Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 4, 2013

L. Huang
A. Clemm
Cisco Systems
August 31, 2012

YANG Data Model for Access Control List Configuration
draft-huang-netmod-acl-00.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 March 4, 2013.

Copyright Notice

Table of Contents

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.

The applying of ACL to device configuration to interfaces and other components is out of 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" 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-obj-grp
          |  +--rw port-group [name]
          |     +--rw name
          |     +--rw groups
         +--rw timerange-obj-grp
          |  +--rw timerange-group [name]
          |     +--rw name
          |     +--rw time-ranges
         +--rw ip-address-obj-grp
    rpcs:
      +--x reset-match-counter
         +--ro input
            +--ro name
            +--ro sequence-num?
```

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 list element in that list, which is 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 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. the rpc "reset-match-counter" can be used to

reset that counter.

port-obj-grp, timerange-obj-grp, ip-address-obj-grp are auxiliary objects used to define groupings of ports and of IP ranges 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 respecify 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 indicate 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 the reusable objects, port-obj-grp, timerange-obj-grp, and ip-address-obj-grp, or general objects 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 [sequence-num]
         |  |    +--rw acl-ip:sequence-num
         |  |    +--rw (remark-or-filter)?
         |  |        +--:(remark)
         |  |        |   +--rw acl-ip:remark
         |  |        +--:(filter)
         |  |        |   +--rw acl-ip:filter
         |  |        |       +-- filter parameters
         |  |        |   +--rw acl-ip:actions
         |  |        |       +-- action parameters
         |  |        +-- ro acl-ip:statistics
```


module: acl

```
   +--rw acls
       +--rw acl [name]
         |  +--rw acl-ip:afi
         |  +--rw acl-ip:ipv4-aces
         |  |  +--rw acl-ip:ipv4-ace [sequence-num]
         |  |     +--rw acl-ip:sequence-num
         |  |     +--rw (remark-or-filter)?
         |  |        +--:(remark)
         |  |        |   +--rw acl-ip:remark
         |  |        +--:(filter)
         |  |        |   +--rw acl-ip:filters
         |  |        |       +-- filter parameters
         |  |        |   +--rw acl-ip:actions
         |  |        |       +-- action parameters
         |  |        +-- ro acl-ip:match


   module: acl
       +--rw acls
           +--rw acl [name]
             |  +--rw acl-mac:mac-aces
             |  |  +--rw acl-mac:mac-ace [sequence-num]
             |  |     +--rw acl-mac:sequence-num
             |  |     +--rw (remark-or-filter)?
             |  |        +--:(remark)
             |  |        |   +--rw acl-mac:remark
             |  |        +--:(filter)
             |  |        |   +--rw acl-mac:filters
             |  |        |       +-- filter parameters
             |  |        |   +--rw acl-mac:actions
             |  |        |       +-- action parameters
             |  |        +-- ro acl-mac:match

   module: acl
       +--rw acls
           +--rw acl [name]
             |  +--rw acl-arp:arp-aces
             |  |  +--rw acl-arp:arp-ace [sequence-num]
             |  |     +--rw acl-arp:sequence-num
             |  |     +--rw (remark-or-filter)?
             |  |        +--:(remark)
             |  |        |   +--rw acl-arp:remark
             |  |        +--:(filter)
             |  |        |   +--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 numeric identifier by which ACEs in the list are ordered. (Numeric identifiers do not have to be sequential, but are unique.) Each ACE consists either of a remark or an actual access control rule, i.e. a "filter". 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 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 compleixty of definitions within the modules as low as possible, at the price of not needing to repeat a few data nodes that provide the overall top level structure.

4. acl Module

acl module is a top container module for all acls. It contains a list of named acl. acl-ip, acl-map and acl-arp augment the acl list in acls container. In addition to the acls container, it also defined features, reusable types and reusable grouping.

4.1. Features

When it comes to ACL implementation, there is a wide range of capabilities 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. The following data types are introduced in this data model. Table 1 has the data types that is unique to acl model. Table 2 has the data types that will reused in other models defined in common-types module. For details of each type, see each typedef descriptions and reference in the model.

YANG type	base type
Comparator	enumeration
ACL-Action	enumeration
IP-Network-Kind	enumeration
Mac-Network-Kind	enumeration
Sequence-Number	uint32
Remark	string
Acl-Type-Ref	identityref
Acl-Ref	leafref
Port-Group-Ref	leafref
IP-Address-Group-Ref	leafref
Time-Range-Ref	leafref

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 a number of groupings, consisting of sets of objects that are used in multiple places. In order to keep the overview of the data model concise, those groupings are introduced in the following subsections and subsequently referred to by name, rather than expanding groupings in the data model.

4.3.1. PORT-GROUP grouping

```
PORT-GROUP
+--rw (port-number-or-range)?
  +--:(port-number-range)
  |  +--rw port-lower?        inet:port-number
  |  +--rw port-upper?        inet:port-number
  +--:(port-number)
    +--rw comparator         Comparator
    +--rw port?              inet:port-number
```

PORT-GROUP has two ways to express a group of ports:

Using port-upper and port-lower couple to specify a port range. The value of port-lower must be less or equal to port-upper

Using comparator and port couple to specify the port(s). For example:

```
<port-lower>20</port-lower>
<port-upper>21</port-upper>

<comparator>eq</comparator>
```

```
        <port>80</port>
```

See Comparator typedef in model appendix for possible comparator values.

4.3.2. MAC-SOURCE-NETWORK grouping

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


MAC-SOURCE-ADDRESS is a reusable grouping. It uses four leaves: mac-source-kind, source-address, source-address-mask and source-host-name to express the three kinds network.

any network: use mac-source-kind value any to express any network.

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

single host network.

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

host address with a mask.

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

4.3.3. MAC-DESTINATION-NETWORK grouping

```
MAC-DESTINATION-NETWORK
    +--rw acl-mac:mac-destination-kind?           MAC-Network-Kind
      +--rw (dest-network)?
       +--:(address)
```

```
        |  +--rw acl-mac:dest-address              yang:mac-address
        |  +--rw acl-mac:dest-address-mask         yang:mac-address
        +--:(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>
      <mac-source-kind>mac</mac-source-kind>
      <source-address>0180.c200.000</source-address>
      <source-address-mask>0000.0000.0000</source-address-mask>
      <mac-dest-kind>any</mac-dest-kind>
    </filters>
```


4.3.4. IP-ADDRESS-AND-MASK grouping

```
      IP-ADDRESS-AND-MASK
         +--rw ip-address-kind      IP-Network-Kind
         +--rw (mask-or-host)?
           +--:(mask)
            |  +--rw ip-address       inet:ip-address
            |  +--rw ip-mask?         inet:ip-prefix
           +--:(host)
            |  +--rw (ip-host-address)?    inet:host
```

IP-ADDRESS-AND-MASK is a reusable grouping. It has four leaves: ip-address-kind, ip-source-address, ip-source-mask and ip-host-address to express the three kinds network.

any network.

```
    <ip-address-kind>any</ip-address-kind>
```

single host network.

```
<ip-address-kind>host</ip-address-kind>
<ip-host-address>my-host</ip-host-address>
```

host address with a mask.

