NETMOD WG
Internet-Draft
Intended status: Standards Track
Expires: April 4, 2019

M. Jethanandani
VMware
S. Agarwal
Cisco Systems, Inc.
L. Huang

D. Blair
October 1, 2018

Network Access Control List (ACL) YANG Data Model
draft-ietf-netmod-acl-model-20

Abstract

This document defines a data model for Access Control List (ACL). An ACL is a user-ordered set of rules, used to configure the forwarding behavior in device. Each rule is used to find a match on a packet, and define actions that will be performed on the packet.

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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on April 4, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
1.1. Definitions and Acronyms 4
1.2. Terminology 4
1.3. Tree Diagram 4
2. Problem Statement 4
3. Understanding ACL's Filters and Actions 5
3.1. ACL Modules 5
4. ACL YANG Models 9
4.1. IETF Access Control List module 9
4.2. IETF Packet Fields module 24
4.3. ACL Examples 37
4.4. Port Range Usage and Other Examples 39
5. Security Considerations 43
6. IANA Considerations 44
6.1. URI Registration 44
6.2. YANG Module Name Registration 44
7. Acknowledgements 45
8. References 45
8.1. Normative References 45
8.2. Informative References 47
Appendix A. Extending ACL model examples 48
A.1. A company proprietary module example 48
A.2. Linux nftables 51
A.3. Ethertypes 52
Authors' Addresses 60

1. Introduction

Access Control List (ACL) is one of the basic elements used to configure device forwarding behavior. It is used in many networking technologies such as Policy Based Routing (PBR), firewalls etc.

An ACL is an user-ordered set of rules, that is used to filter traffic on a networking device. Each rule is represented by an Access Control Entry (ACE).

Each ACE has a group of match criteria and a group of actions.

The match criteria allow for definition of packet headers and metadata, the contents of which must match the definitions.

- Packet header matches apply to fields visible in the packet such as address or Class of Service (CoS) or port numbers.

- In case a vendor supports it, metadata matches apply to fields associated with the packet but not in the packet header such as input interface or length of the packet as received over the wire.

The actions specify what to do with the packet when the matching criteria are met. These actions are any operations that would apply to the packet, such as counting, policing, or simply forwarding. The list of potential actions is unbounded depending on the capabilities of the networking devices.

Access Control List is also widely knowns as ACL (pronounce as [ak-uh l]) or Access List. In this document, Access Control List, ACL and Access List are used interchangeably.

The matching of filters and actions in an ACE/ACL are triggered only after the application/attachment of the ACL to an interface, VRF, vty/tty session, QoS policy, or routing protocols, amongst various other configuration attachment points. Once attached, it is used for filtering traffic using the match criteria in the ACEs and taking appropriate action(s) that have been configured against that ACE. In order to apply an ACL to any attachment point other than an interface, vendors would have to augment the ACL YANG model.

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. Please note that no other RFC Editor instructions are specified anywhere else in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements

- "XXXX" --> the assigned RFC value for this draft both in this draft and in the YANG models under the revision statement.

- Revision date in model, in the format 2018-10-01 needs to get updated with the date the draft gets approved. The date also needs to get reflected on the line with <CODE BEGINS>.

1.1. Definitions and Acronyms

ACE: Access Control Entry

ACL: Access Control List

CoS: Class of Service

DSCP: Differentiated Services Code Point

ICMP: Internet Control Message Protocol

IP: Internet Protocol

IPv4: Internet Protocol version 4

IPv6: Internet Protocol version 6

MAC: Media Access Control

PBR: Policy Based Routing

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Tree Diagram

For a reference to the annotations used in tree diagrams included in this draft, please see YANG Tree Diagrams [RFC8340].

2. Problem Statement

This document defines a YANG 1.1 [RFC7950] data model for the configuration of ACLs. It is very important that model can be used easily by application/attachment models.

ACL implementations in every device may vary greatly in terms of the filter constructs and actions that they support. Therefore this

draft proposes a model that can be augmented by standard extensions and vendor proprietary models.

3. Understanding ACL's Filters and Actions

Although different vendors have different ACL data models, there is a common understanding of what Access Control List (ACL) is. A network system usually has a list of ACLs, and each ACL contains an ordered list of rules, also known as Access Control Entries (ACE). Each ACE has a group of match criteria and a group of actions. The match criteria allow for definition of contents of the packet headers or metadata, if supported by the vendor. Packet header matching applies to fields visible in the packet such as address or CoS or port numbers. Metadata matching applies to fields associated with the packet, but not in the packet header, such as input interface, packet length, or source or destination prefix length. The actions can be any sort of operation from logging to rate limiting or dropping to simply forwarding. Actions on the first matching ACE are applied with no processing of subsequent ACEs.

The model also includes a container to hold overall operational state for each ACL and operational state for each ACE. One ACL can be applied to multiple targets within the device, such as interface of a networking device, applications or features running in the device, etc. When applied to interfaces of a networked device, distinct ACLs are defined for the ingress (input) or egress (output) interface.

This draft tries to address the commonalities between all vendors and create a common model, which can be augmented with proprietary models. The base model is simple in design, and we hope to achieve enough flexibility for each vendor to extend the base model.

The use of feature statements in the model allows vendors to advertise match rules they are capable and willing to support. There are two sets of feature statements a device needs to advertise. The first set of feature statements specify the capability of the device. These include features such as "Device can support matching on Ethernet headers" or "Device can support matching on IPv4 headers". The second set of feature statements specify the combinations of headers the device is willing to support. These include features such as "Plain IPv6 ACL supported" or "Ethernet, IPv4 and IPv6 ACL combinations supported".

3.1. ACL Modules

There are two YANG modules in the model. The first module, "ietf-access-control-list", defines generic ACL aspects which are common to all ACLs regardless of their type or vendor. In effect, the module

can be viewed as providing a generic ACL "superclass". It imports the second module, "ietf-packet-fields". The match container in "ietf-access-control-list" uses groupings in "ietf-packet-fields" to specify match fields such as port numbers or protocol. The combination of 'if-feature' checks and 'must' statements allow for the selection of relevant match fields that a user can define rules for.

