I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: December 18, 2016

S. Hares
Q. Wu
Huawei
R. White
Ericsson

June 16, 2016

Filter-Based Packet Forwarding ECA Policy
draft-ietf-i2rs-pkt-eca-data-model-00.txt

Abstract

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. 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. The policy is composed of an ordered list of policy rules. Each policy policy rule contains a set of match conditions that filters for packets plus a set of actions to modify the packet and forward packets. The match conditions can match tuples in multiple layers (L1-L4, application), interface received on, and and other conditions regarding the packet (size of packet, time of day). The modify packet actions allow for setting things within the packet plus decapsulation and encapsulation packet. The forwarding actions include forwarding via interfaces, tunnels, or nexthops and dropping the packet. The policy model can be used with the session ephemeral (BGP Flow Specifications), reboot ephemeral state (I2RS ephemeral), and non-ephemeral routing/forwarding state (e.g. configuration state).

This Internet-Draft will expire on December 18, 2016.

Copyright Notice

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. If one considers the reception of a packet as an event, this minimalistic Event-Match Condition-Action policy. If one considers the reception of packets containing Layer 1 to Layer 4 + application data a single packet, then this minimalistic policy can be called a packet-only ECA policy. This document will use the term packet-only ECA policy for this model utilizing the term "packet" in this fashion.

This packet-only ECA policy data model supports an ordered list of ECA policy rules where each policy rule has a name. The match condition filters include matches on

- packet headers for layer 1 to layer 4,
- application protocol data and headers,
- 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 forwardingng 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) plus overlay or application data.

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

1.2. Antecedents this Policy in IETF

Antecedents to this generic policy are the generic policy work done in PCIM WG. The PCIM work contains a Policy Core Information Model (PCIM) [RFC3060], Policy Core Informational Model Extensions [RFC3460] and the Quality of Service (QoS) Policy Information Model (QPIM) ([RFC3644]) From PCIM comes the concept that policy rules which are combined into policy groups. PCIM also refined a concept

of policy sets that allowed the nesting and aggregation of policy groups. This generic model did not utilize the concept of sets of groups, but could be expanded to include sets of groups in the future.

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.

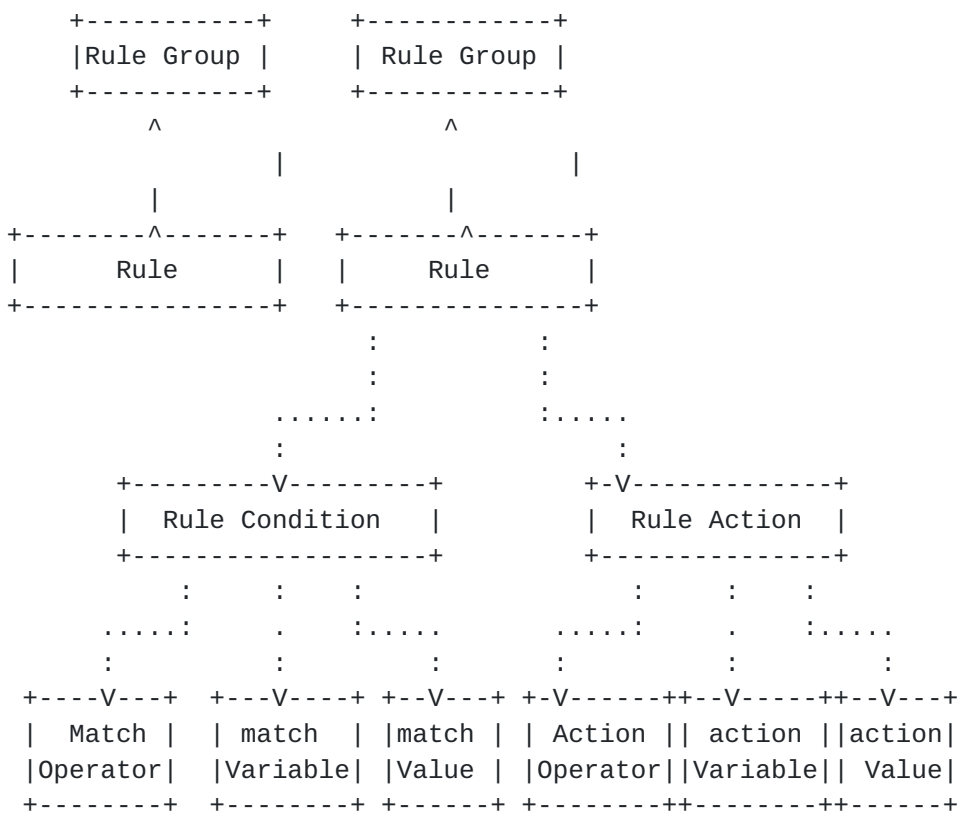


Figure 1: ECA rule structure

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:

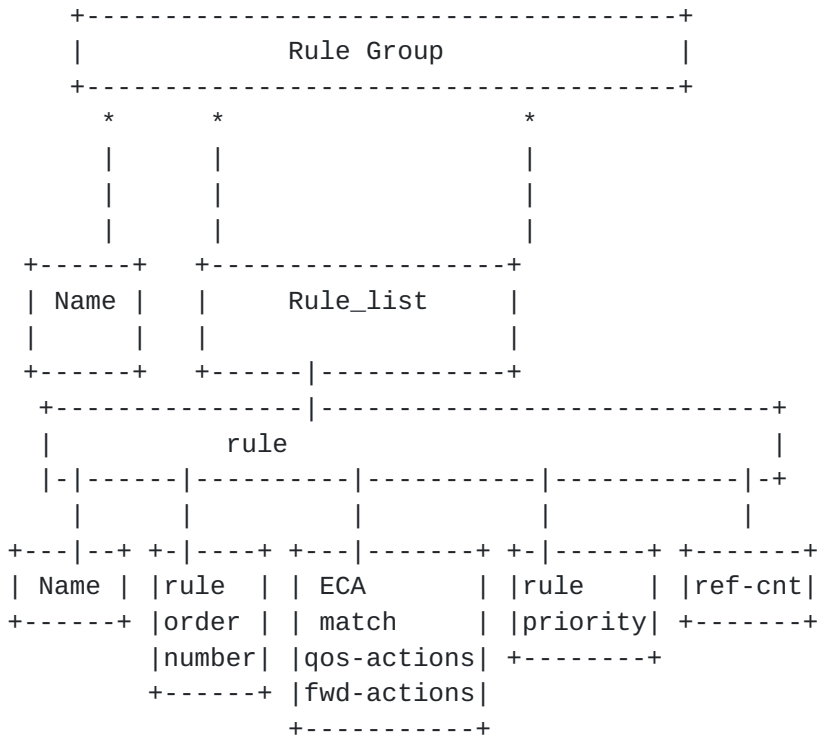
- name that identifies the grouping of policy rules
- module reference - reference to a yang module(s) in the yang module library that this group of policy writes policy to
- list of rules

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.

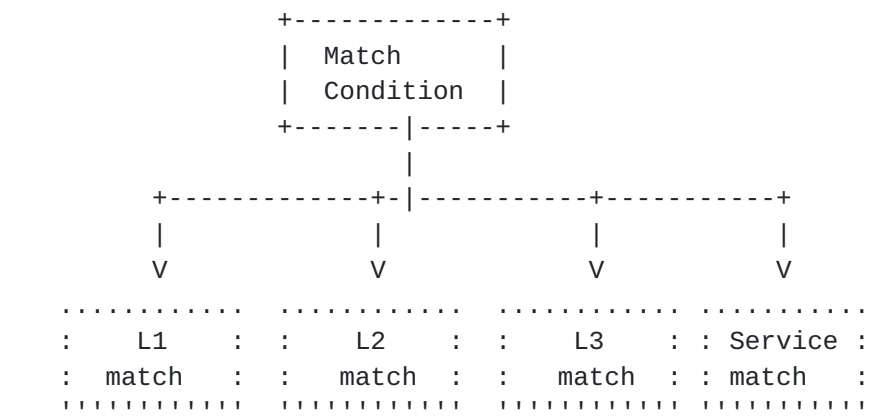
Rules can exist with the same rule order and same priority. Rules with the same rule order and same priority are not guaranteed to be at any specific ordering. The order number and priority have sufficient depth that administrators who wish order can specify it.