```
<ip-address-kind>mac</ip-address-kind>
<ip-address>192.168.1.0</ip-address>
<ip-mask>255.255.255.0</ip-mask>
```

4.3.5. IP-SOURCE-NETWORK grouping

```
IP-SOURCE-NETWORK
   +--rw ip-source-kind      IP-Network-Kind
   +--rw (source-address-host-group)?
     +--:(address)
      |  +--rw ip-source-address       inet:ip-address
      |  +--rw ip-source-mask?       inet:ip-prefix
     +--:(host-name)
      |  +--rw (ip-source-host-address)?    inet:host
     +--:(group)
        +--rw ip-source-group?      IP-Address-Group-Ref
```

IP-SOURCE-NETWORK is a reusable grouping. It has four leaves: ip-address-kind, ip-source-address, ip-source-mask, ip-host-address, and ip-source-group to express the four kinds network. The following are valid instances:

ip-source-address is an ip address:

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

ip-source-address is 'any':

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

ip-source-address is an 'host' with host-name:

```
<ip-source-kind>host</ip-source-kind>
<ip-source-host-address>switch1</ip-source-host-address>
```

```
ip-source-address is an 'host' with host-name:
<ip-source-kind>host</ip-source-kind>
<ip-source-host-address>192.168.1.2</ip-source-host-address>

ip-source-address is an 'group':
<ip-source-kind>group</ip-source-kind>
<ip-source-group>Email-Server-IPV4</ip-source-group>
```

4.3.6. IP-DESTINATION-NETWORK grouping

```
IP-DESTINATION-NETWORK
   +--rw ip-dest-kind       IP-Network-Kind
   +--rw (dest-address-host-group)?
     +--:(address)
      |  +--rw ip-dest-address       inet:ip-address
      |  +--rw ip-dest-mask?       inet:ip-prefix
     +--:(host)
      |  +--rw (ip-dest-host-address)?    inet:ip-host
     +--:(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 IP-SOURCE-NETWORK and IP-DESTINATION-NETWORK grouping is to allow ip-source-address and ip-destination-address leaves appear in the same container. For example:

```
<filter>
  <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>
</filter>
```


4.4. Containers

4.4.1. acls Container

acls container contains a list of named acl. Each acl contains the following global leaves that applies to acl extensions. acl-arp, acl-mac, and acl-ip extends the list acl.

- name
- acl-type
- capture-session-id-global
- enable-match-counter-choices: The difference of these two choices is that enable-match-counter is to collect total match statistics for all aces verus enable-per-entry-match-counter is to collect match statistics for each ace.
- match

4.4.2. port-obj-grp Container

port-obj-grp allows to classify protocol port into groups. port-obj-grp container is a sequence of port ranges. One port-group can apply to multiple aces. This feature allows multiple ace applies to same port group. The following is Netconf XML examples of port-obj-grp and how it is refered from ace.

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

<port-obj-grp>
  <portgroup>
    <name>port-tunnel1</name>
    <groups>
      <group>
        <comparator>gt</comparator>
        <port-number> 8080</port-number>
      </group>
     </groups>
  </portgroup>
</port-obj-grp>
```

4.4.3. timerange-obj-grp Container

timerange-obj-grp container is list of named entry. Each entry contains a time-ranges container. Each time-ranges is a sequence of time-range. Each time-range is a remark (comments for the time-range), or an absolute time for start or end or both, or a periodic time for start or end or both. remark is the comments for the time-range that will be kept in the device. Same time-range can be reused in different aces. The following is Netconf XML examples of time-range-entry and how it is referred from ace.

```
<timerange-obj-grp>
  <entry>
    <name>weekday</name>
      <time-ranges>
        <time-range>
          <sequence-num>10</sequence-num>
          <remark> email server maintainance</remark>
       </time-range>

       <time-range>
         <sequence-num>30</sequence-num>
         <periodic>
           <weekday>
             Monday Tuesday Wednesday Thursday Friday
           </weekday>
           <start> 21:00:00</start>
           <end> 24:00:00</end>
         </periodic>
       </time-range>
     </time-ranges>
  </entry>
</timerange-obj-grp>
```

4.4.4. ip-address-obj-grp Container

ip-address-obj-grp container is list of named ip-address-group. Each ip-address-group is a sequence of ip-address and mask pair or host and host-address pair. Each ipaddress-group can be referred from ace by name. The following is Netconf XML examples of ip-address-group and how it is referred from ace.

```
<ip-address-obj-grp>
  <ip-address-group>
    <name>HR-VM-IPV6</name>
    <ip-addresses>
```

```
        <ip-address>
```



```
          <sequence-num>10</sequence-num>
          <afi>ipv6</afi>
          <ip-address-kind>ip</ip-address-kind>
          <ip-address> FE80::B3FF:FE1E:8329</ip-address>
          <ip-mask>128</ip-mask>
        </ip-address>
        <ip-address>
          <sequence-num>20</sequence-num>
          <afi>ipv6</afi>
          <ip-address-kind>host</ip-address-kind>
          <host-address>21DA::2F3B:2AA:FF:FE28:9C5A</host-address>
        </ip-address>
      </ip-addresses>
    </ip-address-group>

    <ip-address-group>
      <name>Email-Server-IPV4</name>
      <ip-addresses>
        <ip-address>
          <sequence-num>10</sequence-num>
          <ip-address-kind>ip</ip-address-kind>
          <ip-address>128.107.0,0</ip-address>
          <ip-mask>255.255.0.0</ip-mask>
        </ip-address>
        <ip-address>
          <sequence-num>20</sequence-num>
          <ip-address-kind>ip</ip-address-kind>
          <ip-address>139.207.0.0</ip-address>
          <ip-mask>255.255.0.0</ipmask>
        </ip-address>
      </ip-addresses>
    </ip-address-group>
  </ip-address-obj-grp>

  <ip-ace>
    <sequence-num>100</sequence-num>
    <afi>ipv6</afi>
    <action>permit</action>
    <options>
      <ip-source-kind>group</ip-source-kind>
```

```
      <ip-source-group>HR-VM-IPV6</ip-source-group>
      <ip-dest-address>any</ip-dest-address>
    </options>
  </ip-ace>
```


5. acl-ip module

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

5.1. Groupings

5.1.1. ACTIONS Grouping

ACTIONS grouping is a reusable grouping for ace action when there is a match. It has the following leaves:

- action: a mandatory leaf when ace filters match and support permit and deny two actions now.
- log-choice: enable log when ace filters match. Some device support log-input to have additional input interface in the log message when if-feature is enabled.

5.1.2. SRC-PORT-GROUP grouping

```
      SRC-PORT-GROUP
        +-- (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
           +-- :(by-name)
              +--src-port-group-name
```

SRC-PORT-GROUP is a reusable group and allow the following three ways to define a group of ports.

- port-number-range: use src-port-lower and src-port-upper two leaves to specify a port range. src-port-lower has to less or equal than src-port-upper.

- port-number: use comparator and src-port two leaves to specify a port range. See Comparator typedef in the model for the possible values for comparator leaf.

- port range ref: refer to a named port group defined using port-obj-grp. For example:


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

5.1.3. DES-PORT-GROUP grouping

```
      DES-PORT-GROUP
        +-- (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
```

DES-PORT-GROUP is a reusable group and has a similar structure as SRC-PORT-GROUP. The reason to have both SRC-PORT-GROUP and DES-PORT-GROUP is to allow src-port and des-port in the same container..