If there is a need to define a new "matches" choice, such as IPFIX [RFC7011], the container "matches" can be augmented.

```
module: ietf-access-control-list
  +--rw acls
     +--rw acl* [name]
     |  +--rw name    string
     |  +--rw type?   acl-type
     |  +--rw aces
     |     +--rw ace* [name]
     |        +--rw name          string
     |        +--rw matches
     |        |  +--rw (l2)?
     |        |  |  +--:(eth)
     |        |  |     +--rw eth {match-on-eth}?
     |        |  |        +--rw destination-mac-address?
     |        |  |        |       yang:mac-address
     |        |  |        +--rw destination-mac-address-mask?
     |        |  |        |       yang:mac-address
     |        |  |        +--rw source-mac-address?
     |        |  |        |       yang:mac-address
     |        |  |        +--rw source-mac-address-mask?
     |        |  |        |       yang:mac-address
     |        |  |        +--rw ethertype?
     |        |  |                eth:ethertype
     |        |  +--rw (l3)?
     |        |  |  +--:(ipv4)
     |        |  |  |  +--rw ipv4 {match-on-ipv4}?
     |        |  |  |     +--rw dscp?
     |        |  |  |     |       inet:dscp
     |        |  |  |     +--rw ecn?
     |        |  |  |     |       uint8
     |        |  |  |     +--rw length?
     |        |  |  |     |       uint16
     |        |  |  |     +--rw ttl?
     |        |  |  |     |       uint8
     |        |  |  |     +--rw protocol?
     |        |  |  |     |       uint8
     |        |  |  |     +--rw ihl?
     |        |  |  |     |       uint8
```

```
         |        |  |  |     +--rw flags?
         |        |  |  |     |       bits
         |        |  |  |     +--rw offset?
         |        |  |  |     |       uint16
         |        |  |  |     +--rw identification?
         |        |  |  |     |       uint16
         |        |  |  |     +--rw (destination-network)?
         |        |  |  |     |  +--:(destination-ipv4-network)
         |        |  |  |     |     +--rw destination-ipv4-network?
         |        |  |  |     |             inet:ipv4-prefix
         |        |  |  |     +--rw (source-network)?
         |        |  |  |        +--:(source-ipv4-network)
         |        |  |  |           +--rw source-ipv4-network?
         |        |  |  |                   inet:ipv4-prefix
         |        |  |  +--:(ipv6)
         |        |  |     +--rw ipv6 {match-on-ipv6}?
         |        |  |        +--rw dscp?
         |        |  |        |       inet:dscp
         |        |  |        +--rw ecn?
         |        |  |        |       uint8
         |        |  |        +--rw length?
         |        |  |        |       uint16
         |        |  |        +--rw ttl?
         |        |  |        |       uint8
         |        |  |        +--rw protocol?
         |        |  |        |       uint8
         |        |  |        +--rw (destination-network)?
         |        |  |        |  +--:(destination-ipv6-network)
         |        |  |        |     +--rw destination-ipv6-network?
         |        |  |        |             inet:ipv6-prefix
         |        |  |        +--rw (source-network)?
         |        |  |        |  +--:(source-ipv6-network)
         |        |  |        |     +--rw source-ipv6-network?
         |        |  |        |             inet:ipv6-prefix
         |        |  |        +--rw flow-label?
         |        |  |                inet:ipv6-flow-label
         |        |  +--rw (l4)?
         |        |  |  +--:(tcp)
         |        |  |  |  +--rw tcp {match-on-tcp}?
         |        |  |  |     +--rw sequence-number?          uint32
         |        |  |  |     +--rw acknowledgement-number?   uint32
         |        |  |  |     +--rw data-offset?              uint8
         |        |  |  |     +--rw reserved?                 uint8
         |        |  |  |     +--rw flags?                    bits
         |        |  |  |     +--rw window-size?              uint16
         |        |  |  |     +--rw urgent-pointer?           uint16
         |        |  |  |     +--rw options?                  binary
         |        |  |  |     +--rw source-port
```

```
|        |  |  |     |  +--rw (source-port)?
|        |  |  |     |     +--:(range-or-operator)
|        |  |  |     |        +--rw (port-range-or-operator)?
|        |  |  |     |           +--:(range)
|        |  |  |     |           |  +--rw lower-port
|        |  |  |     |           |  |       inet:port-number
|        |  |  |     |           |  +--rw upper-port
|        |  |  |     |           |          inet:port-number
|        |  |  |     |           +--:(operator)
|        |  |  |     |              +--rw operator?     operator
|        |  |  |     |              +--rw port
|        |  |  |     |                      inet:port-number
|        |  |  |     +--rw destination-port
|        |  |  |        +--rw (destination-port)?
|        |  |  |           +--:(range-or-operator)
|        |  |  |              +--rw (port-range-or-operator)?
|        |  |  |                 +--:(range)
|        |  |  |                 |  +--rw lower-port
|        |  |  |                 |  |       inet:port-number
|        |  |  |                 |  +--rw upper-port
|        |  |  |                 |          inet:port-number
|        |  |  |                 +--:(operator)
|        |  |  |                    +--rw operator?     operator
|        |  |  |                    +--rw port
|        |  |  |                            inet:port-number
|        |  |  +--:(udp)
|        |  |  |  +--rw udp {match-on-udp}?
|        |  |  |     +--rw length?             uint16
|        |  |  |     +--rw source-port
|        |  |  |     |  +--rw (source-port)?
|        |  |  |     |     +--:(range-or-operator)
|        |  |  |     |        +--rw (port-range-or-operator)?
|        |  |  |     |           +--:(range)
|        |  |  |     |           |  +--rw lower-port
|        |  |  |     |           |  |       inet:port-number
|        |  |  |     |           |  +--rw upper-port
|        |  |  |     |           |          inet:port-number
|        |  |  |     |           +--:(operator)
|        |  |  |     |              +--rw operator?     operator
|        |  |  |     |              +--rw port
|        |  |  |     |                      inet:port-number
|        |  |  |     +--rw destination-port
|        |  |  |        +--rw (destination-port)?
|        |  |  |           +--:(range-or-operator)
|        |  |  |              +--rw (port-range-or-operator)?
|        |  |  |                 +--:(range)
|        |  |  |                 |  +--rw lower-port
|        |  |  |                 |  |       inet:port-number
```

```
|        |  |  |                |  +--rw upper-port
|        |  |  |                |          inet:port-number
|        |  |  |                +--:(operator)
|        |  |  |                   +--rw operator?     operator
|        |  |  |                   +--rw port
|        |  |  |                           inet:port-number
|        |  |  +--:(icmp)
|        |  |     +--rw icmp {match-on-icmp}?
|        |  |        +--rw type?             uint8
|        |  |        +--rw code?             uint8
|        |  |        +--rw rest-of-header?   binary
|        |  +--rw egress-interface?    if:interface-ref
|        |  +--rw ingress-interface?   if:interface-ref
|        +--rw actions
|        |  +--rw forwarding    identityref
|        |  +--rw logging?      identityref
|        +--ro statistics {acl-aggregate-stats}?
|           +--ro matched-packets?   yang:counter64
|           +--ro matched-octets?    yang:counter64
+--rw attachment-points
   +--rw interface* [interface-id] {interface-attachment}?
      +--rw interface-id    if:interface-ref
      +--rw ingress
      |  +--rw acl-sets
      |     +--rw acl-set* [name]
      |        +--rw name              -> /acls/acl/name
      |        +--ro ace-statistics* [name] {interface-stats}?
      |           +--ro name
      |           |       -> /acls/acl/aces/ace/name
      |           +--ro matched-packets?   yang:counter64
      |           +--ro matched-octets?    yang:counter64
      +--rw egress
         +--rw acl-sets
            +--rw acl-set* [name]
               +--rw name              -> /acls/acl/name
               +--ro ace-statistics* [name] {interface-stats}?
                  +--ro name
                  |       -> /acls/acl/aces/ace/name
                  +--ro matched-packets?   yang:counter64
                  +--ro matched-octets?    yang:counter64
```

4. ACL YANG Models

4.1. IETF Access Control List module

"ietf-access-control-list" module defines the "acls" container that has a list of "acl". Each "acl" has information identifying the access list by a name ("name") and a list ("aces") of rules

associated with the "name". Each of the entries in the list ("aces"), indexed by the string "name", has containers defining "matches" and "actions".

The model defines several ACL types and actions in the form of identities and features. Features are used by implementors to select the ACL types the system can support and identities are used to validate the types that have been selected. These types are implicitly inherited by the "ace", thus safeguarding against misconfiguration of "ace" types in an "acl".

The "matches" define criteria used to identify patterns in "ietf-packet-fields". The choice statements within the match container allow for selection of one header within each of "l2", "l3", or "l4" headers. The "actions" define behavior to undertake once a "match" has been identified. In addition to permit and deny for actions, a logging option allows for a match to be logged that can later be used to determine which rule was matched upon. The model also defines the ability for ACLs to be attached to a particular interface.

Statistics in the ACL can be collected for an "ace" or for an "interface". The feature statements defined for statistics can be used to determine whether statistics are being collected per "ace", or per "interface".

This module imports definitions from Common YANG Data Types [RFC6991], and A YANG Data Model for Interface Management [RFC8343].

