I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2016

S. Hares
Q. Wu
Huawei
J. Tantsura
R. White
Ericsson

July 1, 2015

An Information Model for Basic Network Policy and Filter Rules
draft-hares-i2rs-bnp-eca-data-model-00.txt

Abstract

This document contains the Basic Network Policy and Filters (BNP IM) Data Model which provides a policy model that support an ordered list of match-condition-action (aka event-condition-action (ECA)) for multiple layers (interface, L1-L4, application) and other factors (size of packet, time of day). The actions allow for setting actions (QOS and other), decapsulation, encapsulation, plus forwarding actions. The policy model can be used with the I2RS filter-based RIB.

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 January 2, 2016.

Copyright Notice

Table of Contents

1. Introduction

This generic network policy provide a model to support an ordered list of routing policy or an ordered list of filter rule. ne examples of the ordered-based filters is the I2RS Filter-based RIBs, and another is flow-specification filters. 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.

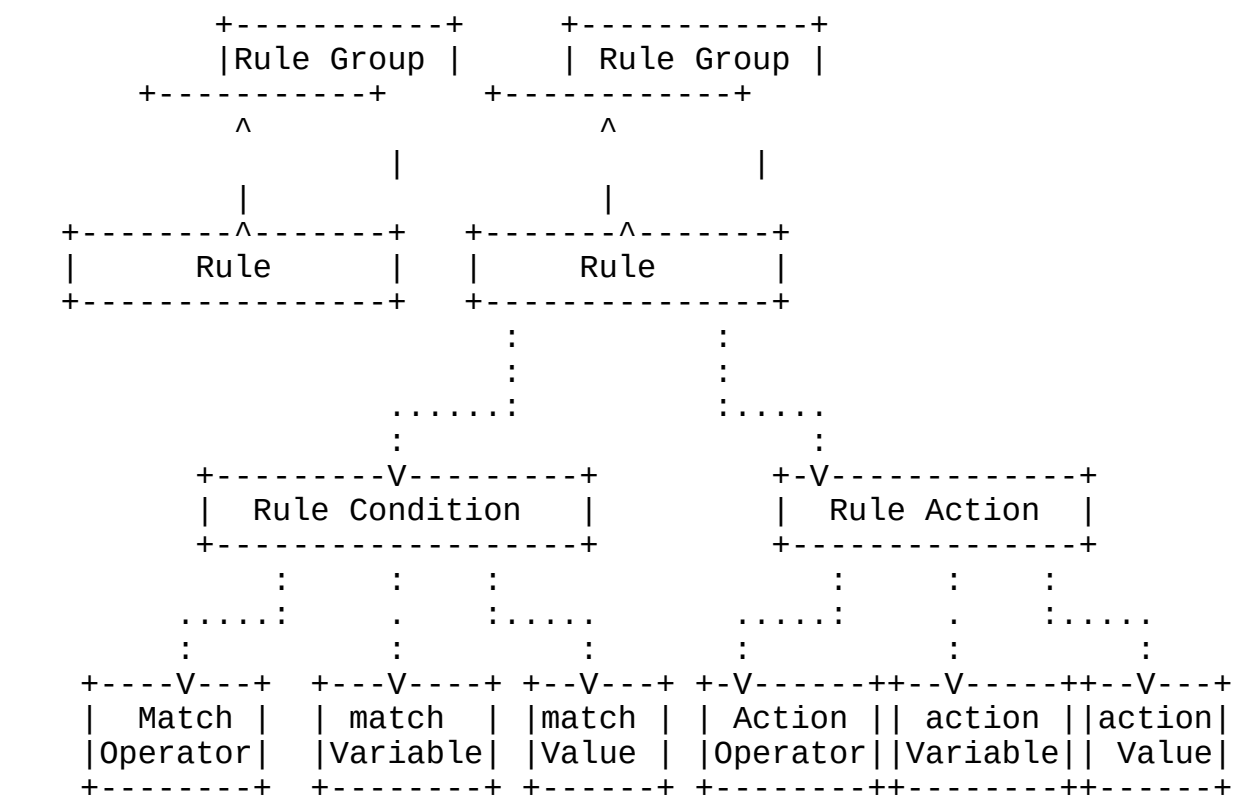


Figure 1: BNP structure

3. BNP Rule Groups

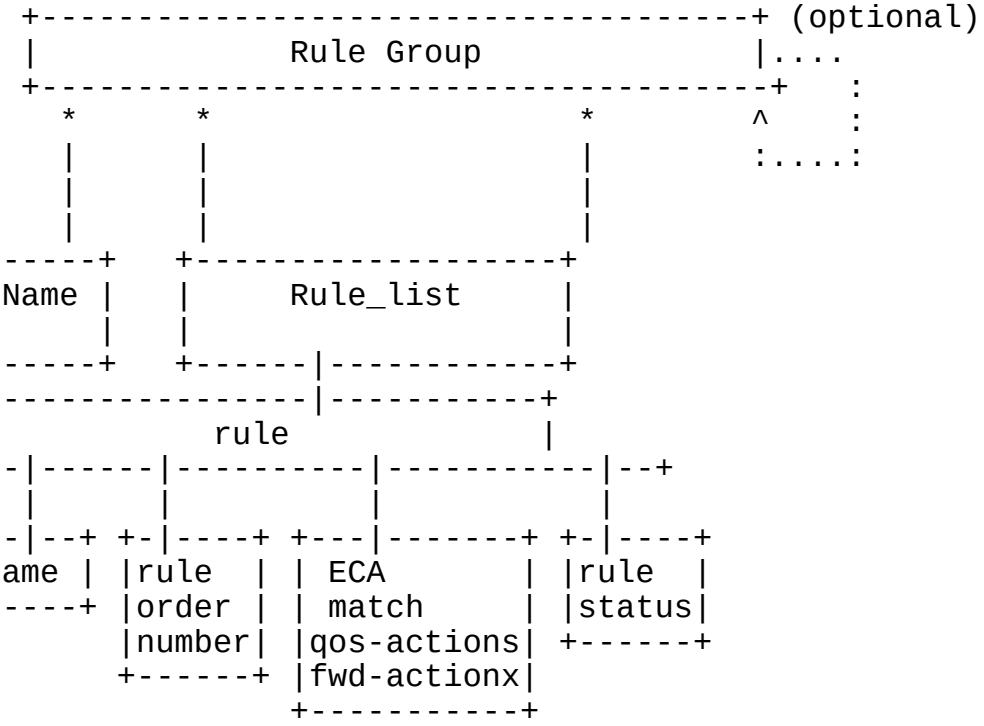
Rule groups have the following elements:

- o name that identifies the grouping of policy rules
- o role - that is a combination of target resource (E.g. IPv4 FB-FIB filters) and a scope (read, read-write, write-only).
- o list of rules

The rule has the following elements: name, order, status, priority, reference cnt, and match-action as shown as shown in figure 2. The order indicates the order of the rule within the 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.

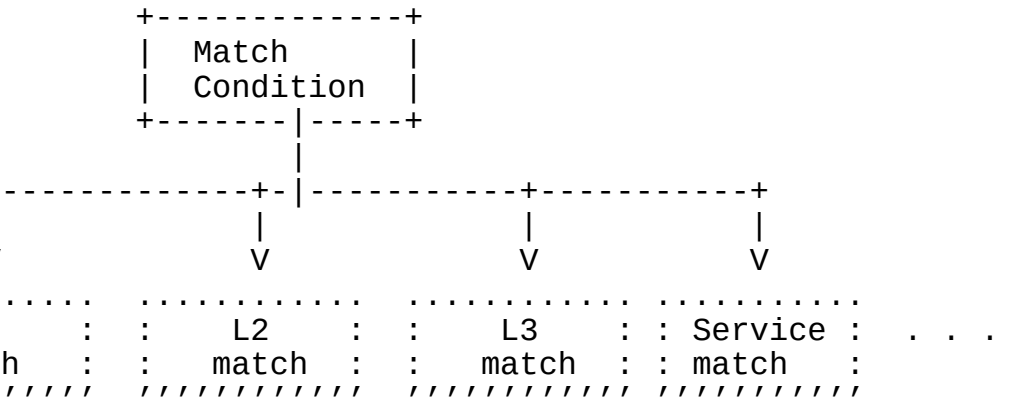
The generic rules can be included with other types of rules as figure 2 shows.