```
<filters>
        <src-port-lower>80</src-port-lower>
        <src-portupper>8080</src-port-upper>
        <des-port-lower>80</des-port-lower>
```

```
            <des-port-upper>8080</des-port-upper>
    </filters>
```

5.1.4. IP-ACE-FILTERS Grouping


```
       IP-ACE-FILTERS
         +--rw acl-ip:protocol?                    c-types:IP-Protocol
         +--rw acl-ip:enable-capture?              boolean
         +--rw acl-ip:capture-session-id?          uint32
         +--rw acl-ip:fragments?                   empty
         +--rw acl-ip:time-range?                  acl:Time-Range-Ref
         +--SRC-PORT-GROUP
         +  DEST-PORT-GROUP
         +--rw (packet-length-or-range)?
         |  +--:(length)
         |  |  +--rw acl-ip:packet-length-comparator    acl:Comparator
         |  |  +--rw acl-ip:packet-length               uint32
         |  +--:(range)
         |     +--rw acl-ip:packet-length-upper         uint32
         |     +--rw acl-ip:packet-length-lower         uint32
         +--rw acl-ip:tcp-flag-value?              c-types:TCP-Flag-Type
         +--rw acl-ip:tcp-flag-mask?               c-types:TCP-Flag-Type
         +--rw acl-ip:tcp-flag-operation?          enumeration
```

IP-ACE-FILTERS defines the following leaves that reused by ipv4 and ipv6 ace.

- protocol
- enable-capture: 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.
- capture-session-id: This optional leaf only applies when enable-capture is true.
- fragments: when presence, it match the non-initial fragment.
- time-range: time-range is to enable packet capture on this filter for a timerange-group by name. time-range is Time-Range-Ref type which is a leafref.
- SRC-PORT-GROUP: see Section 5.1.2
- DES-PORT-GROUP: see Section 5.1.3
- packet-length-or-range: packet-length-or-range is to match package if the pack length is in the range. It allows two ways to specify packet length range.

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

- tcp-flag-value
- tcp-flag-mask

- o tcp-flag-operation

- o tcp-flag-value, tcp-flag-mask and tcp-flag-operation combinations is to 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>
```

5.1.5. IPV4-ACE-GROUP Grouping

The following is IPV4-ACE-GROUP tree. IPV4-ACE-GROUP grouping is used in acl list when acl-type is ip-acl. It is not reused in this module but to keep acl list data model concise, so the acl list structure won't be submerged with ACE details. For the definition of acl list, see section acl list Section 4.4.1.


```
  +--:IPV4-ACE-GROUP
  +--rw ipv4-aces
     +--rw ipv4-ace [sequence-num]
      +--rw sequence-num  Sequence-Number
      +--rw (remark-or-filter)?
```

```
          +--:(remark)
          |  +--rw remark?     string
          +--:(packet-filter)
           +--rw filter
           |  +--rw acl-ip:filters
           |  |  +--acl:IP-SOURCE-NETWORK
           |  |  +--acl:IP-DESTINATION-NETWORK
           |  |  +--IP-ACE-FILTERS
           |  |  +--rw (dscp-or-tos)?
           |  |    +--:(dscp)
           |  |    |  +--rw dscp?              inet:dscp
           |  |    +--:(tos)
           |  |      +--rw tos?            c-types:ToS
           |  |      +--rw precedence?         c-types:Precedence
           |  +--rw actions
           |  |  +--ACTIONS
           |  +--ro match?   uint64
```

IPV4-ACE-GROUP contains a sequenced ipv4-ace.

5.1.5.1. ipv4-ace list

An ipv4-ace list is set of rules that is used to filter ipv4 traffic on a networking device ordered by sequence-num. An ipv4-ace can be a remark or a-filter. A remark is a comment for the ace that will be keptin the device. A-filter consists of two parts:

- Filters: this specifies a set of matching criteria that a packet must satisfy for the rule to be applied.
- Actions: this specifies what to do with the packet when the filters are matched, for example, to drop the packet.

Here are the filters:

- IP-SOURCE-NETWORK: see Section 4.3.5.
- IP-DESTINATION-NETWORK: Section 4.3.6
- IP-ACE-FILTERS: see Section 5.1.4

o dscp-or-tos: Section 5.1.5.1.1

5.1.5.1.1. dscp-or-tos choice

dscp-or-tos is to match package on the dscp or tos value.

case dscp: dscp is to match packet on the dscp value.

case tos:This is to match packet on tos and precedence value. ToS and Precedence typedef is defined in common-types which can be deprecated when IETF has a standard ToS and Precedence YANG definition.

5.1.5.1.2. match leaf

match: this is an operational data. It maintains the packet matches (hits) on the ace.

5.1.6. IPV6-ACE-GROUP Grouping

The following is IPV6-ACE-GROUP tree. IPV6-ACE-GROUP grouping is used in acl list when acl-type is ip-acl. It is not reused in this module but to keep acl list data model concise, so the acl list structure won't be submerged with ACE details. For the definition of acl list, see section acl list Section 4.4.1.

```
  +--:IPV6-ACE-GROUP
  +--rw ipv6-aces
     +--rw ipv6-ace [sequence-num]
      +--rw sequence-num  Sequence-Number
      +--rw (remark-or-filter)?
       +--:(remark)
       |  +--rw remark?     string
       +--:(packet-filter)
        +--rw filter
        |  +--acl:IP-SOURCE-NETWORK
        |  +--acl:IP-DESTINATION-NETWORK
        |  +--IP-ACE-FILTERS
        |  +--rw (dscp-or-tos)?
        |  |  +--:(dscp)
        |  |  |  +--rw dscp?             inet:dscp
        |  +--rw igmp-type?
        |  +--rw actions
        |  |  +--ACTIONS
        +--ro match?   uint64
```

IPV6-ACE-GROUP contains a sequenced ipv6-ace.

5.1.6.1. ipv6-ace list

An ip-ace list is a set of rules that is used to filter ipv4 traffic on a networking device ordered by sequence-num. . An ip-ace can be a remark or an ace-filter. A remark is a comment for the ace that will be keptin the device. Same as ipv4-ace, a filter consists of two parts:

- Filters: this specifies a set of matching criteria that a packet must satisfy for the rule to be applied.
- Actions: this specifies what to do with the packet when the filters are matched, for example, to drop the packet.

Here are the filters:

IP-SOURCE-NETWORK: see Section 4.3.5.

IP-DESTINATION-NETWORK: Section 4.3.6

--IP-ACE-FILTERS: see Section 5.1.4

dscp-or-tos: Section 5.1.6.1.1

igmp-type

5.1.6.1.1. dscp-or-tos choice

Different from ipv4, ipv6 dscp-or-tos is to match package on the dscp value.

case dscp: dscp is to match packet on the dscp value.

5.1.6.1.2. match leaf

match: this is an operational data. It maintains the packet matches (hits) on the ace.