<CODE BEGINS> file "ietf-access-control-list@2018-10-01.yang"

```
module ietf-access-control-list {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-access-control-list";
  prefix acl;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991 - Common YANG Data Types.";
  }

  import ietf-packet-fields {
    prefix pf;
    reference
    "RFC XXXX - Network ACL YANG Model.";
  }

  import ietf-interfaces {
```

```
    prefix if;
    reference
      "RFC 8343 - A YANG Data Model for Interface Management.";
  }

  organization
    "IETF NETMOD (Network Modeling Language)
     Working Group";

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

     Editor: Mahesh Jethanandani
             mjethanandani@gmail.com
     Editor: Lisa Huang
             lyihuang16@gmail.com
     Editor: Sonal Agarwal
             sagarwal12@gmail.com
     Editor: Dana Blair
             dblair@cisco.com";

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

     Copyright (c) 2018 IETF Trust and the persons identified as
     the document authors.  All rights reserved.
     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD
     License set forth in Section 4.c of the IETF Trust's Legal
     Provisions Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX; see
     the RFC itself for full legal notices.";

  revision 2018-10-01 {
    description
      "Initial version.";
    reference
      "RFC XXX: Network Access Control List (ACL) YANG Data Model.";
  }

  /*
   * Identities
   */
```

```
   /*
    * Forwarding actions for a packet
    */
   identity forwarding-action {
     description
       "Base identity for actions in the forwarding category";
   }

   identity accept {
     base forwarding-action;
     description
       "Accept the packet";
   }

   identity drop {
     base forwarding-action;
     description
       "Drop packet without sending any ICMP error message";
   }

   identity reject {
     base forwarding-action;
     description
       "Drop the packet and send an ICMP error message to the source";
   }

   /*
    * Logging actions for a packet
    */
   identity log-action {
     description
       "Base identity for defining the destination for logging actions";
   }

   identity log-syslog {
     base log-action;
     description
       "System log (syslog) the information for the packet";
   }

   identity log-none {
     base log-action;
     description
       "No logging for the packet";
   }

   /*
    * ACL type identities
```

```
    */
   identity acl-base {
     description
       "Base Access Control List type for all Access Control List type
        identifiers.";
   }

   identity ipv4-acl-type {
     base acl:acl-base;
     if-feature "ipv4";
     description
       "An ACL that matches on fields from the IPv4 header
        (e.g. IPv4 destination address) and layer 4 headers (e.g. TCP
        destination port).  An acl of type ipv4 does not contain
        matches on fields in the ethernet header or the IPv6 header.";
   }

   identity ipv6-acl-type {
     base acl:acl-base;
     if-feature "ipv6";
     description
       "An ACL that matches on fields from the IPv6 header
        (e.g. IPv6 destination address) and layer 4 headers (e.g. TCP
        destination port). An acl of type ipv6 does not contain
        matches on fields in the ethernet header or the IPv4 header.";
   }

   identity eth-acl-type {
     base acl:acl-base;
     if-feature "eth";
     description
       "An ACL that matches on fields in the ethernet header,
        like 10/100/1000baseT or WiFi Access Control List. An acl of
        type ethernet does not contain matches on fields in the IPv4
        header, IPv6 header or layer 4 headers.";
   }

   identity mixed-eth-ipv4-acl-type {
     base "acl:eth-acl-type";
     base "acl:ipv4-acl-type";
     if-feature "mixed-eth-ipv4";
     description
       "An ACL that contains a mix of entries that
        match on fields in ethernet headers,
        entries that match on IPv4 headers.
        Matching on layer 4 header fields may also exist in the
        list.";
   }
```

```
   identity mixed-eth-ipv6-acl-type {
     base "acl:eth-acl-type";
     base "acl:ipv6-acl-type";
     if-feature "mixed-eth-ipv6";
     description
       "ACL that contains a mix of entries that
        match on fields in ethernet headers, entries
        that match on fields in IPv6 headers. Matching on
        layer 4 header fields may also exist in the list.";
   }

   identity mixed-eth-ipv4-ipv6-acl-type {
     base "acl:eth-acl-type";
     base "acl:ipv4-acl-type";
     base "acl:ipv6-acl-type";
     if-feature "mixed-eth-ipv4-ipv6";
     description
       "ACL that contains a mix of entries that
        match on fields in ethernet headers, entries
        that match on fields in IPv4 headers, and entries
        that match on fields in IPv6 headers. Matching on
        layer 4 header fields may also exist in the list.";
   }

   /*
    * Features
    */

   /*
    * Features supported by device
    */
   feature match-on-eth {
     description
       "The device can support matching on ethernet headers.";
   }

   feature match-on-ipv4 {
     description
       "The device can support matching on IPv4 headers.";
   }

   feature match-on-ipv6 {
     description
       "The device can support matching on IPv6 headers.";
   }

   feature match-on-tcp {
     description
```

```
      "The device can support matching on TCP headers.";
  }

  feature match-on-udp {
    description
      "The device can support matching on UDP headers.";
  }

  feature match-on-icmp {
    description
      "The device can support matching on ICMP (v4 and v6) headers.";
  }

  /*
   * Header classifications combinations supported by
   * device
   */
  feature eth {
    if-feature "match-on-eth";
    description
      "Plain Ethernet ACL supported";
  }

  feature ipv4 {
    if-feature "match-on-ipv4";
    description
      "Plain IPv4 ACL supported";
  }

  feature ipv6 {
    if-feature "match-on-ipv6";
    description
      "Plain IPv6 ACL supported";
  }

  feature mixed-eth-ipv4 {
    if-feature "match-on-eth and match-on-ipv4";
    description
      "Ethernet and IPv4 ACL combinations supported";
  }

  feature mixed-eth-ipv6 {
    if-feature "match-on-eth and match-on-ipv6";
    description
      "Ethernet and IPv6 ACL combinations supported";
  }

  feature mixed-eth-ipv4-ipv6 {
```

```
    if-feature "match-on-eth and match-on-ipv4
                and match-on-ipv6";
    description
      "Ethernet, IPv4 and IPv6 ACL combinations supported.";
  }

  /*
   * Stats Features
   */
  feature interface-stats {
    description
      "ACL counters are available and reported only per interface";
  }

  feature acl-aggregate-stats {
    description
      "ACL counters are aggregated over all interfaces, and reported
       only per ACL entry";
  }

  /*
   * Attachment point features
   */
  feature interface-attachment {
    description
      "ACLs are set on interfaces.";
  }

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

  /*
   * Groupings
   */
  grouping acl-counters {
    description
      "Common grouping for ACL counters";

    leaf matched-packets {
```

```
      type yang:counter64;
      config false;
      description
        "Count of the number of packets matching the current ACL
         entry.

         An implementation should provide this counter on a
         per-interface per-ACL-entry basis if possible.

         If an implementation only supports ACL counters on a per
         entry basis (i.e., not broken out per interface), then the
         value should be equal to the aggregate count across all
         interfaces.

         An implementation that provides counters on a per entry per
         interface basis is not required to also provide an aggregate
         count, e.g., per entry -- the user is expected to be able
         implement the required aggregation if such a count is
         needed.";
    }

    leaf matched-octets {
      type yang:counter64;
      config false;
      description
        "Count of the number of octets (bytes) matching the current
         ACL entry.

         An implementation should provide this counter on a
         per-interface per-ACL-entry if possible.

         If an implementation only supports ACL counters per entry
         (i.e., not broken out per interface), then the value
         should be equal to the aggregate count across all interfaces.

         An implementation that provides counters per entry per
         interface is not required to also provide an aggregate count,
         e.g., per entry -- the user is expected to be able implement
         the required aggregation if such a count is needed.";
    }
  }

  /*
   * Configuration data nodes
   */
  container acls {
    description
      "This is a top level container for Access Control Lists.
```

```
        It can have one or more acl nodes.";
      list acl {
        key "name";
        description
          "An Access Control List (ACL) is an ordered list of
           Access Control Entries (ACE). Each ACE has a
           list of match criteria and a list of actions.
           Since there are several kinds of Access Control Lists
           implemented with different attributes for
           different vendors, this model accommodates customizing
           Access Control Lists for each kind and, for each vendor.";
        leaf name {
          type string {
            length "1..64";
          }
          description
            "The name of access list. A device MAY restrict the length
             and value of this name, possibly space and special
             characters are not allowed.";
        }
        leaf type {
          type acl-type;
          description
            "Type of access control list. Indicates the primary intended
             type of match criteria (e.g. ethernet, IPv4, IPv6, mixed,
             etc) used in the list instance.";
        }
        container aces {
          description
            "The aces container contains one or more ace nodes.";
          list ace {
            key "name";
            ordered-by user;
            description
              "List of Access Control Entries (ACEs)";
            leaf name {
              type string {
                length "1..64";
              }
              description
                "A unique name identifying this Access Control
                 Entry (ACE).";
            }

            container matches {
              description
                "The rules in this set determine what fields will be
                 matched upon before any action is taken on them.
```

```
                The rules are selected based on the feature set
                defined by the server and the acl-type defined.
                If no matches are defined in a particular container,
                then any packet will match that container. If no
                matches are specified at all in an ACE, then any
                packet will match the ACE.";

              choice l2 {
                container eth {
                  when "derived-from-or-self(/acls/acl/type, " +
                       "'acl:eth-acl-type')";
                  if-feature match-on-eth;
                  uses pf:acl-eth-header-fields;
                  description
                    "Rule set that matches ethernet headers.";
                }
                description
                  "Match layer 2 headers, for example ethernet
                   header fields.";
              }

              choice l3 {
                container ipv4 {
                  when "derived-from-or-self(/acls/acl/type, " +
                       "'acl:ipv4-acl-type')";
                  if-feature match-on-ipv4;
                  uses pf:acl-ip-header-fields;
                  uses pf:acl-ipv4-header-fields;
                  description
                    "Rule set that matches IPv4 headers.";
                }

                container ipv6 {
                  when "derived-from-or-self(/acls/acl/type, " +
                       "'acl:ipv6-acl-type')";
                  if-feature match-on-ipv6;
                  uses pf:acl-ip-header-fields;
                  uses pf:acl-ipv6-header-fields;
                  description
                    "Rule set that matches IPv6 headers.";
                }
                description
                  "Choice of either ipv4 or ipv6 headers";
              }

              choice l4 {
                container tcp {
                  if-feature match-on-tcp;
```

```
        uses pf:acl-tcp-header-fields;
        container source-port {
          choice source-port {
            case range-or-operator {
              uses pf:port-range-or-operator;
              description
                "Source port definition from range or
                 operator.";
            }
            description
              "Choice of source port definition using
               range/operator or a choice to support future
               'case' statements, such as one enabling a
               group of source ports to be referenced.";
          }
          description
            "Source port definition.";
        }
        container destination-port {
          choice destination-port {
            case range-or-operator {
              uses pf:port-range-or-operator;
              description
                "Destination port definition from range or
                 operator.";
            }
            description
              "Choice of destination port definition using
               range/operator or a choice to support future
               'case' statements, such as one enabling a
               group of destination ports to be referenced.";
          }
          description
            "Destination port definition.";
        }
        description
          "Rule set that matches TCP headers.";
      }

      container udp {
        if-feature match-on-udp;
        uses pf:acl-udp-header-fields;
        container source-port {
          choice source-port {
            case range-or-operator {
              uses pf:port-range-or-operator;
              description
                "Source port definition from range or
```

```
                    operator.";
                }
                description
                  "Choice of source port definition using
                   range/operator or a choice to support future
                   'case' statements, such as one enabling a
                   group of source ports to be referenced.";
              }
              description
                "Source port definition.";
            }
            container destination-port {
              choice destination-port {
                case range-or-operator {
                  uses pf:port-range-or-operator;
                  description
                    "Destination port definition from range or
                     operator.";
                }
                description
                  "Choice of destination port definition using
                   range/operator or a choice to support future
                   'case' statements, such as one enabling a
                   group of destination ports to be referenced.";
              }
              description
                "Destination port definition.";
            }
            description
              "Rule set that matches UDP headers.";
          }

          container icmp {
            if-feature match-on-icmp;
            uses pf:acl-icmp-header-fields;
            description
              "Rule set that matches ICMP headers.";
          }
          description
            "Choice of TCP, UDP or ICMP headers.";
        }

        leaf egress-interface {
          type if:interface-ref;
          description
            "Egress interface. This should not be used if this ACL
             is attached as an egress ACL (or the value should
             equal the interface to which the ACL is attached).";
```

```
            }

            leaf ingress-interface {
              type if:interface-ref;
              description
                "Ingress interface. This should not be used if this ACL
                 is attached as an ingress ACL (or the value should
                 equal the interface to which the ACL is attached)";
            }
          }

          container actions {
            description
              "Definitions of action for this ace entry";
            leaf forwarding {
              type identityref {
                base forwarding-action;
              }
              mandatory true;
              description
                "Specifies the forwarding action per ace entry";
            }

            leaf logging {
              type identityref {
                base log-action;
              }
              default log-none;
              description
                "Specifies the log action and destination for
                 matched packets. Default value is not to log the
                 packet.";
            }
          }
          container statistics {
            if-feature "acl-aggregate-stats";
            config false;
            description
              "Statistics gathered across all attachment points for the
               given ACL.";
            uses acl-counters;
          }
        }
      }
    }
    container attachment-points {
      description
        "Enclosing container for the list of
```

```
          attachment-points on which ACLs are set";

      /*
       * Groupings
       */
      grouping interface-acl {
        description
          "Grouping for per-interface ingress ACL data";

        container acl-sets {
          description
            "Enclosing container the list of ingress ACLs on the
             interface";

          list acl-set {
            key "name";
            ordered-by user;
            description
              "List of ingress ACLs on the interface";

            leaf name {
              type leafref {
                path "/acls/acl/name";
              }
              description
                "Reference to the ACL name applied on ingress";
            }

            list ace-statistics {
              if-feature "interface-stats";
              key "name";
              config false;
              description
                "List of Access Control Entries (ACEs)";
              leaf name {
                type leafref {
                  path "/acls/acl/aces/ace/name";
                }
                description
                  "The ace name";
              }
              uses acl-counters;
            }
          }
        }
      }

      list interface {
```

```
        if-feature interface-attachment;
        key "interface-id";
        description
          "List of interfaces on which ACLs are set";

        leaf interface-id {
          type if:interface-ref;
          description
            "Reference to the interface id list key";
        }

        container ingress {
          uses interface-acl;
          description
            "The ACLs applied to ingress interface";
        }
        container egress {
          uses interface-acl;
          description
            "The ACLs applied to egress interface";
        }
      }
    }
  }
}

<CODE ENDS>
```