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 (L1, L2, L3, service layer) or time of day or packet size. The qos actions can be setting fields in the packet at any layer (L1-L4, service) 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:bnp-eca-policy
 import ietf-inet-types  {prefix "inet"}
 import ietf-interface  {prefix "if"}
 import ietf-i2rs-rib   {prefix "i2rs-rib"}

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

       import ietf-i2rs-rib {
    prefix "i2rs-rib";
```

Below is the high level yang diagram

```
   Packet Reception ECA policy
   module ietf-pkt-eca-policy
     +--rw pkt-eca-policy-cfg
     |  +--rw pkt-eca-policy-set
     |     +--rw groups* [group-name]
     |     |  +--rw vrf-name string
     |     |  +--rw address-family
     |     |  +--rw group-rule-list* [rule-name]
     |     |  |  +--rw rule-name
     |     |  |  +--rw rule-order-id
     |     +--rw rules* [order-id rule-name]
     |        +--rw eca-matches
     |        | ...
     |        +--rw eca-qos-actions
     |        | ...
     |        +--rw eca-fwd-actions
     |        | ...
     +--rw pkt-eca-policy-opstate
        +--rw pkt-eca-opstate
           +--rw groups* [group-name]
           |  +--rw rules-installed;
           |  +--rw rules_status* [rule-name]
           +--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-forwarde
```

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:   bnp-eca-policy-set:rules

Operational State for Policy
=========================================
Policy level: pkt-eca-policy-opstate
group level:  pkt-eca-policy-opstate:groups-status
rule level:   bnp-eca-policy-opstate:rules_opstate*
              bnp-eca-policy-opstate:rules_opstats*
```

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: L1-header-match
  |     |  |  +--case: L2-header-match
  |     |  |  +--case: L3-header-match
  |     |  |  +--case: L4-header-match
  |     |  |  +--case: Service-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 L1-action
     |        |  |  +--case L2-action
     |        |  |  +--case L3-action
     |        |  |  +--case L4-action
     |        |  |  +--case service-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@2016-02-09.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 "i2rs-rib";
       }

  // 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 "2016-02-09" {
           description "initial revision";
           reference "draft-hares-i2rs-pkt-eca-policy-dm-00";
         }


  // interfaces - no identity  matches


  // L1 header match identities
        identity l1-header-match-type {
      description
      " L1 header type for match ";
    }

  identity l1-hdr-sonet-type {
    base l1-header-match-type;
    description
      " L1 header sonet match ";
   }

   identity l1-hdr-OTN-type {
       base l1-header-match-type;
       description
      " L1 header OTN match ";
        }

        identity l1-hdr-dwdm-type {
       base l1-header-match-type;
          description
      " L1 header DWDM match ";
        }
```

```
        // 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 type for GRE tunnel match ";
        }

        // L4 header match identities

        identity l4-header-match-type {
         description "L4 header
         match types. (TCP, UDP,
         SCTP, 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-sctp-header {
          base l4-header-match-type;
          description "L4 header match for SCTP";
         }

         // Service header identities

         identity service-header-match-type {
          description "service header
           match types: service function path
           (sf-path)), SF-chain, sf-discovery,
            and others (added here)";
```

```
         }

          identity sf-chain-meta-match {
            base service-header-match-type;
            description "service header match for
                meta-match header";
        }

          identity sf-path-meta-match {
            base  service-header-match-type;
                description "service header match for
                 path-match header";
        }

        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 L1-header-match {
              choice l1-header-match-type {
                case l1-hdr-sonet-type {
                // sonet matches
                }
                case L1-hdr-OTN-type {
                // OTN matches
                }
                case L1-hdr-dwdm-type {
                // DWDM matches
                }
           description
                 "The Layer 1 header match choices";
              }
           description
                 "The Layer 1 header match includes
```

```
                any reference to L1 technology";
        }

        grouping L1-header-actions {
                 leaf l1-hdr-sonet-act {
                  type uint8;
                  description "sonet actions";
                 }
                 leaf l1-hdr-OTN-act {
                  type uint8;
                  description "OTN actions";
                 }
                 leaf l1-hdr-dwdm-act {
                    type uint8;
                   description "DWDM actions";
                 }
                description "L1 header match
                 types";
         }