5.2. augment

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

afi

IPV6-ACE-GROUP

IPV4-ACE-GROUP

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>gragment_test1</name>
    <afi>ipv4</afi>
    <acl-type>ip-acl</acl-type>
    <ip-aces>
        <sequence-num>10</sequence-num>
        <actions>
            <action>permit</action>
        </actions>
        <filters>
            <ip-source-address>1.1.1.1</ip-source-address>
            <ip-source-mask-length>16</ip-source-mask-length>
            <ip-dest-address>any</ip-dest-address>
        </filters>
    </ip-aces>
    <ip-aces>
        <sequence-num>20</sequence-num>
        <actions>
            <action>permit</action>
        </actions>
        <filters>
            <ip-source-address>2.2.2.2</ip-source-address>
            <ip-source-mask-length>16</ip-source-mask-length>
            <ip-dest-address>any</ip-dest-address>
             <fragments/>
        </filters>
```

```
            </ip-aces>
          </acl>
        </acls>
```

By taking all the leaves value out, the above yang can be express in the following lines:

```
        fragment_test1 ip-acl ipv4
        10 permit ip 1.1.1.1/16 any
        20 permit ip 2.2.2.2/16 any fragment.
```

The acl configuration with implicit ace in the device withwill be:


```
        fragment_test1 ip-acl ipv4
        10 permit ip 1.1.1.1/16 any
        11 permit ip 1.1.1.1/16 any fragment
        20 permit ip 2.2.2.2/16 any fragment.
        100 deny any any
        110 deny any any fragment
```

Notice three lines of configuration are implicit.

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

```
        <acls>
          <acl>
            <name>gragment_test2</name>
            <acl-type>ip-acl</acl-type>
            <global-gragments>deny-all</global-gragments>
            <afi>ipv4</afi>
            <ip-aces>
                <sequence-num>10</sequence-num>
```

```
                <actions>
                    <action>permit</action>
                </actions>
                <filters>
                    <ip-source-address>1.1.1.1</ip-source-address>
                    <ip-source-mask-length>16</ip-source-mask-length>
                    <ip-dest-address>any</ip-dest-address>
                </filters>
            </ip-aces>
            <ip-aces>
                <sequence-num>20</sequence-num>
                <actions>
                    <action>permit</action>
                </actions>
                <filters>
                    <ip-source-address>2.2.2.2</ip-source-address>
                    <ip-source-mask-length>16</ip-source-mask-length>
                    <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 leaves value out, the above yang can be express in the following lines:

```
fragment_test2 ip-acl ipv4 deny-all
10 permit ip 1.1.1.1/16 any
20 permit ip 2.2.2.2/16 any fragment.
```

```
fragment_test2 ip-acl ipv4
10 permit ip 1.1.1.1/16 any
20 permit ip 2.2.2.2/16 any fragment.
100 deny any any
110 deny any any fragment
```

6. acl-mac module

6.1. MAC-ACE-GROUP group

The following is MAC-ACE-GROUP tree. MAC-ACE-GROUP grouping is used in acl list when acl-type is mac-acl. It is not reused in this module but to keep acl list data model concise, so the acl list structure won't be submerged with ACE details. For the definition of acl list, see section acl list Section 4.4.1.

```
    MAC-ACE-GROUP
    +--rw mac-aces
      +--rw mac-ace [sequence-num]
       +--rw sequence-num  Sequence-Number
       +--rw (remark-or-ace-filter)?
        +--:(remark)
        |  +--rw remark      Remark
        +--:(ace-filter)
           +--rw filter
           |  +--acl:MAC-SOURCE-NETWORK
           |  +--acl:MAC-DESTINATION-NETWORK
           |  +--rw ethertype?          c-types:Ether-Type
           |  +--rw ethertype-mask?       c-types:Ether-Type
           |  +--rw enable-capture?     boolean
           |  +--rw capture-session-id?     uint8
           |  +--rw cos?            c-types:CoS
           |  +--rw time-range?         Time-Range-Ref
           |  +--rw vlan?             c-types:Vlan-Identifier
           +-- rw actions
           |  +--rw action      ACL-Action
           |  +--rw log?          boolean
           +--ro match?   uint64
```


MAC-ACE-GROUP has a mac-aces container. The mac-aces container contains a list of sequenced mac-ace.

6.1.1. mac-ace list

A mac-ace list is a set of rules ordere by sequence-num that is used to filter traffic on a networking device using information in layer 2

header. A mac-ace can be a remark or an ace-filter. A remark is a comment for the ace that will be keptin the device. An ace-filter consists of two parts:

- Filters: this specifies a set of matching criteria that a packet must satisfy for the rule to be applied.

- Actions: this specifies what to do with the packet when the filters are matched, for example, to drop the packet.

Here are the filters:

MAC-SOURCE-NETWORK: see Section 4.3.2.

MAC-DESTINATION-NETWORK: Section 4.3.3

ethertype-mask: see definition in "acl-ip" module.

cos

time-range:

vlan

enable-capture

capture-session-id

The definitions and descriptions of the above leaves can be found in the text of the module "acl-mac"

6.2. augment

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

7. acl-arp module

7.1. ARP-ACE-GROUP group

The following is ARP-ACE-GROUP. ARP-ACE-GROUP. grouping is used in acl list when acl-type is arp-acl. It is not reused in this module but to keep acl list data model concise, so the acl list structure won't be submerged with ACE details. For the definition of acl list, see section acl list Section 4.4.1. ARP-ACE-GROUP has a arp-aces container. The arp-aces container contains a sequenced arp-ace.

```
    +--:ARP-ACE-GROUP
    +--rw arp-aces
       +--rw arp-ace [sequence-num]
         +--rw sequence-num  Sequence-Number
         +--rw (remark-or-ace-filter)?
         +--:(remark)
          |  +--rw remark     string
         +--:(ace-filter)
          | +--rw filters
          |   +--rw direction?         enumeration
          |   +--rw acl:IP-SOURCE-NETWORK
          |   +--rw acl:IP-DESTINATION-NETWORK
          |   +--rw acl:MAC-SOURCE-NETWORK
          |   +--rw acl:MAC-DESTINATION-NETWORK
          |   +--rw enable-capture?      boolean
          |   +--rw capture-session-id?      uint32
         +--rw actions
          |   +--rw action        ACL-Action
          |   +--rw log?        boolean
           +--ro match?    uint64
```

7.2. arp-ace list

An arp-ace list is a set of rules ordere by sequence-num that is used to filter arp traffic on a networking device. A arp-ace can be a remark or an ace-filter. A remark is a comment for the ace that will be keptin the device. An ace-filter consists of two parts:

- Filters: this specifies a set of matching criteria that a packet must satisfy for the rule to be applied.

- Actions: this specifies what to do with the packet when the filters are matched, for example, to drop the packet.

Here are the filters

direction: it is the comments for the ace that will be kept in the device.

IP-SOURCE-NETWORK: see Section 4.3.5

IP-DESTINATION-NETWORK: Section 4.3.6

MAC-SOURCE-NETWORK: see Section 4.3.2.

MAC-DESTINATION-NETWORK: Section 4.3.3

enable-capture

capture-session-id

The definitions and descriptions of the above leaves can be found in the text of the module "acl-arp".