4.2. IETF Packet Fields module

The packet fields module defines the necessary groups for matching on fields in the packet including ethernet, ipv4, ipv6, and transport layer fields. The "type" node determines which of these fields get included for any given ACL with the exception of TCP, UDP and ICMP header fields. Those fields can be used in conjunction with any of the above layer 2 or layer 3 fields.

Since the number of match criteria are very large, the base draft does not include these directly but references them by 'uses' statement to keep the base module simple. In case more match conditions are needed, those can be added by augmenting choices within container "matches" in ietf-access-control-list.yang model.

This module imports definitions from Common YANG Data Types [RFC6991] and references IP [RFC0791], ICMP [RFC0792], TCP [RFC0793], Definition of the Differentiated Services Field in the IPv4 and IPv6 Headers [RFC2474], The Addition of Explicit Congestion Notification (ECN) to IP [RFC3168], , IPv6 Scoped Address Architecture [RFC4007],

IPv6 Addressing Architecture [RFC4291], A Recommendation for IPv6 Address Text Representation [RFC5952], IPv6 [RFC8200].

<CODE BEGINS> file "ietf-packet-fields@2018-10-01.yang"

```
module ietf-packet-fields {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-packet-fields";
  prefix packet-fields;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991 - Common YANG Data Types.";
  }

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991 - Common YANG Data Types.";
  }

  import ietf-ethertypes {
    prefix eth;
    reference
      "RFC XXXX - Network ACL YANG Model.";
  }

  organization
    "IETF NETMOD (Network Modeling Language) Working
     Group";

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

     Editor: Mahesh Jethanandani
             mjethanandani@gmail.com
     Editor: Lisa Huang
             lyihuang16@gmail.com
     Editor: Sonal Agarwal
             sagarwal12@gmail.com
     Editor: Dana Blair
             dblair@cisco.com";

  description
    "This YANG module defines groupings that are used by
    ietf-access-control-list YANG module. Their usage is not
```

```
    limited to ietf-access-control-list and can be
    used anywhere as applicable.

    Copyright (c) 2018 IETF Trust and the persons identified as
    the document authors.  All rights reserved.
    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD
    License set forth in Section 4.c of the IETF Trust's Legal
    Provisions Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.";

  revision 2018-10-01 {
    description
      "Initial version.";
    reference
      "RFC XXX: Network Access Control List (ACL) YANG Data Model.";
  }

  /*
   * Typedefs
   */
  typedef operator {
    type enumeration {
      enum lte {
        description
          "Less than or equal.";
      }
      enum gte {
        description
          "Greater than or equal.";
      }
      enum eq {
        description
          "Equal to.";
      }
      enum neq {
        description
          "Not equal to.";
      }
    }
    description
      "The source and destination port range definitions
       can be further qualified using an operator. An
       operator is needed only if lower-port is specified
```

```
       and upper-port is not specified. The operator
       therefore further qualifies lower-port only.";
  }

  /*
   * Groupings
   */
  grouping port-range-or-operator {
    choice port-range-or-operator {
      case range {
        leaf lower-port {
          type inet:port-number;
          must ". <= ../upper-port" {
            error-message
              "The lower-port must be less than or equal to
               upper-port.";
          }
          mandatory true;
          description
            "Lower boundry for a port.";
        }
        leaf upper-port {
          type inet:port-number;
          mandatory true;
          description
            "Upper boundry for port.";
        }
      }
      case operator {
        leaf operator {
          type operator;
          default eq;
          description
            "Operator to be applied on the port below.";
        }
        leaf port {
          type inet:port-number;
          mandatory true;
          description
            "Port number along with operator on which to
             match.";
        }
      }
      description
        "Choice of specifying a port range or a single
         port along with an operator.";
    }
    description
```

```
      "Grouping for port definitions in the form of a
       choice statement.";
  }

  grouping acl-ip-header-fields {
    description
      "IP header fields common to ipv4 and ipv6";
    reference
      "RFC 791: Internet Protocol.";

    leaf dscp {
      type inet:dscp;
      description
        "Differentiated Services Code Point.";
      reference
        "RFC 2474: Definition of Differentiated services field
         (DS field) in the IPv4 and IPv6 headers.";
    }

    leaf ecn {
      type uint8 {
        range 0..3;
      }
      description
        "Explicit Congestion Notification.";
      reference
        "RFC 3168: Explicit Congestion Notification.";
    }

    leaf length {
      type uint16;
      description
        "In IPv4 header field, this field is known as the Total Length.
         Total Length is the length of the datagram, measured in octets,
         including internet header and data.

         In IPv6 header field, this field is known as the Payload
         Length, the length of the IPv6 payload, i.e. the rest of
         the packet following the IPv6 header, in octets.";
      reference
        "RFC 791: Internet Protocol,
         RFC 8200: Internet Protocol, Version 6 (IPv6) Specification.";
    }

    leaf ttl {
      type uint8;
      description
        "This field indicates the maximum time the datagram is allowed
```

```
        to remain in the internet system.  If this field contains the
        value zero, then the datagram must be dropped.

        In IPv6, this field is known as the Hop Limit.";
      reference
        "RFC 791: Internet Protocol,
        RFC 8200: Internet Protocol, Version 6 (IPv6) Specification.";
    }

    leaf protocol {
      type uint8;
      description
        "Internet Protocol number. Refers to the protocol of the
        payload. In IPv6, this field is known as 'next-header,
        and if extension headers are present, the protocol is
        present in the 'upper-layer' header.";
      reference
        "RFC 791: Internet Protocol,
        RFC 8200: Internet Protocol, Version 6 (IPv6) Specification.";
    }
  }

  grouping acl-ipv4-header-fields {
    description
      "Fields in IPv4 header.";

    leaf ihl {
      type uint8 {
        range "5..60";
      }
      description
        "An IPv4 header field, the Internet Header Length (IHL) is
        the length of the internet header in 32 bit words, and
        thus points to the beginning of the data. Note that the
        minimum value for a correct header is 5.";
    }

    leaf flags {
      type bits {
        bit reserved {
          position 0;
          description
            "Reserved. Must be zero.";
        }
        bit fragment {
          position 1;
          description
            "Setting value to 0 indicates may fragment, while setting
```

```
            the value to 1 indicates do not fragment.";
        }
        bit more {
          position 2;
          description
            "Setting the value to 0 indicates this is the last fragment,
             and setting the value to 1 indicates more fragments are
             coming.";
        }
      }
      description
        "Bit definitions for the flags field in IPv4 header.";
    }

    leaf offset {
      type uint16 {
        range "20..65535";
      }
      description
        "The fragment offset is measured in units of 8 octets (64 bits).
         The first fragment has offset zero. The length is 13 bits";
    }

    leaf identification {
      type uint16;
      description
        "An identifying value assigned by the sender to aid in
         assembling the fragments of a datagram.";
    }

    choice destination-network {
      case destination-ipv4-network {
        leaf destination-ipv4-network {
          type inet:ipv4-prefix;
          description
            "Destination IPv4 address prefix.";
        }
      }
      description
        "Choice of specifying a destination IPv4 address or
         referring to a group of IPv4 destination addresses.";
    }
    choice source-network {
      case source-ipv4-network {
        leaf source-ipv4-network {
          type inet:ipv4-prefix;
          description
            "Source IPv4 address prefix.";
```

```
        }
      }
      description
        "Choice of specifying a source IPv4 address or
         referring to a group of IPv4 source addresses.";
    }
  }

  grouping acl-ipv6-header-fields {
    description
      "Fields in IPv6 header";

    choice destination-network {
