I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

S. Hares
L. Dunbar
Huawei
R. White
Ericsson

March 13, 2017

Filter-Based Packet Forwarding ECA Policy
draft-ietf-i2rs-pkt-eca-data-model-03.txt

Abstract

This document describes the yang data model for packet forwarding policy that filters received packets and forwards (or drops) the packets. Filters for Layer 2, Layer 3, Layer 4, and packet-arrival time are linked together to support filtering for the routing layer. Prior to forwarding the packets out other interfaces, some of the fields in the packets may be modified. (If one considers the packet reception an event, this packet policy is a minimalistic Event-Match Condition-Action policy.) This policy controls forwarding of packets received by a routing device on one or more interfaces on which this policy is enabled.

This data model may be used in either the configuration datastore, control plane datastores, or the I2RS ephemeral control plane datastore.

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 September 14, 2017.

Table of Contents

1. Introduction

This document describes the yang data model for packet forwarding policy that filters received packets and forwards (or drops) the packets. Prior to forwarding the packets out other interfaces, some of the fields in the packets may be modified. Filters for Layer 2, Layer 3, Layer-4 and packet arrival time are linked together to support filtering for the routing layer.

If one considers the reception of a packet as an event, this minimalistic Event-Match Condition-Action policy. Full event-match-condition policy can be found at [I-D.ietf-supa-generic-policy-data-model] (or the information model at [I-D.ietf-supa-generic-policy-info-model]). This document will use the term packet-only ECA policy for this model utilizing the term "packet" in this fashion.

ACL data models [I-D.ietf-netmod-acl-model] can also provide a minimal set of filtering for packet-eca by compiling a large group of filters. However, this data model also provides the L2-L4 filters plus a concept of grouping and policy rules. The pkt-eca structure helps create users with structures with more substantial policy for security or data flow direction.

This packet-only ECA policy data model supports an ordered list of ECA policy rules

- packet headers for layer 2 to layer 4,
- interfaces the packet was received on,
- time packet was received, and
- size of packet.

The actions include packet modify actions and forwarding options. The modify options allow for the following:

- setting fields in the packet header at Layer 2 (L2) to Layer 4 (L4), and
- encapsulation and decapsulation the packet.

The forwardinng actions allow forwardsing the packet via interfaces, tunnels, next-hops, or dropping the packet. setting things within the packet at Layer 2 (L2) to layer 4 (L4).

This packet policy draft has been developed as a set of protocol independent policy It may be used for the configuration datastore, a control plane datastore, or an I2RS ephemeral control plane datastore [RFC7921]. For more information configuration and control plane datastores please see [I-D.ietf-netmod-revised-datastores]. This yang model may be transmitted over NETCONF [RFC6241] or RESTCONF [RFC8040]. For use with the control plane datastores and ephemeral control plane datastores, additional capabilities support control plane daatastores will need to be added to the base NETCONF and RESTCONF to support these datastores.

This yang data model depends on the the I2RS RIB [I-D.ietf-i2rs-rib-data-model] which can be deployed in an configuration datastore, a control plane datastore, or the I2RS ephemeral control plane datastore.)for informational module see [I-D.ietf-i2rs-rib-info-model]. The update of RIB entries via the rpc features allows datastore validation differences to be handled in the rpc code.

The first section of this draft contains an overview of the policy structure. The second provides a high-level yang module. The third contains the yang module.

1.1. Definitions and Acronyms

INSTANCE: Routing Code often has the ability to spin up multiple copies of itself into virtual machines. Each Routing code instance or each protocol instance is denoted as Foo_INSTANCE in the text below.

NETCONF: The Network Configuration Protocol

PCIM - Policy Core Information Model

RESTconf - http programmatic protocol to access yang modules

2. Generic Route Filters/Policy Overview

This generic policy model represents filter or routing policies as rules and groups of rules.

The basic concept are:

Rule Group

A rule group is is an ordered set of rules .

Rule

A Rule is represented by the semantics "If Condition then Action". A Rule may have a priority assigned to it.