7.3. augment

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

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 [sequence-num]
|  |     +--rw acl-ip:sequence-num
```



```
|  |     +--rw (remark-or-filter)?
|  |        +--:(remark)
|  |        |  +--rw acl-ip:remark?
|  |        +--:(filter)
|  |           +--rw acl-ip:filters
|  |              |  +--rw acl-ip:ip-source-kind
|  |              |  +--rw (source-address-host-group)?
|  |              |  |  +--:(mask)
|  |              |  |  |  +--rw acl-ip:ip-source-address?
|  |              |  |  |  +--rw acl-ip:ip-source-mask
|  |              |  |  +--:(host-address)
|  |              |  |  |  +--rw acl-ip:ip-source-host-address
|  |              |  |  +--:(group)
|  |              |  |     +--rw acl-ip:ip-source-group
|  |              |  +--rw acl-ip:ip-dest-kind IP-Network-Kind
|  |              |  +--rw (dest-mask-host-group)?
|  |              |  |  +--:(mask)
|  |              |  |  |  +--rw acl-ip:ip-dest-address?
|  |              |  |  |  +--rw acl-ip:ip-dest-mask?
|  |              |  |  +--:(host-address)
|  |              |  |  |  +--rw acl-ip:ip-dest-host-address?
|  |              |  |  +--:(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 (des-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 (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 (dscp-or-tos)?
|  |           |  |  +--:(dscp)
|  |           |  |     +--rw acl-ip:dscp?
|  |           |  +--rw acl-ip:igmp-type?
|  |           +--rw acl-ip:actions
|  |           |  +--rw acl-ip:action
|  |           |  +--rw (log-choice)?
|  |           |     +--:(log)
|  |           |     |  +--rw acl-ip:log?
|  |           |     +--:(log-input)
|  |           |        +--rw acl-ip:log-input?
|  |           +--ro acl-ip:match?
|  +--rw acl-ip:ipv4-aces
|  |  +--rw acl-ip:ipv4-ace [sequence-num]
|  |     +--rw acl-ip:sequence-num
|  |     +--rw (remark-or-filter)?
|  |        +--:(remark)
|  |        |  +--rw acl-ip:remark
|  |        +--:(filter)
|  |           +--rw acl-ip:filters
|  |           |  +--rw acl-ip:ip-source-kind
```

```
|  |           |  +--rw (source-address-host-group)?
|  |           |  |  +--:(mask)
|  |           |  |  |  +--rw acl-ip:ip-source-address?
|  |           |  |  |  +--rw acl-ip:ip-source-mask
|  |           |  |  +--:(host-address)
|  |           |  |  |  +--rw acl-ip:ip-source-host-address
|  |           |  |  +--:(group)
|  |           |  |     +--rw acl-ip:ip-source-group
|  |           |  +--rw acl-ip:ip-dest-kind
|  |           |  +--rw (dest-mask-host-group)?
|  |           |  |  +--:(mask)
|  |           |  |  |  +--rw acl-ip:ip-dest-address?
|  |           |  |  |  +--rw acl-ip:ip-dest-mask?
|  |           |  |  +--:(host-address)
|  |           |  |  |  +--rw acl-ip:ip-dest-host-address?
|  |           |  |  +--:(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 (des-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 (packet-length-or-range)?
```

```
|  |  |           |  |  +--:(length)
|  |  |           |  |  |  +--rw acl-ip:packet-length-comparator
|  |  |           |  |  |  +--rw acl-ip:packet-length   uint32
|  |  |           |  |  +--:(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 (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
|  |  |           |  +--rw (log-choice)?
|  |  |           |     +--:(log)
|  |  |           |     |  +--rw acl-ip:log?
|  |  |           |     +--:(log-input)
|  |  |           |        +--rw acl-ip:log-input?
|  |  |           +--ro acl-ip:match?
|  +--rw acl-ip:global-fragments?
|  +--rw acl-mac:mac-aces
|  |  +--rw acl-mac:mac-ace [sequence-num]
```



```
|  |     +--rw acl-mac:sequence-num
|  |     +--rw (remark-or-ace-filter)
|  |        +--:(remark)
|  |        |  +--rw acl-mac:remark
|  |        +--:(filter)
|  |           +--rw acl-mac:filters
|  |           |  +--rw acl-mac:mac-source-kind?
|  |           |  +--rw (source-network)?
|  |           |  |  +--:(address)
|  |           |  |  |  +--rw acl-mac:source-address
|  |           |  |  |  +--rw acl-mac:source-address-mask
|  |           |  |  +--:(host)
|  |           |  |     +--rw acl-mac:source-host-name
|  |           |  +--rw acl-mac:mac-dest-kind?
|  |           |  +--rw (dest-network)?
```

```
|  |           |  |  +--:(address)
|  |           |  |  |  +--rw acl-mac:destination-address
|  |           |  |  |  +--rw acl-mac:destination-address-mask
|  |           |  |  +--:(host)
|  |           |  |     +--rw acl-mac:dest-host-name
|  |           |  +--rw acl-mac:ethertype?
|  |           |  +--rw acl-mac:ethertype-mask?
|  |           |  +--rw acl-mac:cos?
|  |           |  +--rw acl-mac:time-range?
|  |           |  +--rw acl-mac:vlan?
|  |           |  +--rw acl-mac:enable-capture?
|  |           |  +--rw acl-mac:capture-session-id?
|  |           +--rw acl-mac:actions
|  |           |  +--rw acl-mac:action
|  |           |  +--rw acl-mac:log?
|  |           +--ro acl-mac:match?
|  +--rw acl-arp:arp-aces
|     +--rw acl-arp:arp-ace [sequence-num]
|        +--rw acl-arp:sequence-num
|        +--rw (remark-or-ace-filter)?
|           +--:(remark)
|           |  +--rw acl-arp:remark
|           +--:(filter)
|              +--rw acl-arp:filters
|              |  +--rw acl-arp:direction?
|              |  +--rw acl-arp:ip-source-kind
|              |  +--rw (source-address-host-group)?
|              |  |  +--:(mask)
|              |  |  |  +--rw acl-arp:ip-source-address?
|              |  |  |  +--rw acl-arp:ip-source-mask
|              |  |  +--:(host-address)
|              |  |  |  +--rw acl-arp:ip-source-host-address
|              |  |  +--:(group)
```



```
|              |  |     +--rw acl-arp:ip-source-group
|              |  +--rw acl-arp:ip-dest-kind
|              |  +--rw (dest-mask-length-host-group)?
|              |  |  +--:(mask)
|              |  |  |  +--rw acl-arp:ip-dest-address?
|              |  |  |  +--rw acl-arp:ip-dest-mask?
|              |  |  +--:(host-address)
|              |  |  |  +--rw acl-arp:ip-dest-host-address?
```

```
|              |  |  +--:(group)
|              |  |     +--rw acl-arp:ip-dest-group?
|              |  +--rw acl-arp:mac-source-kind?
|              |  +--rw (source-network)?
|              |  |  +--:(address)
|              |  |  |  +--rw acl-arp:source-address
|              |  |  |  +--rw acl-arp:source-address-mask
                                      yang:mac-address
|              |  |  +--:(host)
|              |  |     +--rw acl-arp:source-host-name
|              |  +--rw acl-arp:mac-dest-kind?
|              |  +--rw (dest-network)?
|              |  |  +--:(address)
|              |  |  |  +--rw acl-arp:destination-address
|              |  |  |  +--rw acl-arp:destination-address-mask
|              |  |  +--:(host)
|              |  |     +--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-obj-grp
|  +--rw port-group [name]
|     +--rw name      string
|     +--rw groups
|        +--rw groups [sequence-num]
|           +--rw sequence-num
|           +--rw (port-number-or-range)?
|              +--:(port-number-range)
|              |  +--rw port-lower
|              |  +--rw port-upper
|              +--:(port-number)
|                 +--rw comparator
|                 +--rw port
+--rw timerange-obj-grp
|  +--rw timerange-group [name]
|     +--rw name           string
|     +--rw time-ranges
```