      case destination-ipv6-network {
        leaf destination-ipv6-network {
          type inet:ipv6-prefix;
          description
            "Destination IPv6 address prefix.";
        }
      }
      description
        "Choice of specifying a destination IPv6 address
         or referring to a group of IPv6 destination
         addresses.";
    }

    choice source-network {
      case source-ipv6-network {
        leaf source-ipv6-network {
          type inet:ipv6-prefix;
            description
              "Source IPv6 address prefix.";
        }
      }
      description
        "Choice of specifying a source IPv6 address or
         referring to a group of IPv6 source addresses.";
    }

    leaf flow-label {
      type inet:ipv6-flow-label;
      description
        "IPv6 Flow label.";
    }
    reference
      "RFC 4291: IP Version 6 Addressing Architecture
       RFC 4007: IPv6 Scoped Address Architecture
       RFC 5952: A Recommendation for IPv6 Address Text
```

```
                Representation";
  }

  grouping acl-eth-header-fields {
    description
      "Fields in Ethernet header.";

    leaf destination-mac-address {
      type yang:mac-address;
      description
        "Destination IEEE 802 MAC address.";
    }
    leaf destination-mac-address-mask {
      type yang:mac-address;
      description
        "Destination IEEE 802 MAC address mask.";
    }
    leaf source-mac-address {
      type yang:mac-address;
      description
        "Source IEEE 802 MAC address.";
    }
    leaf source-mac-address-mask {
      type yang:mac-address;
      description
        "Source IEEE 802 MAC address mask.";
    }
    leaf ethertype {
      type eth:ethertype;
      description
        "The Ethernet Type (or Length) value represented
         in the canonical order defined by IEEE 802.
         The canonical representation uses lowercase
         characters.";
      reference
        "IEEE 802-2014 Clause 9.2";
    }
    reference
      "IEEE 802: IEEE Standard for Local and Metropolitan
       Area Networks: Overview and Architecture.";
  }

  grouping acl-tcp-header-fields {
    description
      "Collection of TCP header fields that can be used to
       setup a match filter.";

    leaf sequence-number {
```

```
      type uint32;
      description
        "Sequence number that appears in the packet.";
    }

    leaf acknowledgement-number {
      type uint32;
      description
        "The acknowledgement number that appears in the
         packet.";
    }

    leaf data-offset {
      type uint8 {
        range "5..15";
      }
      description
        "Specifies the size of the TCP header in 32-bit
         words. The minimum size header is 5 words and
         the maximum is 15 words thus giving the minimum
         size of 20 bytes and maximum of 60 bytes,
         allowing for up to 40 bytes of options in the
         header.";
    }

    leaf reserved {
      type uint8;
      description
        "Reserved for future use.";
    }

    leaf flags {
      type bits {
        bit cwr {
          position 1;
          description
            "Congestion Window Reduced (CWR) flag is set by
             the sending host to indicate that it received
             a TCP segment with the ECE flag set and had
             responded in congestion control mechanism.";
          reference
            "RFC 3168: The Addition of Explicit Congestion
                       Notification (ECN) to IP.";
        }
        bit ece {
          position 2;
          description
            "ECN-Echo has a dual role, depending on the value
```

```
            of the SYN flag. It indicates:
            If the SYN flag is set (1), that the TCP peer is ECN
            capable. If the SYN flag is clear (0), that a packet
            with Congestion Experienced flag set (ECN=11) in IP
            header was received during normal transmission
            (added to header by RFC 3168). This serves as an
            indication of network congestion (or impending
            congestion) to the TCP sender.";
          reference
            "RFC 3168: The Addition of Explicit Congestion
                       Notification (ECN) to IP.";
        }
        bit urg {
          position 3;
          description
            "Indicates that the Urgent pointer field is significant.";
        }
        bit ack {
          position 4;
          description
            "Indicates that the Acknowledgment field is significant.
             All packets after the initial SYN packet sent by the
             client should have this flag set.";
        }
        bit psh {
          position 5;
          description
            "Push function. Asks to push the buffered data to the
             receiving application.";
        }
        bit rst {
          position 6;
          description
            "Reset the connection.";
        }
        bit syn {
          position 7;
          description
            "Synchronize sequence numbers. Only the first packet
             sent from each end should have this flag set. Some
             other flags and fields change meaning based on this
             flag, and some are only valid for when it is set,
             and others when it is clear.";
        }
        bit fin {
          position 8;
          description
            "Last package from sender.";
```

```
        }
      }
      description
        "Also known as Control Bits. Contains 9 1-bit flags.";
      reference
        "RFC 793: Transmission Control Protocol (TCP).";
    }

    leaf window-size {
      type uint16;
      units "bytes";
      description
        "The size of the receive window, which specifies
         the number of window size units beyond the segment
         identified by the sequence number in the acknowledgment
         field that the sender of this segment is currently
         willing to receive.";
    }

    leaf urgent-pointer {
      type uint16;
      description
        "This field is an offset from the sequence number
         indicating the last urgent data byte.";
    }

    leaf options {
      type binary {
        length "1..40";
      }
      description
        "The length of this field is determined by the
         data offset field. Options have up to three
         fields: Option-Kind (1 byte), Option-Length
         (1 byte), Option-Data (variable). The Option-Kind
         field indicates the type of option, and is the
         only field that is not optional. Depending on
         what kind of option we are dealing with,
         the next two fields may be set: the Option-Length
         field indicates the total length of the option,
         and the Option-Data field contains the value of
         the option, if applicable.";
    }
  }

  grouping acl-udp-header-fields {
    description
      "Collection of UDP header fields that can be used
```

```
       to setup a match filter.";

    leaf length {
      type uint16;
      description
        "A field that specifies the length in bytes of
         the UDP header and UDP data. The minimum
         length is 8 bytes because that is the length of
         the header. The field size sets a theoretical
         limit of 65,535 bytes (8 byte header + 65,527
         bytes of data) for a UDP datagram. However the
         actual limit for the data length, which is
         imposed by the underlying IPv4 protocol, is
         65,507 bytes (65,535 minus 8 byte UDP header
         minus 20 byte IP header).

         In IPv6 jumbograms it is possible to have
         UDP packets of size greater than 65,535 bytes.
         RFC 2675 specifies that the length field is set
         to zero if the length of the UDP header plus
         UDP data is greater than 65,535.";
    }
  }

  grouping acl-icmp-header-fields {
    description
      "Collection of ICMP header fields that can be
       used to setup a match filter.";

    leaf type {
      type uint8;
      description
        "Also known as Control messages.";
      reference
        "RFC 792: Internet Control Message Protocol (ICMP),
         RFC 4443: Internet Control Message Protocol (ICMPv6)
                   for Internet Protocol Version 6 (IPv6)
                   Specifciation.";
    }

    leaf code {
      type uint8;
      description
        "ICMP subtype. Also known as Control messages.";
      reference
        "RFC 792: Internet Control Message Protocol (ICMP),
         RFC 4443: Internet Control Message Protocol (ICMPv6)
                   for Internet Protocol Version 6 (IPv6)
```

```
                  Specifciation.";
    }

    leaf rest-of-header {
      type binary;
      description
        "Unbounded in length, the contents vary based on the
         ICMP type and code. Also referred to as 'Message Body'
         in ICMPv6.";
      reference
        "RFC 792: Internet Control Message Protocol (ICMP),
         RFC 4443: Internet Control Message Protocol (ICMPv6)
                   for Internet Protocol Version 6 (IPv6)
                   Specifciation.";
    }
  }
}
```