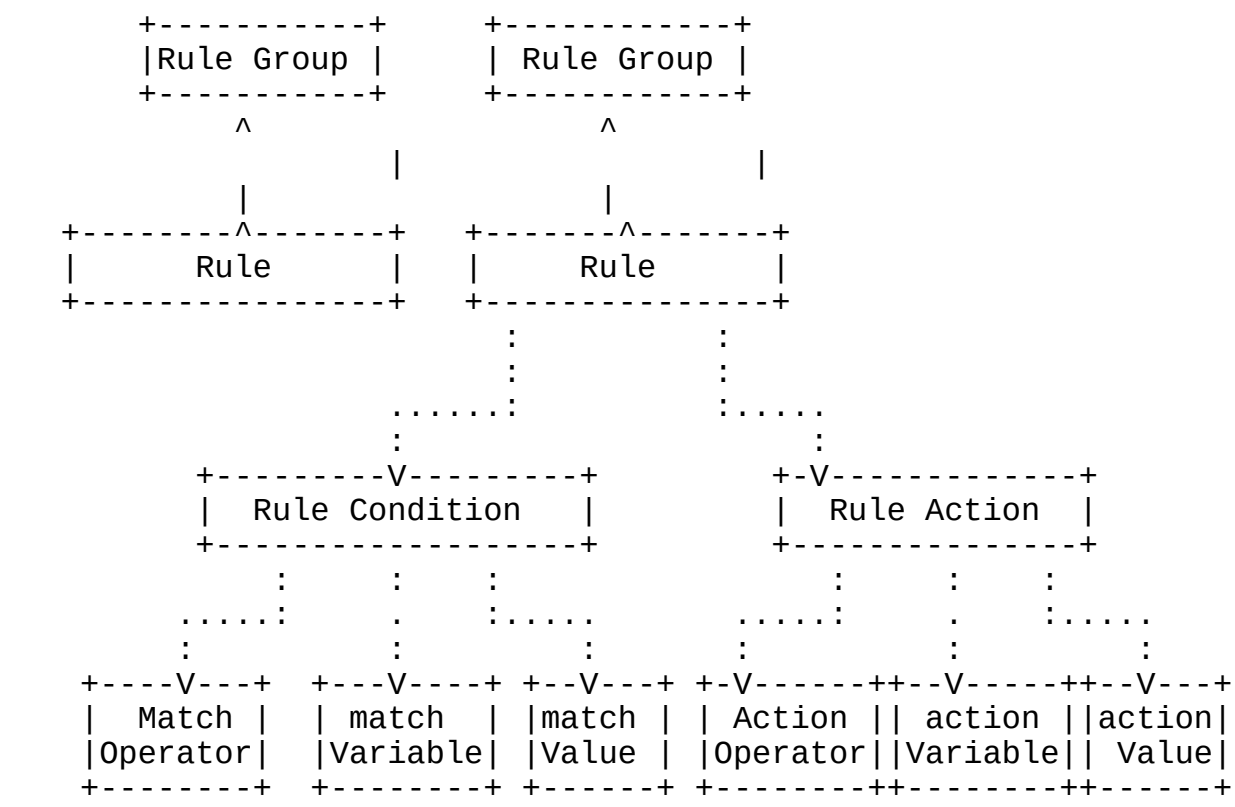


Figure 1: ECA rule structure

3. BNP Rule Groups

The pkt ECA policy is an order set of pkt-ECA policy rules. The rules assume the event is the reception of a packet on the machine on a set of interfaces. This policy is associated with a set of interfaces on a routing device (physical or virtual).

A Rule group allows for the easy combination of rules for management stations or users. A Rule group has the following elements:

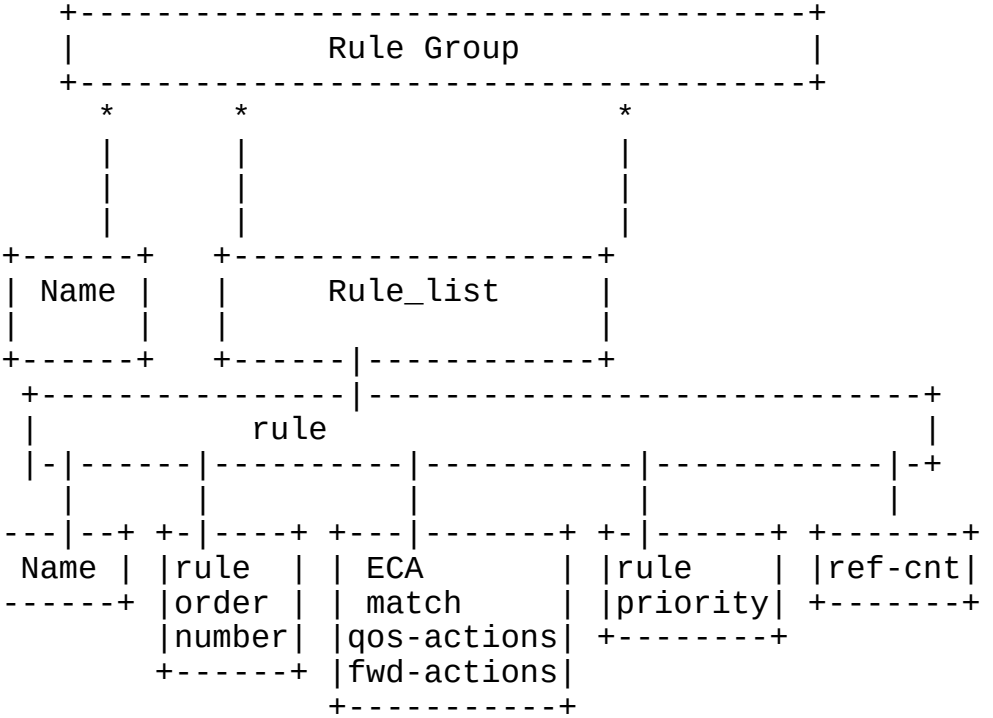
- name that identifies the grouping of policy rules
- module reference - reference to a yang module(s) in the yang module library that this group of policy writes policy to
- list of rules

Rule groups may have multiple policy groups at specific orders. For example, policy gorup 1 could have three policy rules at rule order 1 and four policy rules at rule order 5.

The rule has the following elements: name, order, status, priority, reference cnt, and match condition, and action as shown as shown in figure 2. The order indicates the order of the rule within the the complete list. The status of the rule is (active, inactive). The priority is the priority within a specific order of policy/filter rules. A reference count (refcnt) indicates the number of entities (E.g. network modules) using this policy. The generic rule match-action conditions have match operator, a match variable and a match value. The rule actions have an action operator, action variable, and an action value.

Rules can exist with the same rule order and same priority. Rules with the same rule order and same priority are not guaranteed to be at any specific ordering. The order number and priority have sufficient depth that administrators who wish order can specify it.