Figure 2 - Rule Group



The generic match conditions are specific to a particular layer are refined by matches to a specific layer (as figure 4 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 4



4. BNP Generic Info Model in High Level Yang

Below is the high level inclusion

Figure 5

```
module:bnp-eca-policy
 import ietf-inet
 import ietf-interface
 import ietf-i2rs-rib
 import service-function-type prefix-sft
 import service-function prefix-sf
 import service-fucntion-chain prefix-sfc-sfc
```

Below is the high level yang diagram

```
   module:bnp-eca-policy
     +--bnp-eca-policy
        +--rw rule-group* [group-name]
               +--rw group-name
           +--rw rule*  [rule-name]
              +--rw rule-name string
              +--rw order unit16
                      +--rw installer
              +--rw rule-match-act
              |  +--rw bnp-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
              |  +--rw bnp-action
              |  |  +--rw number-actions
                        |  |  |  +--case interface-actions
              |  |  |  +--case L1-action
              |  |  |  +--case L2-action
              |  |  |  +--case L3-action
              |  |  |  +--case L4-action
              |  |  |  +--case service-action
              |  |  +--rw bnp-forward
              |  |  |  +--rw interface interface-ref
              |  |  |  +--rw next-hop  rib-nexthop-ref
              |  |  |  +--rw route-attributes
              |  |  |  +--rw rib-route-attributes-ref
              |  |  +--rw fb-std-drop
                  +--rw rule_status
              |  +--ro rules-status
              |  +--ro rule-inactive-reason
              |  +--ro rule-installer
              |  +--ro refcnt unit16
```