<CODE ENDS>

4.3. ACL Examples

Requirement: Deny tcp traffic from 192.0.2.0/24, destined to 198.51.100.0/24.

Here is the acl configuration xml for this Access Control List:

[note: '\' line wrapping for formatting only]

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <acls
    xmlns="urn:ietf:params:xml:ns:yang:ietf-access-control-list">
    <acl>
      <name>sample-ipv4-acl</name>
      <type>ipv4-acl-type</type>
      <aces>
        <ace>
          <name>rule1</name>
          <matches>
            <ipv4>
              <protocol>6</protocol>
              <destination-ipv4-network>198.51.100.0/24</destination\
-ipv4-network>
              <source-ipv4-network>192.0.2.0/24</source-ipv4-network\
>
            </ipv4>
          </matches>
          <actions>
            <forwarding>drop</forwarding>
          </actions>
        </ace>
      </aces>
    </acl>
  </acls>
</config>
```

The acl and aces can be described in CLI as the following:

```
      acl ipv4 sample-ipv4-acl
      deny tcp 192.0.2.0/24 198.51.100.0/24
```

Requirement: Accept all DNS traffic destined for 2001:db8::/32 on port 53.

```
[note: '\' line wrapping for formatting only]

<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <acls
    xmlns="urn:ietf:params:xml:ns:yang:ietf-access-control-list">
    <acl>
      <name>allow-dns-packets</name>
      <type>ipv6-acl-type</type>
      <aces>
        <ace>
          <name>rule1</name>
          <matches>
            <ipv6>
              <destination-ipv6-network>2001:db8::/32</destination-i\
pv6-network>
            </ipv6>
            <udp>
              <destination-port>
                <operator>eq</operator>
                <port>53</port>
              </destination-port>
            </udp>
          </matches>
          <actions>
            <forwarding>accept</forwarding>
          </actions>
        </ace>
      </aces>
    </acl>
  </acls>
</config>
```

4.4. Port Range Usage and Other Examples

When a lower-port and an upper-port are both present, it represents a range between lower-port and upper-port with both the lower-port and upper-port included. When only a port is present, it represents a port, with the operator specifying the range.

The following XML example represents a configuration where TCP traffic from source ports 16384, 16385, 16386, and 16387 is dropped.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <acls
    xmlns="urn:ietf:params:xml:ns:yang:ietf-access-control-list">
    <acl>
      <name>sample-port-acl</name>
      <type>ipv4-acl-type</type>
      <aces>
        <ace>
          <name>rule1</name>
          <matches>
            <tcp>
              <source-port>
                <lower-port>16384</lower-port>
                <upper-port>16387</upper-port>
              </source-port>
            </tcp>
          </matches>
          <actions>
            <forwarding>drop</forwarding>
          </actions>
        </ace>
      </aces>
    </acl>
  </acls>
</config>
```

The following XML example represents a configuration where all IPv4 ICMP echo requests are dropped.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <acls
    xmlns="urn:ietf:params:xml:ns:yang:ietf-access-control-list">
    <acl>
      <name>sample-icmp-acl</name>
      <aces>
        <ace>
          <name>rule1</name>
          <matches>
            <ipv4>
              <protocol>1</protocol>
            </ipv4>
            <icmp>
              <type>8</type>
              <code>0</code>
            </icmp>
          </matches>
          <actions>
            <forwarding>drop</forwarding>
          </actions>
        </ace>
      </aces>
    </acl>
  </acls>
</config>
```

The following XML example represents a configuration of a single port, port 21 that accepts TCP traffic.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <acls
    xmlns="urn:ietf:params:xml:ns:yang:ietf-access-control-list">
    <acl>
      <name>sample-ipv4-acl</name>
      <type>ipv4-acl-type</type>
      <aces>
        <ace>
          <name>rule1</name>
          <matches>
            <tcp>
              <destination-port>
                <operator>eq</operator>
                <port>21</port>
              </destination-port>
            </tcp>
          </matches>
          <actions>
            <forwarding>accept</forwarding>
          </actions>
        </ace>
      </aces>
    </acl>
  </acls>
</config>
```

The following XML example represents a configuration specifying all ports that are not equal to 21, that will drop TCP packets destined for those ports.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <acls
    xmlns="urn:ietf:params:xml:ns:yang:ietf-access-control-list">
    <acl>
      <name>sample-ipv4-acl</name>
      <type>ipv4-acl-type</type>
      <aces>
        <ace>
          <name>rule1</name>
          <matches>
            <tcp>
              <destination-port>
                <operator>neq</operator>
                <port>21</port>
              </destination-port>
            </tcp>
          </matches>
          <actions>
            <forwarding>drop</forwarding>
          </actions>
        </ace>
      </aces>
    </acl>
  </acls>
</config>
```

5. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocol such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Access Control Model (NACM [RFC8341]) provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in the YANG module which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., <edit-config>) to these data nodes without proper protection can have a negative effect on network operations.

These are the subtrees and data nodes and their sensitivity/ vulnerability:

/acls/acl/aces: This list specifies all the configured access control entries on the device. Unauthorized write access to this list can allow intruders to modify the entries so as to permit traffic that should not be permitted, or deny traffic that should be permitted. The former may result in a DoS attack, or compromise the device. The latter may result in a DoS attack. The impact of an unauthorized read access of the list will allow the attacker to determine which rules are in effect, to better craft an attack.

/acls/acl/aces/ace/actions/logging: This node specifies ability to log packets that match this ace entry. Unauthorized write access to this node can allow intruders to enable logging on one or many ace entries, overwhelming the server in the process. Unauthorized read access of this node can allow intruders to access logging information, which could be used to craft an attack the server.

6. IANA Considerations

This document registers three URIs and three YANG modules.

6.1. URI Registration

This document registers three URIs in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-access-control-list
URI: urn:ietf:params:xml:ns:yang:ietf-packet-fields
URI: urn:ietf:params:xml:ns:yang:ietf-ethertypes

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

6.2. YANG Module Name Registration

This document registers three YANG module in the YANG Module Names registry YANG [RFC6020].