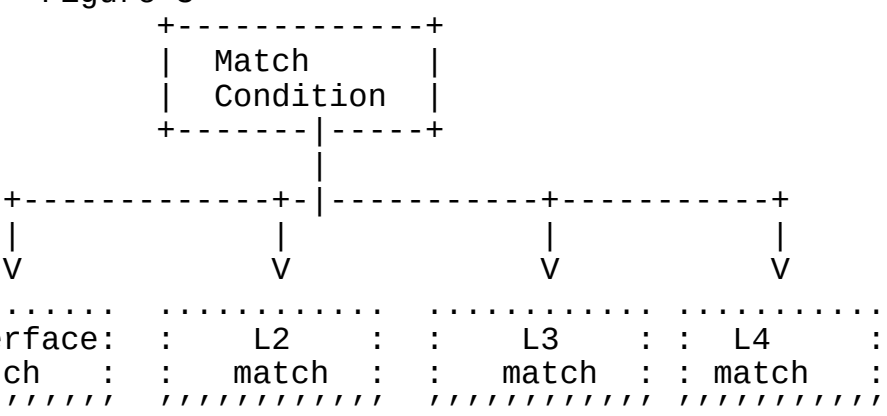
Figure 2 - Rule Group



The generic match conditions are specific to a particular layer are refined by matches to a specific layer (as figure 3 shows), and figure 5’s high-level yang defines. The general actions may be generic actions that are specific to a particular layer (L2, L3, or L4) or time of day or packet size. The qos actions can be setting fields in the packet at any layer (L2-l4) or encapsulating or decapsulating the packet at a layer. The fwd-actions are forwarding

functions that forward on an interface or to a next-hop. The rule status is the operational status per rule.

Figure 3



4. BNP Generic Info Model in High Level Yang

Below is the high level inclusion

Figure 5

```
module:pkt-eca-policy
 import ietf-inet-types  {prefix "inet"}
 import ietf-interface  {prefix "if"}
 import ietf-i2rs-rib   {prefix "iir"}

      import ietf-interfaces {
     prefix "if";
   }
      import ietf-inet-types {
     prefix inet;
     //rfc6991
    }
```

Below is the high level yang diagram

```
module ietf-pkt-eca-policy
  +--rw pkt-eca-policy-cfg
  |  +--rw pkt-eca-policy-set
  |     +--rw groups* [group-name]
  |     |  +--rw group-name string
  |     |  +--rw vrf-name string
  |     |  +--rw address-family
  |     |  +--rw group-rule-list* [rule-name]
  |     |  |  +--rw rule-name
```

```
|     |  |  +--rw rule-order-id
|     |  |  +--rw default-action-id integer
|     |  |  +--rw default-resolution-strategy-id integer
|     +--rw rules* [order-id rule-name]
|        +--rw order-id
|        +--rw rule-name
|        +--rw cfg-rule-conditions [cfgr-cnd-id]
|        |  +--rw cfgr-cnd-id integer
|        |  +--rw eca-event-match
|        |  |  +--rw time-event-match*
|        |  |  |   .. (time of day)
|        |  +--rw eca-condition-match
|        |  |  +--rw eca-pkt-matches*
|        |  |  |  ... (L2-L4 matches)
|        +--rw cfg-rule-actions [cfgr-action-id]
|        |  +--rw cfgr-action-id
|        |  +--rw eca-actions* [action-id]
|        |  |  +--rw action-id uint32
|        |  |  +--rw eca-ingress-act*
|        |  |  | ... (permit, deny, mirror)
|        |  |  +--rw eca-fwd-actions*
|        |  |  | ...  (invoke, tunnel encap, fwd)
|        |  |  +--rw eca-egress-act*
|        |  |  | .. .
|        |  |  +--rw eca-qos-actions*
|        |  |  | ...
|        |  |  +--rw ext-data-id integer
|        +--rw cfg-external-data* [cfg-ext-data-id]
|        |  +--rw cfg-ext-data-id integer
|        |  +--rw data-type integer
|        |  +--rw priority uint64
|        |  |  uses external-data-forms
|        |  ... (other external data)
+--rw pkt-eca-policy-opstate
   +--rw pkt-eca-opstate
      +--rw groups* [group-name]
      |  +--rw rules-installed;
      |  +--rw rules_status* [rule-name]
                |  +--rw strategy-used [strategy-id]
                |  +--rw
      +--rw rule-group-link* [rule-name]
      |  +--rw group-name
      +--rw rules_opstate* [rule-order rule-name]
      |  +--rw status
      |  +--rw rule-inactive-reason
      |  +--rw rule-install-reason
      |  +--rw rule-installer
      |  +--rw refcnt
```

```
          +--rw rules_op-stats* [rule-order rule-name]
          |  +--rw pkts-matched
          |  +--rw pkts-modified
          |  +--rw pkts-forward
                +--rw op-external-data [op-ext-data-id]
                |  +--rw op-ext-data-id integer
                |  +--rw type identityref
                |  +--rw installed-priority integer
                |  |  (other details on external data )
```

The three levels of policy are expressed as:

```
Config Policy definitions
=========================================
Policy level: pkt-eca-policy-set
group level:  pkt-eca-policy-set:groups
rule level:   pkt-eca-policy-set:rules
external id:  pkt-eca-policy-set:cfg-external-data

Operational State for Policy
=========================================
Policy level: pkt-eca-policy-opstate
group level:  pkt-eca-opstate:groups
group-rule:   pkt-eca-opstate:rule-group-link*
rule level:   pkt-eca_opstate:rules_opstate*
              pkt-eca_op-stats
```

figure

The filter matches struture is shown below

```
   module:i2rs-pkt-eca-policy
       +--rw pkt-eca-policy-cfg
     |  +--rw pkt-eca-policy-set
     |     +--rw groups* [group-name]
     |     |  ...
     |     +--rw rules [order-id rule-name]
     |        +--rw eca-matches
     |     |  |  +--case: interface-match
     |     |  |  +--case: L2-header-match
     |     |  |  +--case: L3-header-match
     |     |  |  +--case: L4-header-match
     |     |  |  +--case: packet-size
     |     |  |  +--case: time-of-day


    module:i2rs-pkt-eca-policy
     +--rw pkt-eca-policy-cfg
     |  +--rw pkt-eca-policy-set
     |     +--rw groups* [group-name]
     |     |  ...
     |     +--rw rules* [order-id rule-name]
     |        +--rw eca-matches
     |        |  . . .
     |        +--rw  ecq-qos-actions
     |        |  +--rw cnt-actions
     |        |  +--rw mod-actions
     |        |  |  +--case interface-actions
     |        |  |  +--case L2-action
     |        |  |  +--case L3-action
     |        |  |  +--case L4-action
     |        +--rw eca-fwd-actions
     |        |  +--rw num-fwd-actions
     |        |  +--rw fwd-actions
     |        |  |  +--rw interface interface-ref
     |        |  |  +--rw next-hop  rib-nexthop-ref
     |        |  |  +--rw route-attributes
     |        |  |  +--rw rib-route-attributes-ref
     |        |  |  +--rw fb-std-drop
```

