner]
0x22	IGMPv3 Membership Report	[RFC3376]

";

reference

"IANA Internet Group Management Protocol (IGMP) Type Numbers";

}

typedef icmp-type {

type uint32 {

range "0..255";

}

description

"icmp-type is the Internet Control Message Protocol (ICMP) 'type' field.

The ICMP header starts after the IPv4 header. All ICMP packets will have an 8-byte header and variable-sized data section.

The first 4 bytes of the header will be consistent.

The first byte is for the ICMP type. The second byte is for the ICMP code.

ICMP type is specified below

Type	Name	Reference
----	-----	-----
0	Echo Reply	[RFC792]

1	Unassigned	[JBP]
2	Unassigned	[JBP]
3	Destination Unreachable	[RFC792]
4	Source Quench	[RFC792]
5	Redirect	[RFC792]
6	Alternate Host Address	[JBP]
7	Unassigned	[JBP]
8	Echo	[RFC792]
9	Router Advertisement	[RFC1256]
10	Router Selection	[RFC1256]
11	Time Exceeded	[RFC792]
12	Parameter Problem	[RFC792]
13	Timestamp	[RFC792]
14	Timestamp Reply	[RFC792]
15	Information Request	[RFC792]

Huang, et al.

Expires April 17, 2013

[Page 76]

Internet-Draft

yang-acl

October 2012

16	Information Reply	[RFC792]
17	Address Mask Request	[RFC950]
18	Address Mask Reply	[RFC950]
19	Reserved (for Security)	[Solo]
20-29	Reserved (for Robustness Experiment)	[ZSu]
30	Traceroute	[RFC1393]
31	Datagram Conversion Error	[RFC1475]
32	Mobile Host Redirect	[David Johnson]
33	IPv6 Where-Are-You	[Bill Simpson]
34	IPv6 I-Am-Here	[Bill Simpson]
35	Mobile Registration Request	[Bill Simpson]
36	Mobile Registration Reply	[Bill Simpson]
37-255	Reserved	[JBP]";

reference

"[RFC1700](#) ASSIGNED NUMBERS

[RFC792](#) Internet Control Message Protocol

[RFC4443](#) Internet Control Message Protocol (ICMPv6)
for the Internet Protocol Version 6 (IPv6)
Specification

[RFC2780](#) IANA Allocation Guidelines For Values In
the Internet Protocol and Related Headers";

}

```
typedef icmp-code {
    type uint32 {
        range "0..255";
```

```

    }
    description
        "ICMP subtype to the given type.
        The ICMP header starts after the IPv4 header. All ICMP
        packets will have an 8-byte header and variable-sized
        data section.
        The first 4 bytes of the header will be consistent.
        The first byte is for the ICMP type. The second byte
        is for the ICMP code. ";
    reference "RFC2 INTERNET CONTROL MESSAGE PROTOCOL";
}

typedef vlan-identifier {
    type uint16 {
        range "1 .. 4095";
    }
    description
        "This type denotes a VLAN tag. ";
    reference
        "RFC3069 VLAN Aggregation for Efficient IP Address
        Allocation
        IEEE 802.1Q";
}

```

```

    }

    typedef time-to-live {
        type uint8 {
            range "0..255";
        }
        description "The TTL is an 8-bit field in IP header.
        The maximum TTL value is 255.";
    }
}

</CODE ENDS>

```

[15.](#) Security Considerations

.

[16.](#) Open Issues

1. Are there any compatibility issues related to ACE ordering because a YANG user-order list is used instead of sequence IDs?
2. Is an administrative function to test a packet against a specified ACL needed? The server would return an indication of permit or deny, and a leaf-list of the ACE entries that were evaluated.

.

17. Acknowledgements

The author wish to acknowledge the contributions of Louis Fourie and Dana Blair to the design of this YANG data model and also wishes to thank Tula Kraiser, Patrick Gili, and George Serpa for their helpful comments and suggestions.

18. Normative References

- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), October 2010.
- [RFC6021] Schoenwaelder, J., "Common YANG Data Types", [RFC 6021](#), October 2010.

Authors' Addresses

Lisa Huang
Cisco Systems

EMail: yihuan@cisco.com

Alexander Clemm
Cisco Systems

EMail: alex@cisco.com

Andy Bierman
YumaWorks

EMail: andy@yumaworks.com