5. bnp-eac-policy Yang module

```
        file "bnp-eca-policy.yang";

   module bnp-eca-policy {

  yang-version "1";

  // namespace
```

```
  namespace "urn:TBD1:params:xml:ns:yang:rt:i2rs-bnp-eca-pol";
    // replace with iana namespace when assigned
    prefix "bnp-eca-pol";

  // import some basic inet types

  import ietf-inet-types { prefix "inet"; }  // RFC6991
  import ietf-yang-types { prefix "yang"; }
  import ietf-interfaces { prefix "if"; }
  import ietf-routing {prefix "rt"; }
  import i2rs-rib { prefix "i2rs-rib";}
  import service-function-type {prefix-sft;}
  import service-function {prefix "sfc-sf";}
  import service-function-type {prefix "sfc-sft";}
  import service-funion-chain {prefix "sfc-sfc";}

  // meta
  organization
    "TBD2""

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

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

       revision "2015-07-01" {
          description "initial revision"
          references "draft-hares-i2rs-bnp-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;
   }

   identity l1-hdr-OTN-type {
```

```
    base l1-header-match-type;
        }

        identity l1-hdr-dwdm-type {
        base l1-header-match-type;
        }

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

  identity l2-802-1Q {
    base l2-header-match-type;
   }

   identity l2-802-11 {
    base l2-header-match-type;
        }

        identity l2-802-15 {
        base l2-header-match-type;
        }

        identity l2-NVGRE {
        base l2-header-match-type;
        }
        identity l2-MPLS {
        base l2-header-match-type;
        }

        identity l2-VXLAN {
        base l2-header-match-type;
        }

        identity l2-mpls {
        base l2-header-match-type;
        }

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

        identity l3-ipv4-hdr {
         base l3-header-match-type;
```

```
        }

        identity l3-ipv6-hdr {
         base l3-header-match-type;
        }

        identity l3-gre-tunnel {
         base l3-header-match-type;
        }

        // 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;
         }

         identity l4-udp-header {
          base l4-header-match-type;
         }

         identity l4-sctp-header {
          base l4-header-match-type;
         }

         // 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;
         }

          identity sf-path-meta-match {
            base  service-header-match-type;
          }
```

```
     grouping interface-match {
          description "interface
           has name, description, type, enabled
           as potential matches";

          uses if:interfaces:interface
         }

         grouping interface-action {
           description
           "interface action up/down and
            enable/disable"
                leaf interface-up {type boolean;}
                leaf interface-down {type boolean;}
                leaf interface-enable {type boolean;}
                leaf interface-disable {type boolean;}
                }
                }

         grouping L1-header-match {
            description
                    "The Layer 1 header match includes
                     any reference to L1 technology";
               // matches for OTN, SDH, DWDM
                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
                  }
                }
         }

         grouping L1-header-actions {
          choice l1-header-match-type {
                  case: l1-hdr-sonet-type {
                  // sonet actions
                  }
                  case: L1-hdr-OTN-type {
                  // OTN actions
                  }
                  case: L1-hdr-dwdm-type {
                  // DWDM actions
         }
```

```
        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;}
                 leaf Qtag1-present{type Boolean;}
                 leaf Qtag2-present {type Boolean;}

                 leaf dest-mac { type uint48;}
                 leaf src-mac {type uint48;}
                 leaf vlan-tag {type uint16;}
                 leaf Qtag1 {type uint32;}
                 leaf QTag2 {type uint32;}
                 leaf ethertype {type uint16;}
                }


        group L2-VXLAN-header {
           description
               "This vXLAN header may
               be replaced by actual VXLAN yang
               module reference";
                container {
                  leaf outer-mac-header {
                    uses L2-802_1Q-header;
                        }
                  leaf outer-ip-header {
                    uses i2rs-rib:ipv4-header;
                        }
                  leaf vxlan-network-id {
                    uint32;
                  }
                  leaf inner-mac-header {
                    uses L2-802_1Q-header;
                  }
               }

        group L2-NVGRE-header {
           description
               "This NVGRE header may
               be replaced by actual NVGRE yang
               module reference";
                container {
                  leaf outer-mac-header {
```

```
                  uses L2-802_1Q-header;
                      }
                leaf outer-ip-header {
                  uses i2rs-rib:ipv4-header;
                      }
                leaf gre-version {
                  type uint8;
                }
                leaf gre-proto {
                  type uint16;
                      }
                leaf virtual-subnet-id {
           type uint32;
           }
         leaf flow-id {
           type uint16
                }
                leaf inner-mac-header {
                  uses L2-802_1Q-header;
                }
              }


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

```
        }
        grouping L2-header-actions {
            description
                " The layer 2 header match includes
                  any reference to L2 technology"
                  choice l2-header-match-type {
                    case l2-802-1Q {
                        // actions for L2-802-1Q
                        }
                        case l2-802-11 {
                          // actions for L2-802-11
                          }
                        case l2-802-15 {
                     // actions 802.1 Ethernet
                    }
                        case l2-NVGRE {
                         // actions for NVGRE
                          leaf set-vsid {type boolean;}
                          leaf set-flowid {type boolean;}
                          leaf vsi {type unit32;}
                          leaf flow-id {type uint16; }
                        }
                        case l2-VXLAN-header {
                          leaf set-network-id {type boolean;}
                          leaf network-id {type uint32;}
                        }
                        case l2-mpls-header {
                          leaf pop {type boolean;}
                          leaf push {type boolean;}
                          mpls label {uint32;}
                        }
               }
        }

        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;
                  }
                }
        }
        grouping ipv4-encapsulate-gre {
```