5. i2rs-eca-policy Yang module

```
   <CODE BEGINS> file "ietf-pkt-eca-policy@2017-03-13.yang"
    module ietf-pkt-eca-policy {
       namespace "urn:ietf:params:xml:ns:yang:ietf-pkt-eca-policy";
      // replace with iana namespace when assigned
       prefix "pkt-eca-policy";
```

```
          import ietf-routing {
             prefix "rt";
           }
          import ietf-interfaces {
        prefix "if";
    }
          import ietf-inet-types {
        prefix inet;
        //rfc6991
     }

          import ietf-i2rs-rib {
     prefix "iir";
          }

  // meta
    organization "IETF I2RS WG";

  contact
     "email: shares@ndzh.com
          email: russ.white@riw.com
          email: linda.dunbar@huawei.com
      email: bill.wu@huawei.com";

  description
    "This module describes a basic network policy
        model with filter per layer.";

        revision "2017-03-13" {
           description "third revision";
           reference "draft-ietf-i2rs-pkt-eca-policy-dm-03";
         }


  // interfaces - no identity  matches


        // L2 header match identities
    identity l2-header-match-type {
    description
      " l2 header type for match ";
    }

  identity l2-802-1Q {
    base l2-header-match-type;
    description
      " l2 header type for 802.1Q match ";
   }
```

```
identity l2-802-11 {
  base l2-header-match-type;
  description
   " l2 header type for 802.11 match ";
     }

     identity l2-802-15 {
  base l2-header-match-type;
      description
   " l2 header type for 802.15 match ";
     }

     identity l2-NVGRE {
   base l2-header-match-type;
   description
   " l2 header type for NVGRE match ";
     }
     identity l2-mpls {
      base l2-header-match-type;
               description
   " l2 header type for MPLS match ";
     }

     identity l2-VXLAN {
  base l2-header-match-type;
      description
   " l2 header type for VXLAN match ";
     }


     // L3 header match identities
     identity l3-header-match-type {
 description
   " l3 header type for match ";
 }

     identity l3-ipv4-hdr {
      base l3-header-match-type;
      description
   " l3 header type for IPv4 match ";
     }

     identity l3-ipv6-hdr {
      base l3-header-match-type;
      description
   " l3 header type for IPv6 match ";
     }
```

```
identity l3-gre-tunnel {
 base l3-header-match-type;
   description "l3 header r
   type for GRE tunnel match ";
}

identity l3-icmp-header {
  base l3-header-match-type;
  description "L3 header match for ICMP";
 }

identity l3-ipsec-ah-header {
  base l3-header-match-type;
  description "AH IPSEC header ";
 }

identity l3-ipsec-esp-header {
  base l3-header-match-type;
  description "AH IPSEC header ";
 }

// L4 header match identities

identity l4-header-match-type {
 description "L4 header
 match types. (TCP, UDP,
 SCTP, UDPLite, etc. )";
 }

 identity l4-tcp-header {
  base l4-header-match-type;
  description "L4 header for TCP";
 }

 identity l4-udp-header {
  base l4-header-match-type;
  description "L4 header match for UDP";
 }

 identity l4-udplite {
  base l4-header-match-type;
  description "L4 header match for
    UDP lite";
 }

 identity l4-sctp-header {
  base l4-header-match-type;
  description "L4 header match for SCTP";
```

```
            }



          identity rule-status-type {
        description "status
            values for rule: invalid (0),
            valid (1), valid and installed (2)";
      }

          identity rule-status-invalid {
         base rule-status-type;
             description "invalid rule status.";
      }

      identity rule-status-valid {
         base rule-status-type;
             description "This status indicates
              a valid rule.";

          }

      identity rule-status-valid-installed {
         base rule-status-type;
             description "This status indicates
          an installed rule.";
      }
      identity rule-status-valid-inactive {
         base rule-status-type;
             description "This status indicates
                  a valid ruled that is not installed.";
      }



       grouping interface-match {
             leaf match-if-name {
                  type if:interface-ref;
              description "match on interface name";
             }
             description "interface
             has name, description, type, enabled
             as potential matches";
           }

           grouping interface-actions {
             description
             "interface action up/down and
```

```
            enable/disable";
                leaf interface-up {
                 type boolean;
                 description
                  "action to put interface up";
                 }
                leaf interface-down {
                  type boolean;
                 description
                  "action to put interface down";
                  }
                leaf interface-enable {
                  type boolean;
                   description
                  "action to enable interface";
                  }
                leaf interface-disable {
         type boolean;
                 description
                  "action to disable interface";
                }
        }


         grouping L2-802-1Q-header {
             description
                 "This is short-term 802.1 header
                  match which will be replaced
                  by reference to IEEE yang when
                  it arrives. Qtag 1 is 802.1Q
                  Qtag2 is 802.1AD";

                  leaf vlan-present {
                    type boolean;
                    description " Include VLAN in header";
                        }
                  leaf qtag1-present {
                    type boolean;
                     description " This flag value indicates
                         inclusion of one 802.1Q tag in header";
                        }
                  leaf qtag2-present{
                    type boolean;
                description "This flag indicates the
                           inclusion of second 802.1Q tag in header";
                    }

                  leaf dest-mac {
```

```
                type uint64;  //change to uint48
              description "IEEE destination MAC value
                          from the header";
                        }
                       leaf src-mac {
                         type uint64;                //change to uint48
                        description "IEEE source MAC
                         from the header";

                             }
                       leaf vlan-tag {
                             type uint16;
                        description "IEEE VLAN Tag
                         from the header";
                             }
                       leaf qtag1 {
                             type uint32;
                         description "Qtag1 value
                              from the header";
                             }
                       leaf qtag2 {
                          type uint32;
                              description "Qtag1 value
                              from the header";
                       }
                       leaf L2-ethertype {
                             type uint16;
                             description "Ether type
                             from the header";
                       }
              }


           grouping L2-VXLAN-header {
              container vxlan-header {
                     uses iir:ipv4-header;
                     leaf vxlan-network-id {
                         type uint32;
                         description "VLAN network id";
                        }
                     description " choices for
                       L2-VLAN header matches.
                           Outer-header only.
                           Need to fix inner header. ";
              }
               description
                   "This VXLAN header may
                   be replaced by actual VXLAN yang
```

```
            module reference";
        }

        grouping L2-NVGRE-header {

                container nvgre-header {
                   uses L2-802-1Q-header;
                   uses iir:ipv4-header;
                  leaf gre-version {
                    type uint8;
                    description "L2-NVGRE GRE version";
                    }
                  leaf gre-proto {
                    type uint16;
                    description "L2-NVGRE protocol value";
                        }
                  leaf virtual-subnet-id {
            type uint32;
                     description "L2-NVGRE subnet id value";
            }
         leaf flow-id {
            type uint16;
                     description "L2-NVGRE Flow id value";
                  }
                    // uses L2-802-1Q-header;
           description
                  "This NVGRE header may
                  be replaced by actual NVGRE yang
                   module reference";
           }
               description "Grouping for
                  L2 NVGRE header.";
        }


        grouping L2-header-match {

           choice l2-header-match-type {
                    case l2-802-1Q {
                          uses L2-802-1Q-header;
                       }
                       case l2-802-11 {
                         // matches for 802.11 headers
                       }
                       case l2-802-15 {
                        // matches for 802.1 Ethernet
                       }
                       case l2-NVGRE {
```

```
                        // matches for NVGRE
                        uses L2-NVGRE-header;