```
name: ietf-access-control-list
namespace: urn:ietf:params:xml:ns:yang:ietf-access-control-list
prefix: acl
reference: RFC XXXX

name: ietf-packet-fields
namespace: urn:ietf:params:xml:ns:yang:ietf-packet-fields
prefix: packet-fields
reference: RFC XXXX

name: ietf-ethertypes
namespace: urn:ietf:params:xml:ns:yang:ietf-ethertypes
prefix: ethertypes
reference: RFC XXXX
```

7. Acknowledgements

Alex Clemm, Andy Bierman and Lisa Huang started it by sketching out an initial IETF draft in several past IETF meetings. That draft included an ACL YANG model structure and a rich set of match filters, and acknowledged contributions by Louis Fourie, Dana Blair, Tula Kraiser, Patrick Gili, George Serpa, Martin Bjorklund, Kent Watsen, and Phil Shafer. Many people have reviewed the various earlier drafts that made the draft went into IETF charter.

Dean Bogdanovic, Kiran Agrahara Sreenivasa, Lisa Huang, and Dana Blair each evaluated the YANG model in previous drafts separately, and then worked together to created a ACL draft that was supported by different vendors. That draft removed vendor specific features, and gave examples to allow vendors to extend in their own proprietary ACL. The earlier draft was superseded with this updated draft and received more participation from many vendors.

Authors would like to thank Jason Sterne, Lada Lhotka, Juergen Schoenwalder, David Bannister, Jeff Haas, Kristian Larsson and Einar Nilsen-Nygaard for their review of and suggestions to the draft.

8. References

8.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005, <https://www.rfc-editor.org/info/rfc4007>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

8.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/info/rfc7011>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Appendix A. Extending ACL model examples

A.1. A company proprietary module example

Module "example-newco-acl" is an example of company proprietary model that augments "ietf-acl" module. It shows how to use 'augment' with an XPath expression to add additional match criteria, actions, and default actions for when no ACE matches are found. All these are company proprietary extensions or system feature extensions. "example-newco-acl" is just an example and it is expected that vendors will create their own proprietary models.

```
module example-newco-acl {

  yang-version 1.1;

  namespace "http://example.com/ns/example-newco-acl";

  prefix example-newco-acl;

  import ietf-access-control-list {
    prefix "acl";
  }

  organization
    "Newco model group.";

  contact
    "abc@newco.com";
  description
    "This YANG module augments IETF ACL Yang.";

  revision 2018-10-01 {
    description
      "Creating NewCo proprietary extensions to ietf-acl model";

    reference
      "RFC XXXX: Network Access Control List (ACL)
       YANG Data  Model";
  }

  augment "/acl:acls/acl:acl/" +
          "acl:aces/acl:ace/" +
          "acl:matches" {
```

```
      description "Newco proprietary simple filter matches";
      choice protocol-payload-choice {
        description "Newco proprietary payload match condition";
        list protocol-payload {
          key value-keyword;
          ordered-by user;
          description "Match protocol payload";
          uses match-simple-payload-protocol-value;
        }
      }

      choice metadata {
        description "Newco proprietary interface match condition";
        leaf packet-length {
          type uint16;
          description "Match on packet length";
        }
      }
    }

    augment "/acl:acls/acl:acl/" +
            "acl:aces/acl:ace/" +
            "acl:actions" {
      description "Newco proprietary simple filter actions";
      choice action {
        description "";
        case count {
          description "Count the packet in the named counter";
          leaf count {
            type uint32;
            description "Count";
          }
        }
        case policer {
          description "Name of policer to use to rate-limit traffic";
          leaf policer {
            type string;
            description "Name of the policer";
          }
        }
        case hiearchical-policer {
          leaf hierarchitacl-policer {
            type string;
            description
              "Name of the hierarchical policer.";
          }
          description
            "Name of hierarchical policer to use to
```

```
             rate-limit traffic";
        }
      }
    }

    augment "/acl:acls/acl:acl" +
            "/acl:aces/acl:ace/" +
            "acl:actions" {
      leaf default-action {
        type identityref {
          base acl:forwarding-action;
        }
        default acl:drop;
        description
          "Actions that occur if no ace is matched.";
      }
      description
        "Newco proprietary default action";
    }

    grouping match-simple-payload-protocol-value {
      description "Newco proprietary payload";
      leaf value-keyword {
        type enumeration {
          enum icmp {
            description "Internet Control Message Protocol";
          }
          enum icmp6 {
            description
              "Internet Control Message Protocol
               Version 6";
          }
          enum range {
            description "Range of values";
          }
        }
        description "(null)";
      }
    }
  }
```

The following figure is the tree diagram of example-newco-acl. In this example, /ietf-acl:acls/ietf-acl:acl/ietf-acl:aces/ietf-acl:ace/ietf-acl:matches are augmented with two new choices, protocol-payload-choice and metadata. The protocol-payload-choice uses a grouping with an enumeration of all supported protocol values. Metadata matches apply to fields associated with the packet but not

in the packet header such as overall packet length. In another example, /ietf-acl:acls/ietf-acl:acl/ietf-acl:aces/ietf-acl:ace/ietf-acl:actions are augmented with a new choice of actions.

```
module: example-newco-acl
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
    +--rw (protocol-payload-choice)?
    |  +--:(protocol-payload)
    |     +--rw protocol-payload* [value-keyword]
    |        +--rw value-keyword    enumeration
    +--rw (metadata)?
       +--:(packet-length)
          +--rw packet-length?      uint16
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:actions:
    +--rw (action)?
       +--:(count)
       |  +--rw count?                  uint32
       +--:(policer)
       |  +--rw policer?                string
       +--:(hiearchical-policer)
          +--rw hierarchitacl-policer?   string
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:actions:
    +--rw default-action?   identityref
```

A.2. Linux nftables

As Linux platform is becoming more popular as networking platform, the Linux data model is changing. Previously ACLs in Linux were highly protocol specific and different utilities were used (iptables, ip6tables, arptables, ebtables), so each one had separate data model. Recently, this has changed and a single utility, nftables, has been developed. With a single application, it has a single data model for filewall filters and it follows very similarly to the ietf-access-control list module proposed in this draft. The nftables support input and output ACEs and each ACE can be defined with match and action.

The example in Section 4.3 can be configured using nftable tool as below.

```
nft add table ip filter
nft add chain filter input
nft add rule ip filter input ip protocol tcp ip saddr \
    192.0.2.1/24 drop
```

The configuration entries added in nftable would be.

```
        table ip filter {
          chain input {
            ip protocol tcp ip saddr 192.0.2.1/24 drop
          }
        }
```

We can see that there are many similarities between Linux nftables and IETF ACL YANG data models and its extension models. It should be fairly easy to do translation between ACL YANG model described in this draft and Linux nftables.

A.3. Ethertypes

The ACL module is dependent on the definition of ethertypes. IEEE owns the allocation of those ethertypes. This model is being included here to enable definition of those types till such time that IEEE takes up the task of publication of the model that defines those ethertypes. At that time, this model can be deprecated.

<CODE BEGINS> file "ietf-ethertypes@2018-10-01.yang"

```
module ietf-ethertypes {
  namespace "urn:ietf:params:xml:ns:yang:ietf-ethertypes";
  prefix ethertypes;

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

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

     Editor:   Mahesh Jethanandani
               <mjethanandani@gmail.com>";

  description
    "This module contains the common definitions for the
     Ethertype used by different modules. It is a
     placeholder module, till such time that IEEE
     starts a project to define these Ethertypes
     and publishes a standard.