        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 i2rs-rib: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 i2rs-rib: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 i2rs-rib: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 i2rs-rib:ipv4-header;
                   }
                   case l3-ipv6-hdr {
                     uses i2rs-rib:ipv6-header;
                   }
                   case L3-gre-tunnel {
                     uses i2rs-rib: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 service-header-match {
                  choice service-header-match-type {
                    case sf-chain-meta-match {
                     description "uses
                     sfc-sfc:service-function-chain-grouping:
                     + sfc-sfc:service-function-chain";
                    }
                     case sf-path-meta-match {
```

```
                    description "uses
                        sfc-spf:service-function-paths:
                        + sfc-spf:service-function-path";
                   }
                description "SFC header match
                 choices";
                }
            description "SFC header and path
                 matches";
        }

        grouping sfc-header-actions {
                choice service-header-match-type {
                  case sf-chain-meta-match {
                    leaf set-chain {
                          type boolean;
                       description "flag to set
                           chain in sfc.  Should
                           be amended to use SFC service
                           chain matching.
                          uses sfc-sfc:service-function-chain-grouping:
                          + sfc-sfc:service-function-chain";
                     }
                  }
                  case sf-path-meta-match {
                    leaf set-path {
                         type boolean;
                     description "flag to set path in
                         sfc header. Amend to use sfc-spf
                          function headers. Uses
                          sfc-spf:service-function-paths:
                          + sfc-spf:service-function-path.";
                    }
                  }
                description "choices in SFC for
                 chain match and path match.";
                }
          description "modify action for
            SFC header.";
        }


   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 l1-size-match {
            type uint32;
            description "L1 packet match size.";
                 }
         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.";
                }
         leaf service-meta-size {
            type uint32;
            description "service meta info match size.";
                }
         leaf service-meta-payload {
                type uint32;
            description "service meta-play match size";
           }
     description "packet size by layer
         only non-zero values are matched";
        }

        grouping time-day-match {

        description "matches for
          time of day.";
        }

     grouping eca-matches {
           uses interface-match;
           uses L1-header-match;
           uses L2-header-match;
```

```
          uses L3-header-match;
          uses L4-header-match;
          uses service-header-match;
          uses packet-size-match;
          uses time-day-match;
          description "ECA matches";
        }

        grouping eca-qos-actions {
          leaf cnt-actions {
                 type uint32;
             description "count of ECA actions";
          }
          uses interface-actions;
          uses L1-header-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,
                 L1, L2, L3, L4 nad service forwarding
                 headers.";
         }

         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-fwd-actions {
      leaf interface-fwd {
             type if:interface-ref;
             description "name of interface to forward on";
          }
      uses i2rs-rib:nexthop;
          uses ip-next-fwd;
          leaf drop-packet {
           type boolean;
```

```
           description "drop packet flag";
          }
          description "ECA forwarding actions";
        }


  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 eca-rule-name";
                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-matches;
                 uses eca-qos-actions;
                 uses eca-fwd-actions;
                 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 used in the I2RS FB-RIBs to filter packets in a traffic stream, act to modify packets, and forward data packets. These I2RS filters operate dynamically at same level as currently deployed configured filter-based RIBs to filter, change, and forward traffic. The dynamic nature of this protocol requires that I2RS Filters track the installer of group information and rules.

This section will be augmented after a discussion with security experts.

8. Informative References

[I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-15 (work in progress), April 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-08 (work in progress), October 2015.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-13 (work in progress), April 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-07 (work in progress), March 2016.

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

[RFC3060] Moore, B., Ellesson, E., Strassner, J., and A. Westerinen, "Policy Core Information Model -- Version 1 Specification", RFC 3060, DOI 10.17487/RFC3060, February 2001, <http://www.rfc-editor.org/info/rfc3060>.

[RFC3460] Moore, B., Ed., "Policy Core Information Model (PCIM) Extensions", RFC 3460, DOI 10.17487/RFC3460, January 2003, <http://www.rfc-editor.org/info/rfc3460>.

[RFC3644] Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., and B. Moore, "Policy Quality of Service (QoS) Information Model", RFC 3644, DOI 10.17487/RFC3644, November 2003, <http://www.rfc-editor.org/info/rfc3644>.

Authors' Addresses

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Russ White
Ericsson

Email: russw@riw.us