                       }
                       case l2-VXLAN-header {
                        uses L2-VXLAN-header;
                       }
                       case l2-mpls-header {
                         uses iir:mpls-header;
                       }
              description "Choice of L2
                 headers for L2 match";
             }
        description
              " The layer 2 header match includes
                any reference to L2 technology";
         }

         grouping L2-NVGRE-mod-acts {
          // actions for NVGRE
           leaf set-vsid {
        type boolean;
             description
                 "Boolean flag to set VSID in packet";
             }
           leaf set-flowid {
              type boolean;
              description
                   "Boolean flag to set VSID in packet";
             }
           leaf vsi {
            type uint32;
                description
                  "VSID value to set in packet";
                }
           leaf flow-id {
                 type uint16;
             description
                 "flow-id value to set in packet";
           }
           description "L2-NVRE Actions";
         }

         grouping L2-VXLAN-mod-acts {
           leaf set-network-id {
              type boolean;
              description
                 "flag to set network id in packet";
           }
```

```
            leaf network-id {
              type uint32;
                  description
                  "network id value to set in packet";
            }
            description "VXLAN header
              modification actions.";
          }

          grouping L2-mpls-mod-acts {
            leaf pop {
            type boolean;
             description
                  "Boolean flag to pop mpls header";
            }
            leaf push {
                 type boolean;
             description
                  "Boolean flag to push value into
                  mpls header";
            }
            leaf mpls-label {
                  type uint32;
                  description
                  "mpls label to push in header";
                 }
                 description "MPLS modify
                  header actions";
          }

          grouping l2-header-mod-actions {
                 leaf l2-802-1Q {
                   type uint8;
                   description "actions for 802.1Q";
                  }
                 leaf l2-802-11 {
                   type uint8;
                   description "actions for 802.11";
                   }
                 leaf l2-802-15 {
                   type uint8;
                   description "ations for 802.15";
                 }

                 uses L2-NVGRE-mod-acts;
        uses L2-VXLAN-mod-acts;
                 uses L2-mpls-mod-acts;
```

```
                description
                 " The layer 2 header match includes
                   any reference to L2 technology";
             }


            grouping L3-header-match {

                choice L3-header-match-type {
                       case l3-ipv4-hdr {
                         uses iir:ipv4-header;
                       }
                       case l3-ipv6-hdr {
                         uses iir:ipv6-header;
                       }
                       case L3-gre-tunnel {
                         uses iir:gre-header;
                       }
                         description "match for L3
                             headers for IPv4, IPv6,
                              and GRE tunnels";
                }
            description "match for L3 headers";
             }

            grouping ipv4-encapsulate-gre {
                leaf encapsulate {
                type boolean;
                    description "flag to encapsulate headers";
                      }
                 leaf ipv4-dest-address {
                            type inet:ipv4-address;
                            description
                            "Destination Address for GRE header";
                  }
                    leaf ipv4-source-address {
                             type inet:ipv4-address;
                         description
                             "Source Address for GRE header";
                  }
                    description
                    "encapsulation actions for IPv4 headers";
             }

            grouping L3-header-actions {
              choice l3-header-act-type {
                    case l3-ipv4-hdr {
                        leaf set-ttl {
```

```
                        type boolean;
                        description "flag to set TTL";
                    }
                        leaf set-dscp {
                        type boolean;
                        description "flag to set DSCP";
                        }
                    leaf ttl-value {
                        type uint8;
                        description "TTL value to set";
                        }
                        leaf dscp-val {
      type uint8;
                        description "dscp value to set";
                        }
                }
                    case l3-ipv6-hdr {
                        leaf set-next-header {
                        type boolean;
                        description
                         "flag to set next routing
                         header in IPv6 header";
                        }
                        leaf set-traffic-class {
                        type boolean;
                        description
                         "flag to set traffic class
                         in IPv6 header";

                         }
                        leaf set-flow-label {
                           type boolean;
                           description
                           "flag to set flow label
                             in IPv6 header";
                        }
                        leaf set-hop-limit {
                           type boolean;
                           description "flag
                            to set hop limit in
                                L3 packet";
                         }
                        leaf ipv6-next-header {
                           type uint8;
                           description "value to
                           set in next IPv6 header";
                         }
                        leaf ipv6-traffic-class {
```

```
                        type uint8;
                        description "value to set
                             in traffic class";

                    }
                    leaf ipv6-flow-label {
                        type uint16;
                         description "value to set
                               in IPOv6 flow label";
                    }
                    leaf ipv6-hop-limit {
                        type uint8;
                        description "value to set
                          in hop count";
                }
            }

            case L3-gre-tunnel {
                leaf decapsulate {
                        type boolean;
                   description "flag to
                            decapsulate GRE packet";
                     }
                 description "GRE tunnel
             actions" ;
             }
            description "actions that can
               be performed on L3 header";
      }
      description "actions to
       be performed on L3 header";
    }


     grouping tcp-header-match {
       leaf tcp-src-port {
         type uint16;
         description "source port match value";
             }
       leaf tcp-dst-port {
         type uint16;
     description "dest port value
              to match";
             }
             leaf sequence-number {
              type uint32;
              description "sequence number
               value to match";
```

```
                }
                leaf ack-number {
                type uint32;
                description "action value to
                 match";
                }
           description "match for TCP
                header";
         }

          grouping tcp-header-action {
            leaf set-tcp-src-port {
              type boolean;
              description "flag to set
                    source port value";
            }
            leaf set-tcp-dst-port {
             type boolean;
             description "flag to set source port value";
            }

            leaf tcp-s-port {
              type uint16;
              description "source port match value";
                }
            leaf tcp-d-port {
              type uint16;
          description "dest port value
                    to match";
             }
             leaf seq-num {
                    type uint32;
                    description "sequence number
                     value to match";
                }
                leaf ack-num {
                type uint32;
                description "action value to
                 match";
                }
           description "Actions to
              modify TCP header";
         }

         grouping udp-header-match {
             leaf udp-src-port {
                  type uint16;
              description "UDP source
```

```
                  port match value";
                }
               leaf udp-dst-port {
                 type uint16;
             description "UDP Destination
                   port match value";
                }
            description "match values for
                 UDP header";

          }

          grouping udp-header-action {
              leaf set-udp-src-port {
                   type boolean;