```
|        +--rw time-range [sequence-num]
```

```
|           +--rw sequence-num
|           +--rw (range-type)?
|              +--:(remark)
|              |  +--rw remark?
|              +--:(absolute)
|              |  +--rw absolute
|              |     +--rw start?
|              |     +--rw end?
|              +--:(periodic)
|                 +--rw periodic
|                    +--rw weekdays?
|                    +--rw start?
|                    +--rw end?
|
|     +--rw ip-address-obj-grp
|        +--rw ip-address-group [name]
|           +--rw name
|           +--rw afi?
|           +--rw ip-addresses
|              +--rw ip-address [sequence-num]
|                 +--rw sequence-num
|                 +--rw ip-address-kind?
|                 +--rw (mask-or-host)?
|                    +--:(mask)
|                    |  +--rw ip-address?
|                    |  +--rw ip-mask
|                    +--:(host)
|                       +--rw ip-host-address
rpcs:
   +---x reset-match
      +--ro input
         +--ro name            string
         +--ro sequence-num?   Sequence-Number
module: acl-ip
module: acl-mac
module: acl-arp
```

Figure 3

9. ACL YANG Module

This module imports type definitions from [RFC6021].

<CODE BEGINS> file "acl@2012-08-30.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";

    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
";

```
    revision 2012-08-31 {
        description "Initial revision. ";
    }


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



```
    }

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

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

    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";
    }

    feature match-counter {
        description
            "The ability to maintain global or local match statistics
            for each ACL ruels.";
```

```
    }

    feature port-group {
        description
            "The ability to define named groups for lists of ports. ";
    }
    feature ip-group-address {
        description
            "The ability to define named groups for lists of
            ip addresses. ";
    }
    feature arp-acl {
        description "The ability to support ARP ACLs.";

    }
    feature capture-session-id {
```



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

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

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

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

/* Types */


typedef 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 permit {
                description "Apply permit action to the traffic";
            }
            enum deny {
                description "Apply deny action to the traffic";
            }
        }

    }

    typedef IP-Network-Kind {
        description "A enumeration to indicate the network to be
            expressed as an IP address and mask
            or simply identify a system as a host,
            or as a member of a pre-defined address group,
            or as any system.";
        type enumeration {
            enum ip {
                description "Used with address and mask couple
                    to express network.
                    ";
            }
            enum any {
                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'";
            }
            enum 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'";
            }
            enum group {
                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.";
            }
        }
    }

    typedef MAC-Network-Kind {
        description
            "A enumeration data type to express the different ways
            to express network or address ";
        type enumeration {
            enum mac {
                description "Used with address and mask couple
                    to express network.";
            }
            enum any {
                description "To express Any network or address";
            }
            enum host {
                description " Use the host address
                    combination as an abbreviation for an address
                    and wildcard of address 0.0.0.0";
            }
        }
    }

    typedef Sequence-Number {
        description
            "An data type based on uint32 to determine the order
            in which the statements within the access list will
            be evaluated.";
        type uint32 {
            range "1..2147483646";
```



```
        }

    }
```

```
typedef Remark {
    type string {
        length "0..100";
    }
}

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";
    }
}

typedef Port-Group-Ref {
    description
        "This type is used to refer to a Portgroup object.";
    type leafref {
        path "/acls/port-obj-grp/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-obj-grp/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-obj-grp/timerange-group/name";
```

```
        }

    }

    grouping PORT-GROUP {
        description
            "Using comparator and port, or port-upper and port-lower
            to specify a port range. ";
        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 Comparator;
                    mandatory true;
                }
                leaf port {
                    type inet:port-number;
                    description "Port number.";
                    mandatory true;
                }
            }
        }
    }
```

```
     /*
      * MAC-SOURCE-ADDRESS and MAC-DESTINATION-ADDRESS
      */
     grouping MAC-SOURCE-NETWORK {
         description "MAC address and mask pair for source.";
         leaf mac-source-kind {
             type MAC-Network-Kind ;
         }
         choice source-network {
             when "mac-source-kind != 'any'" ;
             case address {
                     when "mac-source-kind = 'mac'" ;
                 leaf source-address {
                     type yang:mac-address;
                     mandatory true;
                     description "A source MAC address.";
                 }
                 leaf source-address-mask {
                     type yang:mac-address;
                     mandatory true;
                 }
             }
            case host {
                 when "mac-source-kind = 'host'" ;
                 leaf source-host-name {
                     type inet:host;
                     mandatory true;
                 }
             }
         }
     }

     grouping MAC-DESTINATION-NETWORK {
         description "MAC address and mask pair for destination.";
         leaf mac-dest-kind {
             type MAC-Network-Kind ;
         }
         choice dest-network {
             when "mac-dest-kind != 'any'" ;
             case address {
```

```
                    when "mac-dest-kind = 'mac'" ;
                leaf dest-address {
                    type yang:mac-address;
                    mandatory true;
                    description "A destination MAC address";
                }
                leaf dest-address-mask {
                    type yang:mac-address;
```



```
                    mandatory true;
                }
            }
           case host {
                when "mac-dest-kind = 'host'" ;
                leaf dest-host-name {
                    type inet:host;
                    mandatory true;
                }
            }
        }
    }


    grouping IP-ADDRESS-AND-MASK {
        description "Reusable IP address and mask pair.";
        leaf ip-address-kind {
            type IP-Network-Kind ;
        }
        choice mask-or-host {
            when "ip-address-kind != 'any'";
            case mask {
                when "ip-address-kind = 'ip'" ;
                leaf ip-address {
                    type inet:ip-address;
                }
                leaf ip-mask {
                    type inet:ip-prefix;
                    mandatory true;
                }
            }
            case host {
                when "ip-address-kind = 'host'";
```

```
                leaf ip-host-address {
                    type inet:host;
                    mandatory true;
                }
            }
        }

    }

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

        leaf ip-source-kind {
            type IP-Network-Kind ;
            mandatory true;
```



```
        }
        choice source-address-host-group {
            when "ip-source-kind != 'any'";
            case mask {
                when "ip-source-kind = 'ip'" ;
                leaf ip-source-address {
                    type inet:ip-address;
                }
                leaf ip-source-mask {
                    type inet:ip-prefix;
                    mandatory true;
                }
            }
            case host {
                when "ip-source-kind = 'host'";
                leaf ip-source-host-address {
                    type inet:host;
                    mandatory true;
                }
            }
            case group {
                when "ip-source-kind = 'group'";
                leaf ip-source-group  {
                    type IP-Address-Group-Ref;
                    mandatory true;
                }
```

```
                if-feature ip-group-address;
            }

        }
    }

    grouping IP-DESTINATION-NETWORK {
        description
            "Reusable IP address and mask pair for destination.";
        leaf ip-dest-kind {
            type IP-Network-Kind ;
            mandatory true;
        }
        choice dest-address-host-group {
            when "ip-dest-kind != 'any'";
            case mask {
                when "ip-dest-kind = 'ip'" ;
                leaf ip-dest-address {
                    type inet:ip-address;
                }
                leaf ip-dest-mask {
                    type inet:ip-prefix;
```



```
                }
            }
            case host-address {
                when "ip-dest-address = 'host'";
                leaf ip-dest-host-address {
                    type inet:host;
                }
            }
            case group {
                when "ip-dest-address = 'group'";
                leaf ip-dest-group {
                    type IP-Address-Group-Ref;
                }
            }
        }
    }

    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 string;
            }

            leaf acl-type {
                type Acl-Type-Ref;
                description "Type of ACL";
                mandatory true;
            }
            leaf enable-capture-global{
                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 {
                when "../enable-capture-global = 'true'";
                type uint32 {
                    range "1..48";
                }
                if-feature capture-session-id;
                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
                            "Eanble 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 uint64;
            description
                "The total packet that have matched for the
                particular access list";
        }
    }


    container port-obj-grp {
        list port-group { //CLI: object-group ip port foogroup
            key "name";
            leaf name {
                type string;
            }
            container groups {
                list groups {
                    key "sequence-num";
                    leaf sequence-num {
                        type Sequence-Number;
                    }
                    //unique "comparator port-number
                    //port-lower port-upper";
                    uses PORT-GROUP;
                }
            }
```