```
            leaf encapsulate {
                  type boolean;}
            leaf ipv4-dest-address
                        {type inet:ipv4-address;}
                leaf ipv4-source-address
            { type inet-ipv4-address;}
        }

        grouping l3-header-actions {
               choice l3-header-match-type {
                 case l3-ipv4-hdr {
                     leaf set-ttl {
                            type uint8; }
                       leaf set-dscp {
              type uint8; }
                       leaf encapsulate {
                         }
           }
                  case l3-ipv6-hdr {
                    leaf set-next-header {
                          type boolean;}
                        leaf set-traffic-class {
                          type boolean;}
                        leaf set-flow-label {
                          type boolean;}
                        leaf set-hop-limit {
                          type boolean;}
                        leaf next-header {
                               type uint8;}
                         leaf traffic-class {
                               type uint8;
                               }
                       leaf flow-label {
                               type uint16;
                               }
                       leaf hop-limit {
                               type uint8;
                       }
                  }

                  case L3-gre-tunnel {
                    leaf decapsulate {
                         type boolean; }
                  }
                }
               }
        }
```

```
         grouping tcp-header-match {
           leaf source-port {
             type uint16; }
           leaf dest-port {
             type uint16;
                 }
                 leaf sequence-number {
                 type uint32
                 }
                 leaf ack-number {
                 type uint32
                 }
           }

         grouping tcp-header-action {
           leaf set-source-port {
            type boolean;}
           leaf set-dest-port {
            type boolean;
           }
           uses tcp-header-match;
           }

         grouping udp-header-match {
            leaf source-port {
                 type uint16;
                 }
                leaf dest-port {
                 type unit16;
                 }
       }

       grouping udp-header-action {
            leaf set-source-port {
                 type boolean;
                 }
                leaf set-dest-port {
                 type boolean;}
                 uses udp-header-match;
                }

       group sctp-chunk {
         leaf chunk-type {
           type uint8;
               }
         leaf chunk-flag {
           type uint8;
               }
```

```
          leaf chunk-length {
            type uint16;
          }
          leaf chunk-data-0 {
           type uint32
          }
        }

         grouping sctp-header-match {
            leaf source-port {
                 type uint16;
                 }
                leaf dest-port {
                 type unit16;
                 }
                 leaf verification-tag {
                 type unit32;
                 }
                 leaf chunk1 {
                  uses sctp-chunk
                 }
        }
        group sctp-header-action {
            leaf set-source-port {
                 type boolean;}
                leaf set-dest-port {
                 type boolean;}
                 leaf set-chunk1 {
                  type boolean;}
                  uses sctp-header-match;
                }


         grouping l4-header-match {
             choice L3-header-match-type {
                   case l4-tcp-header {
                     use tcp-header-match;
                   }
                   case l4-udp-header {
                     uses udp-header-match;
                   }
                   case l4-sctp {
                     uses sctp-header-match;
                   }
                 }
         }
```

```
         grouping l4-header-action {
             choice L3-header-match-type {
                   case l4-tcp-header {
                     use tcp-header-action;
                   }
                   case l4-udp-header {
                     uses udp-header-action;
                   }
                   case l4-sctp {
                     uses sctp-header-action;
                   }
                 }
         }

    grouping service-header-match {
                choice service-header-match-type {
                  case sf-chain-meta-match {
                   uses sfc-sfc:service-function-chain-grouping:service-func
tion-chain;
                  }
                   case sf-path-meta-match {
                    uses sfc-spf:service-function-paths:service-function-pat
h;
                   }
                }
        }
        grouping service-header-actions {
                choice service-header-match-type {
                  case sf-chain-meta-match {
                    leaf set-chain {
                         type boolean;}
                  uses sfc-sfc:service-function-chain-grouping:service-funct
ion-chain;
                  }
                   case sf-path-meta-match {
                    leaf set-path {
                         type boolean;}
                uses sfc-spf:service-function-paths:service-function-path;
                   }
                }
        }