               description "flag to set
                     UDP source port match value";
                }
               leaf set-udp-dst-port {
                type boolean;
               description
                     "flag to set UDP destination port match value";
                }
               leaf udp-s-port {
                type uint16;
               description "UDP source
                     port match value";
                }
               leaf udp-d-port {
                 type uint16;
              description "UDP Destination
                    port match value";
                }

             description "actions to set
              values in UDP header";
          }

          grouping sctp-chunk {
                  leaf chunk-type {
                    type uint8;
                    description "sctp chunk type value";
                  }
                  leaf chunk-flag {
                    type uint8;
                    description "sctp chunk type
                     flag value";
              }
```

```
                leaf chunk-length {
                  type uint16;
                  description "sctp chunk length";
               }

              leaf chunk-data-byte-zero {
                     type uint32;
                     description "byte zero of
                      stcp chunk data";
              }
              description "sctp chunck
               header match fields";
          }

           grouping sctp-header-match {
              uses sctp-chunk;
              leaf stcp-src-port {
                   type uint16;
               description "sctp header match
                     source port value";
                 }
                 leaf sctp-dst-port {
                  type uint16;
                description "sctp header match
                    destination port value";
                 }
                 leaf sctp-verify-tag {
                  type uint32;
               description "sctp header match
                     verification tag value";
                 }
                 description "SCTP header
                      match values";
          }

          grouping sctp-header-action {
              leaf set-stcp-src-port {
                   type boolean;
               description "set source port in sctp header";
                 }
                 leaf set-stcp-dst-port {
                  type boolean;
               description "set destination port in sctp header";
                 }
                 leaf set-stcp-chunk1 {
                  type boolean;
               description "set chunk value in sctp header";
                 }
```

```
                leaf chunk-type-value  {
                  type uint8;
                  description "sctp chunk type value";
                }
                leaf chunk-flag-value {
                  type uint8;
                  description "sctp chunk type
                   flag value";
              }

                leaf chunk-len {
                  type uint16;
                  description "sctp chunk length";
              }

             leaf chunk-data-bzero {
                    type uint32;
                    description "byte zero of
                     stcp chunk data";
             }
             description "sctp qos actions";
          }


          grouping L4-header-match {
               choice l4-header-match-type {
                     case l4-tcp-header {
                       uses tcp-header-match;
                     }
                     case l4-udp-header {
                       uses udp-header-match;
                     }
                     case l4-sctp {
                       uses sctp-header-match;
                     }
                   description "L4 match
                   header choices";
                   }
              description "L4 header
                    match type";
          }


          grouping L4-header-actions {
               uses tcp-header-action;
               uses udp-header-action;
               uses sctp-header-action;
                  description "L4 header matches";
```

```
            }

      grouping rule_status {
             leaf rule-status {
               type string;
                   description "status information
                    free form string.";
             }
             leaf rule-inactive-reason {
               type string;
               description "description of
                   why rule is inactive";
             }
             leaf rule-install-reason {
               type string;
               description "response on rule installed";
             }
             leaf rule-installer {
               type string;
                description "client id of installer";
             }
             leaf refcnt {
               type uint16;
           description "reference count on rule. ";
         }
         description
              "rule operational status";
           }

   // group status
           grouping groups-status  {
             list group_opstate {
               key "grp-name";
               leaf grp-name {
               type string;
               description "eca group name";
              }
              leaf rules-installed {
               type uint32;
                   description "rules in
                    group installed";
              }
              list rules_status {
               key "rule-name";
                   leaf rule-name {
                    type string;
                    description "name of rule ";
                   }
```

```
              leaf rule-order {
                   type uint32;
                   description "rule-order";
                  }
                  description "rules per
                  group";
               }
               description "group operational
                status";
             }
            description "group to rules
                list";
            }

   // links between rule to group

             grouping rule-group-link {
              list rule-group {
               key rule-name;
               leaf rule-name {
                 type string;
                 description "rule name";
               }
               leaf group-name {
                  type string;
                      description "group name";
               }
               description "link between
                 group and link";
              }
              description "rule-name to
               group link";
            }

   // rule status by name
            grouping rules_opstate {
               list rules_status {
                    key "rule-order rule-name";
                      leaf rule-order {
                       type uint32;
                       description "order of rules";
                      }
                    leaf rule-name {
                     type string;
                     description "rule name";
                    }
                uses rule_status;
                    description "eca rule list";
```

```
            }
           description "rules
                  operational state";
          }

   // rule statistics by name and order
          grouping rules_opstats {
                  list rule-stat {
                key "rule-order rule-name";
                    leaf rule-order {
                     type uint32;
                     description "order of rules";
                    }
            leaf rule-name {
                     type string;
                     description "name of rule";
                    }
                    leaf pkts-matched {
                      type uint64;
                      description "number of
                       packets that matched filter";
                     }
                    leaf pkts-modified {
                      type uint64;
                      description "number of
                       packets that filter caused
                       to be modified";
                     }
                          leaf pkts-dropped {
                      type uint64;
                      description "number of
                       packets that filter caused
                       to be modified";
                     }
                  leaf bytes-dropped {
                      type uint64;
                      description "number of
                       packets that filter caused
                       to be modified";
                     }
                     leaf pkts-forwarded {
                      type uint64;
                      description "number of
                       packets that filter caused
                       to be forwarded.";
                      }
                      leaf bytes-forwarded {
                      type uint64;
```

```
                    description "number of
                     packets that filter caused
                     to be forwarded.";
                    }

                        description "list of
                        operational statistics for each
                        rule.";
                  }
                  description "statistics
                     on packet filter matches, and
                     based on matches on many were
                     modified and/or forwarded";
         }



      grouping packet-size-match {
            leaf l2-size-match {
               type uint32;
                   description "L2 packet match size.";
                   }
            leaf l3-size-match {
               type uint32;
               description "L3 packet match size.";
                   }
            leaf l4-size-match {