```
        }

    }

    container timerange-obj-grp {
```

```
            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 string;
                    mandatory true;
                }
                container time-ranges {
                    list time-range {
                        key "sequence-num";
                        leaf sequence-num {
                            type Sequence-Number;
                            mandatory true;
                            description
                                "Sequence number. This number
                                determines the order of the statements
                                in the timerange list.
                                Range is 1 to 2147483646. ";
                        }

                        choice range-type {
                            case remark {
                                leaf remark {
                                    type string;
                                    description "The remark is the
                                        comments about each time range
                                        which is to make
                                        the time range entry easier for
                                        the network administrator to
                                        understand";
                                }
                            }
                            case absolute {
                                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";
                }

            }
        }
        case periodic {
            container periodic {
                description "To specify a periodic
                    time and date.";
                leaf 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;
                        }
                    }
                }
                leaf start {
                    type yang:timestamp;
                    description "Start time";
                }
                leaf end {
                    type yang:timestamp;
                    description "End time";
```

```
                                    }
                                }
                            }
                        }
                    }
                }
            }

        }
        container ip-address-obj-grp {
            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 string;
                }
                leaf afi {
                    default "ipv4";
                    type inet:ip-version;
                    description "Address Field Identifier (AFI).";
                }
                container ip-addresses {
                    list ip-address {
                        key "sequence-num";
                        leaf sequence-num {
                            type Sequence-Number;
                            mandatory true;
                            description
                                "Sequence number. This number
                                determines the order of the statements
                                 in the timerange list. Range is 1 to
                                 2147483646. ";

                        }

                        uses IP-ADDRESS-AND-MASK ;
                        //unique "ip-address ip-mask";
                        //unique "ip-host-address";
                    }
                }
```

```
            }
        }
    }
    rpc reset-match-counter {
        description "This RPC is to reset the statistics for the
            ACL matches.
```



```
            When name is the only input, it
            clears ACL match statistics and individual
            ACE match statistics.
            When name and sequence-numer are both inputs, it clears
            the ACE match statistics.";
        input {
            leaf name {
                type string;
                mandatory true;
                description "This name is the acl name.";
            }
            leaf sequence-num {
                type Sequence-Number;
                description
                    "This number is the ACE sequence-num.";
            }
        }
    }

 }
```

</CODE>

10. ACL-IP YANG Module

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

```
 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";


 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
    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
       ";

    revision 2012-08-31 {
        description "Initial revision. ";
```

```
    }

    grouping ACTIONS {
        leaf action {
            type acl:ACL-Action;
            mandatory true;
            description "Permit/deny action.";
        }
        choice log-choice {
            case log {
                leaf log {
                    type empty;
                    description
                        "Causes an informational
                        logging message about the
                        packet that matches the entry
                        to be sent to the console.";
                }
            }
            case log-input {
                leaf log-input {
                    type empty;
                    if-feature acl:log-input;
                    description
                        "Provides the same function
                        as the enable-log leaf,  except
                        that the logging message also
```



```
                        includes the input interface";
                }
            }
        }
    }


    grouping SRC-PORT-GROUP {
        description
            "Using comparator and port, or port-upper and port-lower
            to specify a port range. ";
        choice src-ports {
            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:Comparator;
                    mandatory true;
                }
                leaf src-port {
                    type inet:port-number;
                    description "Port number.";
                    mandatory true;
                }
            }
            case port-group-ref {
```



```
                if-feature acl:port-group;
                leaf src-port-group-name {
                    type acl:Port-Group-Ref;
                    mandatory true;
                    description
                        "Reference a port group by the Port
                        Group name.";
```

```
                }
            }
        }
    }

    grouping DES-PORT-GROUP {
        description "Using comparator and port, or port-upper and
            port-lower to specify a port range. ";
        choice des-ports {
            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:Comparator;
                    mandatory true;
                }
                leaf des-port {
                    type inet:port-number;
                    description "Port number.";
                    mandatory true;
                }
```

```
        }
        case port-group-ref {
            if-feature acl:port-group;
            leaf des-port-group-name {
                type acl:Port-Group-Ref;
                mandatory true;
                description
                "Reference a port group by the Port Group name.";
            }
        }
    }
}
grouping IP-ACE-FILTERS {
    leaf protocol {
        type c-types:IP-Protocol;
        description "IP protocol number.";

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

    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).";
    }
    uses SRC-PORT-GROUP {
         when
```

```
                "../protocol = '6' or
                ../protocol = '17' or
                ../protocol = '132'";
            description
                "Apply only when the protocol is TCP,
                UDP or SCTP.";
        }

        uses DES-PORT-GROUP {
             when
                "../protocol = '6' or
                ../protocol = '17' or
                ../protocol = '132'";
            description
                "Apply only when the protocol is TCP,
                UDP or SCTP.";
        }
        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: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 legth value for
                        operation gt, eq, etc, other
                        than range";
                    mandatory true;
                }
            }
            case range {
                leaf packet-length-upper {
                    type uint32 {
                        range "20..9210";
                    }
                    description "Upper Packet legth";
                    mandatory true;
                }

                leaf packet-length-lower {
                    type uint32 {
                        range "20..9210";
                    }
                    description "Lower packet length";
                    must
                        "number(../packet-length-lower) <=
                        number(../packet-length-upper)";
                    mandatory true;
                }
                description
                    "Packet operater 'range' takes
                    both lower and upper value.";
            }
        }
        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 acl:time-to-live;
            case value {
                leaf ttl-comparator {
                    type 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.";
```



```
                }
            }
        }
    }

    grouping IPV4-ACE-GROUP {
        description "Layer 3 Access Control Entry (ACE).";

        container ipv4-aces {
            description
                " The ip-aces container contains a list of ip-ace.
                Each ip-ace is made of a sequence number and a choice
                of remark(comment) or filter. The filter requires a
                mandatory action (permit/deny) and one or more options
                such as source-address with mask,ttl etc";

            list ipv4-ace {
                key "sequence-num";
                description "Layer 3 Access Control Element (ACE)";

                leaf sequence-num {
                    type acl:Sequence-Number;
```

```
                description "This number determines the order in
                    which the statements within the access list
                    will be evaluated.";
            }

            choice remark-or-filter {
                case remark {
                    leaf remark {
                        type acl:Remark;
                        mandatory true;
                        description "A remark is a comment that
                            can be inserted into an ACL in order
                            to make the access list easier for
                            the network administrator to
                            understand.
                            It is retained to facilitate
                            co-existence with  CLI.";
                    }
                }
                case filter {
                    container filters {

                        uses acl:IP-SOURCE-NETWORK;
                        uses acl:IP-DESTINATION-NETWORK;
                        uses IP-ACE-FILTERS ;
```



```
                        choice dscp-or-tos {
                            case dscp {
                                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";
                                    }
                                }
                            }
                            container actions {
                                uses ACTIONS ;
                            }

                            leaf match {
                                config false;
                                type uint64;
                                description "The total packet that have
                                    matched for the particular ACE";
                            }
                        }
                    }
                }

            }


        }
        grouping IPV6-ACE-GROUP {
            description "Layer 3 Access Control Entry (ACE).";
```