   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;
   }

   identity rule-status-valid {
       base rule-status-type;
        }

   identity rule-status-valid-installed {
       base rule-status-type;
   }
      identity rule-status-valid-inactive {
       base rule-status-type;
   }


   grouping rule_status {
     description
         "rule operational status";
          leaf rule-status {
            type rule-status type;
          }
          leaf rule-status-inactive {
            type string;
          }
          leaf rule-status-installer {
            type string;
          }
          leaf refcnt uint64;
   }

   grouping packet-size-match {
     description "packet size by layer
         only non-zero values are matched";
         leaf l1-size-match {type uint32;}
         leaf l2-size-match {type uint32;}
         leaf l3-size-match {type uint32;}
         leaf l4-size-match {type uint32;}
         leaf service-meta-size {type uint32;}
         leaf service-meta-payload {type uint32}
         }

        grouping time-day-match {
        //matches for time of day;
        }


    container bnp-eca-matches {
          description "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;
          uess time-day-match;
        }

        container bnp-eca-qos-actions {
         description "ECA set or change
         packet Actions"
          leaf cnt-actions;
          uses interface-actions;
          uses L1-action;
          uses L2-action;
          uses L3-actions;
          uses L4-actions;
          uses service-actions;
         }

         container ip-next-fwd {
          leaf rib-name {
           type string;
            }
          leaf next-hop-name {
            type string;
                }
         }

        container bnp-fwd-actions {
         description "ECA forwarding
         actions"
      leaf interface-fwd if:interfaces:interface:name;
          leaf i2rs-next-hop i2rs-rib:nexthop-ref;
          leaf rib-next-hop {
           use ip-next-fwd;
           }
          leaf drop-packet {
           type boolean;
          }
        }

  bnp-ecap-policy-set {
   description
   " main bnp ecap policy"
```

```
    container groups {
          list rule-group {
            key "group-name";
            description
            "groups of ECA rules";

                leaf group-name {
                  type string;
                  description
                   "name of group of rules";
                  };

                list rule {
                 key "rule-name"
                 description "ECA rules";
                 leaf rule-name
                   {type string;
                    description "name of rule";}
                 leaf order-id
                   {type uint16;
                        description
                        "Number of order
                        in ordered list (ascending)"
                       }
                 leaf installer
                   {type string;
                        description
                        "Id of I2RS client
                         that installs this rule;"
                       }
                 uses bnp-eca-matches;
                 uses bnp-eca-qos-actions;
                 uses bnp-eca-fwd-actions;
                }  // end of rule
          }  // end of group
        }  // end of policy group
   }
```

6. IANA Considerations

This draft includes no request to IANA.

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.hares-i2rs-usecase-reqs-summary]
Hares, S. and M. Chen, "Summary of I2RS Use Case Requirements", draft-hares-i2rs-usecase-reqs-summary-02 (work in progress), May 2015.

[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-09 (work in progress), March 2015.

[I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Folkes, R., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-06 (work in progress), March 2015.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-06 (work in progress), June 2015.

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Sreenivasa, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-03 (work in progress), June 2015.

[I-D.zhdankin-idr-bgp-cfg]
Alex, A., Patel, K., Clemm, A., Hares, S., Jethanandani, M., and X. Liu, "Yang Data Model for BGP Protocol", draft-zhdankin-idr-bgp-cfg-00 (work in progress), January 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3060] Moore, B., Ellesson, E., Strassner, J., and A. Westerinen, "Policy Core Information Model -- Version 1 Specification", RFC 3060, February 2001.

[RFC3460] Moore, B., "Policy Core Information Model (PCIM) Extensions", RFC 3460, January 2003.

[RFC3644] Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., and B. Moore, "Policy Quality of Service (QoS) Information Model", RFC 3644, November 2003.

[RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, April 2009.

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

Jeff Tantsura
Ericsson

Email: Jeff Tantsura jeff.tantsura@ericsson.com

Russ White
Ericsson

Email: russw@riw.us