               type uint32;
               description "L4 packet match size.";
                   }

        description "packet size by layer
            only non-zero values are matched";
           }

           grouping time-day-match {

          leaf hour {
                type uint8;
                description "hour
                    of day in 24  hours.
                    (add range)";
             }
             leaf minute {
                type uint8;
                    description
                    "minute in day.";
             }
```

```
            leaf second {
               type uint8;
                    description
                    "second in day.";
            }

          description "matches for
            time of day.";

          }



          grouping eca-event-matches {
             uses time-day-match;
             description "matches for events
              which include:
                  time of day.";

          }

        grouping eca-pkt-matches {
            uses interface-match;
            uses L2-header-match;
            uses L3-header-match;
            uses L4-header-match;
            uses packet-size-match;
            description "ECA matches";
          }

         grouping user-status-matches {
            leaf user {
             type string;
             description "user";
            }
            leaf region {
             type string;
             description "region";
             }
            leaf state {
             type string;
             description "state";
            }

            leaf user-status {
             type string;
             description "status of user";
            }
```

```
            description "user status
            matches - region,
            target, location";
          }

          grouping eca-condition-matches {
             uses eca-pkt-matches;
             uses user-status-matches;
             description "pkt
              and user status matches";
          }

          grouping eca-qos-actions {
            leaf cnt-actions {
              type uint32;
              description "count of ECA actions";
             }
            list qos-actions {
               key "action-id";
               leaf action-id {
               type uint32;
               description "action id";
               }
               uses interface-actions;
               uses l2-header-mod-actions;
               uses L3-header-actions;
               uses L4-header-actions;

              description "ECA set or change
               packet Actions. Actions may be
                   added here for interface,
                   L2, L3, and L4
                   headers.";
             }
            description "eca- qos actions";
           }

           grouping ip-next-fwd {
                  leaf rib-name {
                    type string;
                    description "name of RIB";
              }
            leaf next-hop-name {
              type string;
                  description "name of next hop";
                  }
            description "ECA set or change
                  packet Actions";
```

```
           }

         grouping eca-ingress-actions {
           leaf permit {
            type boolean;
            description "permit ingress
             traffic.  False
                 means to deny.";
           }
           leaf mirror {
            type boolean;
            description "copy bytes
             ingressed to mirror port";
           }
           description "ingress eca match";
         }

         grouping eca-fwd-actions {
       leaf interface-fwd {
              type if:interface-ref;
              description "name of interface to forward on";
           }
       uses iir:nexthop;
           uses ip-next-fwd;
           leaf drop-packet {
            type boolean;
            description "drop packet flag";
           }
           description "ECA forwarding actions";
         }

     grouping eca-security-actions {
        leaf actions-exist {
           type boolean;
           description "existance of
            eca security actions";
        }
        description "content actions
             for security. Needs more
             description.";
     }

     grouping eca-egress-actions {
       leaf packet-rate {
             type uint32;
             description "maximum packet-rate";
       }
      leaf byte-rate {
```

```
            type uint64;
            description "maximum byte-rate ";
        }
        description "packet security actions";
      }


        grouping policy-conflict-resolution {
              list resolution-strategy {
               key "strategy-id";
               leaf strategy-id {
                 type uint32;
                 description "Id for strategy";
               }
               leaf stategy-name {
                type string;
                    description "name of strategy";
               }
               leaf filter-strategy {
                 type string;
                 description "type of resolution";

               }
               leaf global-strategy {
                 type boolean;
                 description "global strategy";
              }
              leaf mandatory-strategy {
               type boolean;
               description "required strategy";
              }
              leaf local-strategy {
               type boolean;
               description "local strategy";
              }
              leaf resolution-fcn {
               type uint64;
               description "resolution function id ";
              }
              leaf resolution-value {
               type uint64;
               description "resolution value";
              }
              leaf resolution-info {
               type string;
               description "resolution info";
              }
              list associate-ext-data {
```

```
                key "ext-data-id";
                    leaf ext-data-id {
                      type uint64;
                      description "ID of external data";
                    }
                    leaf ext-data {
                      type string;
                      description "external data";
                    }
                  description "linked external data";
             }
             description "list of strategies";
           }
            description "policy conflict
              resolution strategies";
          }


          grouping cfg-external-data {
            list cfg-ext-data {
              key "cfg-ext-data-id";
               leaf cfg-ext-data-id {
               type uint64;
               description "id for external data";
              }
              leaf data-type {
               type uint32;
               description "external data type ID";
               }
               leaf priority {
                type uint64;
                description "priority of data";
               }
              leaf other-data {
               type string;
               description "string
                    external data";
              }
            description "external data";
          }
           description "external data list";
        }


         grouping pkt-eca-policy-set {
             list groups {
                key "group-name";
                    leaf group-name {
```

```
                  type string;
                  description
                   "name of group of rules";
                 }
                leaf vrf-name {
                  type string;
                  description "VRF name";
                }
                uses rt:address-family;
                list group-rule-list {
                  key "rule-name";
                  leaf rule-name {
                   type string;
                   description "name of rule";
                   }
                  leaf rule-order-id {
                    type uint16;
                        description "rule-order-id";
                  }
                   description "rules per group";
                }
                description "pkt eca rule groups";
          }
          list eca-rules {
                key "order-id";
                ordered-by user;
                leaf order-id {
                   type uint16;
                   description "Number of order
                    in ordered list (ascending)";
                }
                leaf eca-rule-name {
                    type string;
                        description "name of rule";
                }
                 leaf installer {
                   type string;
                   description
                         "Id of I2RS client
                          that installs this rule.";
                 }
                  uses eca-event-matches;
                  uses eca-ingress-actions;
                  uses eca-qos-actions;
                  uses eca-security-actions;
                  uses eca-fwd-actions;
                  uses eca-egress-actions;
                  uses cfg-external-data;
```

```
                    uses policy-conflict-resolution;

                    description "ECA rules";
              }  // end of rule
            description "Policy sets.";
      }

      grouping pkt-eca-opstate {
       uses groups-status;
           uses rule-group-link;
       uses rules_opstate;
           uses rules_opstats;
            description "pkt eca policy
             op-state main";
      }


     container pkt-eca-policy-opstate {
       config "false";
       uses pkt-eca-opstate;
       description "operational state";
       }

     }


     <CODE ENDS>
```