```
            container ipv6-aces {
                description
                    " The ip-aces container contains a list of ip-ace.
                    Each ip-ace is made of a sequence number and a
                    choice of remark(comment) or 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 "sequence-num";
                description "Layer 3 Access Control Element (ACE)";

                leaf sequence-num {
                    type acl:Sequence-Number;
                    description "This number determines the order in
                        which the statements within the access list
                        will be evaluated.";
                }

                choice remark-or-filter {
                    case remark {
                        leaf remark {
                            type string {
                                length "0..100";
                            }
                            description "The remark is the comments
                                about each acl which is to make the
                                access list easier for the network
                                administrator to understand.";
                        }
                    }
                    case filter {
                        container filters {

                            uses acl:IP-SOURCE-NETWORK;
                            uses acl:IP-DESTINATION-NETWORK;
                            uses IP-ACE-FILTERS ;

                            choice dscp-or-tos {
                                case dscp {
                                    leaf dscp {
                                        type inet:dscp;
                                        description
                                            "Match packets with given
                                            dscp value";
                                    }
                                }
                            }
```



```
                            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 {
                                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";
                                if-feature acl:flow-label;
                            }
                        }

                        container actions {
                            uses ACTIONS ;
                        }

                        leaf match {
                            config false;
                            type uint64;
                            description
                                "The total packet that have matched for
                                the particular ACE";
                        }
                    }
                }
            }

        }


    }


    augment "/acl:acls/acl:acl" {
```

```
        when "acl:acl-type = 'ip-acl'";
        leaf afi {
            type inet:ip-version ;
            default "ipv4";
        }
        uses IPV6-ACE-GROUP {
            when "../afi = 'ipv6'" ;
        }
        uses IPV4-ACE-GROUP {
            when "../afi = 'ipv4'" ;
        }

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

 }
```

</CODE>

11. ACL-MAC Configuration YANG Module

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

```
 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";
}

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";

 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
         ";

      revision 2012-08-31 {
          description "Initial revision. ";
      }

      /* Layer 2 ACL */

      grouping MAC-ACE-GROUP {
          description
              "Layer 2 Access Control Entry (ACE). The mac-aces
              container contains a list of mac-ace. Each mac-ace is
              comprised of a sequence number and a choice of remark
              (comment) or 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 sequence-num;
                  leaf sequence-num {
```

```
                type acl:Sequence-Number;
                description
                    "This number determines the order in which the
                    ace within the access list will be evaluated.";
            }
            choice remark-or-ace-filter {
                mandatory true;
                case remark {
                    leaf remark {
                        type acl:Remark;
                        mandatory true;
                        description
                            "A remark is a comment that can be
                            inserted into an ACL in order to make
                            the access list easier for the network
                            administrator to understand.
```



```
                            It is retained to facilitate
                            co-existence with CLI.";
                    }
                }
                case filter {
                    container filters {

                        uses acl:MAC-SOURCE-NETWORK;
                        uses acl:MAC-DESTINATION-NETWORK;

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

                        leaf 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..";
                            if-feature acl:ethertype-mask;
```

```
                    }

                    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 (Max name
                            string size 64).";
                    }

                    leaf vlan {
                        type c-types:Vlan-Identifier;
                        description "VLAN number";
                    }
                    leaf enable-capture {
                        type boolean ;
                        description
```



```
                            "Enable packet capture on this
                            filter for this session.";
                    }
                    leaf capture-session-id {
                        type uint8 {
                            range "1..48";
                        }
                        description
                            "Enable packet capture on this
                            filter for this session.";
                        if-feature acl:capture-session-id;
                    }
                }
                container actions {
                    leaf action {
                        type acl:ACL-Action;
                        mandatory true;
                        description "Permit/deny action.";
```

```
                            }

                            leaf log {
                                type boolean;
                                default false;
                                description
                                    "Causes an informational
                                    logging message about the
                                    packet that matches the entry
                                    to be sent to the console.";
                            }
                        }
                        leaf match {
                            config false;
                            type uint64;
                            description
                                "The total packets number that
                                have matched for the particular ACE";
                        }
                    }
                }
            }
        }

        augment "/acl:acls/acl:acl" {
            when "acl:acl-type = 'mac-acl'";
             uses MAC-ACE-GROUP;
        }
```



```
 }
```

</CODE>

12. ACL-ARP Configuration YANG Module

```
 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;
}

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";

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
    ";

  revision 2012-08-31 {
    description "Initial revision. ";
  }
  grouping ARP-ACE-GROUP {
    description "ARP Access Control Entry (ACE).";
    container arp-aces {
      list arp-ace {
        key "sequence-num";
        leaf sequence-num {
          type acl:Sequence-Number;
          mandatory true;
          description
            "This number determines the order in which the
            statements within the access list will be
            evaluated.";
        }
        choice remark-or-ace-filter {
          case remark {
            leaf remark {
              mandatory true;
              type acl:Remark;
              description
                "A remark is a comment that can be
                inserted into an ACL in order to make
                the access list easier for the network
                administrator to understand.
                It is retained to facilitate
                co-existence with CLI.";
            }
          }
          case filter {
            container filters {
              leaf direction {
                default "bi-direction";
                type enumeration {
```

```
                        enum bi-direction;
                        enum request;
                        enum response;
                    }
                    description "ARP request/response.";
                }

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

                uses acl:MAC-SOURCE-NETWORK;
                uses acl:MAC-DESTINATION-NETWORK {
                    when "../direction = 'response'";
                }
                leaf enable-capture {
                    type boolean ;
                    description
                        "Enable packet capture on this
                        filter for this session.";
                }
                leaf capture-session-id {
                    type uint32 {
                        range "1..48";
                    }
                    description
                        "Enable packet capture on this
                        filter for this session.
                        Session ID range is 1 to 48";
                }

            }
            container actions {
                leaf action {
                    type acl:ACL-Action;
                    mandatory true;
                    description "Permit/deny on match";
                }
                leaf log {
                    type boolean;
                    default false;
                    description "Log on match";
                }

            }
            leaf match {
```

```
                        config false;
```



```
                        type uint64;
                        description
                            "The total packet that have matched for
                            the particular ACE";
                    }
                }
            }
        }
    }

    augment "/acl:acls/acl:acl" {
        when "acl:acl-type = 'arp-acl'";
         uses ARP-ACE-GROUP;
    }
}
                </CODE>
```

13. COMMON-TYPES YANG Module

```
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

        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";

      description
        "This module contains a collection of generally useful
        YANG types that are required by YANG modules for the
```



```
        management of ACLs, yet not specific to ACLs.

        Terms and Acronyms

         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 2012-08-31 {
        description "Initial revision. ";
      }
      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{
        description "";
        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";
            }
        }
        reference "RFC 793 TRANSMISSION CONTROL PROTOCOL";
    }

    typedef Ether-Type {
        type string {
            pattern '0x[0-9a-fA-F]{4}';
        }
        description
                "EtherType 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)

bpdu-sap BPDU SAP encapsulated packets
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{
        type uint8 ;

        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]
            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]
         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{
```



```
        description "The TTL is an 8-bit field in IP header.
            The maximum TTL value is 255.";
        type uint8 {
            range "0..255";
        }
    }
 }
                    </CODE>
```

14. Security Considerations

.

15. Acknowledgements

We wish to acknowledge the contributions of Louis Fourie and Dana Blair to the design of this YANG data model.

16. 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