     At that time this module can be deprecated.";

  revision 2018-10-01 {
    description
      "Initial revision.";
```

```
      reference
        "RFC XXXX: IETF Ethertype YANG Data Module.";
    }

    typedef ethertype {
      type union {
        type uint16;
        type enumeration {
          enum ipv4 {
            value 2048;
            description
              "Internet Protocol version 4 (IPv4) with a
               hex value of 0x0800.";
            reference
              "RFC 791: Internet Protocol.";
          }
          enum arp {
            value 2054;
            description
              "Address Resolution Protocol (ARP) with a
               hex value of 0x0806.";
            reference
              "RFC 826: An Ethernet Address Resolution Protocol.";
          }
          enum wlan {
            value 2114;
            description
              "Wake-on-LAN. Hex value of 0x0842.";
          }
          enum trill {
            value 8947;
            description
              "Transparent Interconnection of Lots of Links.
               Hex value of 0x22F3.";
            reference
              "RFC 6325: Routing Bridges (RBridges): Base Protocol
               Specification.";
          }
          enum srp {
            value 8938;
            description
              "Stream Reservation Protocol. Hex value of
               0x22EA.";
            reference
              "IEEE 801.1Q-2011.";
          }
          enum decnet {
            value 24579;
```

```
            description
              "DECnet Phase IV. Hex value of 0x6003.";
          }
          enum rarp {
            value 32821;
            description
              "Reverse Address Resolution Protocol.
               Hex value 0x8035.";
            reference
              "RFC 903. A Reverse Address Resolution Protocol.";
          }
          enum appletalk {
            value 32923;
            description
              "Appletalk (Ethertalk). Hex value 0x809B.";
          }
          enum aarp {
            value 33011;
            description
              "Appletalk Address Resolution Protocol. Hex value
               of 0x80F3.";
          }
          enum vlan {
            value 33024;
            description
              "VLAN-tagged frame (802.1Q) and Shortest Path
               Bridging IEEE 802.1aq with NNI compatibility.
               Hex value 0x8100.";
            reference
              "802.1Q.";
          }
          enum ipx {
            value 33079;
            description
              "Internetwork Packet Exchange (IPX). Hex value
               of 0x8137.";
          }
          enum qnx {
            value 33284;
            description
              "QNX Qnet. Hex value of 0x8204.";
          }
          enum ipv6 {
            value 34525;
            description
              "Internet Protocol Version 6 (IPv6). Hex value
               of 0x86DD.";
            reference
```

```
            "RFC 8200: Internet Protocol, Version 6 (IPv6)
                       Specification
             RFC 8201: Path MTU Discovery for IPv6.";
          }
          enum efc {
            value 34824;
            description
              "Ethernet flow control using pause frames.
               Hex value of 0x8808";
            reference
              "IEEE Std. 802.1Qbb.";
          }
          enum esp {
            value 34825;
            description
              "Ethernet Slow Protocol. Hex value of 0x8809.";
            reference
              "IEEE Std. 802.3-2015";
          }
          enum cobranet {
            value 34841;
            description
              "CobraNet. Hex value of 0x";
          }
          enum mpls-unicast {
            value 34887;
            description
              "MultiProtocol Label Switch (MPLS) unicast traffic.
               Hex value of 0x8847.";
            reference
              "RFC 3031: Multiprotocol Label Switching Architecture.";
          }
          enum mpls-multicast {
            value 34888;
            description
              "MultiProtocol Label Switch (MPLS) multicast traffic.
               Hex value of 0x8848.";
            reference
              "RFC 3031: Multiprotocol Label Switching Architecture.";
          }
          enum pppoe-discovery {
            value 34915;
            description
              "Point-to-Point Protocol over Ethernet. Used during
               the discovery process. Hex value of 0x8863.";
            reference
              "RFC 2516: A method for Transmitting PPP over Ethernet
                         PPPoE.";
```

```
          }
          enum pppoe-session {
            value 34916;
            description
              "Point-to-Point Protocol over Ethernet. Used during
               session stage. Hex value of 0x8864.";
            reference
              "RFC 2516: A method for Transmitting PPP over Ethernet
                         PPPoE.";
          }
          enum intel-ans {
            value 34925;
            description
              "Intel Advanced Networking Services. Hex value of
               0x886D.";
          }
          enum jumbo-frames {
            value 34928;
            description
              "Jumbo frames or Ethernet frames with more than
               1500 bytes of payload, upto 9000 bytes.";
          }
          enum homeplug {
            value 34939;
            description
              "Family name for the various power line
               communications. Hex value of 0x887B.";
          }
          enum eap {
            value 34958;
            description
              "Ethernet Access Protocol (EAP) over LAN. Hex value
               of 0x888E.";
            reference
              "IEEE 802.1X";
          }
          enum profinet {
            value 34962;
            description
              "PROcess FIeld Net (PROFINET). Hex value of 0x8892.";
          }
          enum hyperscsi {
            value 34970;
            description
              "SCSI over Ethernet. Hex value of 0x889A";
          }
          enum aoe {
            value 34978;
```

```
            description
              "Advanced Technology Advancement (ATA) over Ethernet.
               Hex value of 0x88A2.";
          }
          enum ethercat {
            value 34980;
            description
              "Ethernet for Control Automation Technology (EtherCAT).
               Hex value of 0x88A4.";
          }
          enum provider-bridging {
            value 34984;
            description
              "Provider Bridging (802.1ad) and Shortest Path Bridging
               (801.1aq). Hex value of 0x88A8.";
            reference
              "IEEE 802.1ad, IEEE 802.1aq).";
          }
          enum ethernet-powerlink {
            value 34987;
            description
              "Ethernet Powerlink. Hex value of 0x88AB.";
          }
          enum goose {
            value 35000;
            description
              "Generic Object Oriented Substation Event (GOOSE).
               Hex value of 0x88B8.";
            reference
              "IEC/ISO 8802-2 and 8802-3.";
          }
          enum gse {
            value 35001;
            description
              "Generic Substation Events. Hex value of 88B9.";
            reference
              "IEC 61850.";
          }
          enum sv {
            value 35002;
            description
              "Sampled Value Transmission. Hex value of 0x88BA.";
            reference
              "IEC 61850.";
          }
          enum lldp {
            value 35020;
            description
```

```
            "Link Layer Discovery Protocol (LLDP). Hex value of
             0x88CC.";
          reference
            "IEEE 802.1AB.";
        }
        enum sercos {
          value 35021;
          description
            "Sercos Interface. Hex value of 0x88CD.";
        }
        enum wsmp {
          value 35036;
          description
            "WAVE Short Message Protocl (WSMP). Hex value of
             0x88DC.";
        }
        enum homeplug-av-mme {
          value 35041;
          description
            "HomePlug AV MME. Hex value of 88E1.";
        }
        enum mrp {
          value 35043;
          description
            "Media Redundancy Protocol (MRP). Hex value of
             0x88E3.";
          reference
            "IEC62439-2.";
        }
        enum macsec {
          value 35045;
          description
            "MAC Security. Hex value of 0x88E5.";
          reference
            "IEEE 802.1AE.";
        }
        enum pbb {
          value 35047;
          description
            "Provider Backbone Bridges (PBB). Hex value of
             0x88E7.";
          reference
            "IEEE 802.1ah.";
        }
        enum cfm {
          value 35074;
          description
            "Connectivity Fault Management (CFM). Hex value of
```

```
              0x8902.";
            reference
              "IEEE 802.1ag.";
          }
          enum fcoe {
            value 35078;
            description
              "Fiber Channel over Ethernet (FCoE). Hex value of
               0x8906.";
            reference
              "T11 FC-BB-5.";
          }
          enum fcoe-ip {
            value 35092;
            description
              "FCoE Initialization Protocol. Hex value of 0x8914.";
          }
          enum roce {
            value 35093;
            description
              "RDMA over Converged Ethernet (RoCE). Hex value of
               0x8915.";
          }
          enum tte {
            value 35101;
            description
              "TTEthernet Protocol Control Frame (TTE). Hex value
               of 0x891D.";
            reference
              "SAE AS6802.";
          }
          enum hsr {
            value 35119;
            description
              "High-availability Seamless Redundancy (HSR). Hex
               value of 0x892F.";
            reference
              "IEC 62439-3:2016.";
          }
        }
      }
      description
        "The uint16 type placeholder is defined to enable
         users to manage their own ethertypes not
         covered by the module. Otherwise the module contains
         enum definitions for the more commonly used ethertypes.";
    }
  }
```

<CODE ENDS>

Authors' Addresses

Mahesh Jethanandani
VMware

Email: mjethanandani@gmail.com

Sonal Agarwal
Cisco Systems, Inc.

Email: sagarwal12@gmail.com

Lisa Huang

Email: huangyi_99@yahoo.com

Dana Blair

Email: dana@blairhome.com