6. IANA Considerations

This draft requests IANA Assign a urn in the IETF yang module space for:

"urn:ietf:params:xml:ns:yang:ietf-pkt-eca-policy";

associated prefix "pkt-eca";

7. Security Considerations

These generic filters are filter packets in a traffic stream, act to modify packets, and forward data packets. These filters operate dynamically at same level as currently deployed configured filter-based RIBs to filter, change, and forward traffic.

Due to the potential to use Filters as an attack vector, this data model should be used with the secure transport described in the [I-D.ietf-i2rs-protocol-security-requirements]

8. References

8.1. Normative References

[I-D.ietf-i2rs-rib-data-model]
Wang, L., Ananthakrishnan, H., Chen, M., amit.dass@ericsson.com, a., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-07 (work in progress), January 2017.

8.2. Informative References

[I-D.ietf-i2rs-protocol-security-requirements]
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-ietf-i2rs-protocol-security-requirements-17 (work in progress), September 2016.

[I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-10 (work in progress), December 2016.

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-10 (work in progress), March 2017.

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "A Revised Conceptual Model for YANG Datastores", draft-ietf-netmod-revised-datastores-00 (work in progress), December 2016.

[I-D.ietf-supa-generic-policy-data-model]
Halpern, J. and J. Strassner, "Generic Policy Data Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-data-model-02 (work in progress), October 2016.

[I-D.ietf-supa-generic-policy-info-model]
Strassner, J., Halpern, J., and S. Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-02 (work in progress), January 2017.

[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, <http://www.rfc-editor.org/info/rfc6241>.

[RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <http://www.rfc-editor.org/info/rfc7921>.

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

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Linda Dunbar
Huawei

Email: Linda.Dunbar@huawei.com

Russ White
Ericsson

Email: russw@riw.us