I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

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

October 19, 2015

An Information Model for Basic Network Policy and Filter Rules
draft-hares-i2rs-bnp-eca-data-model-02.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

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.

```
           +-----------+     +------------+
           |Rule Group |     | Rule Group |
       +-----------+     +------------+
            ^                  ^
                    |                  |
            |                  |
   +--------^-------+   +-------^-------+
   |      Rule      |   |     Rule      |
   +----------------+   +---------------+
                          :          :
                          :          :
                    ......:          :.....
                    :                     :
          +---------V---------+         +-V-------------+
          |  Rule Condition   |         |  Rule Action  |
          +-------------------+         +---------------+
              :     :    :                 :     :    :
         .....:     .    :.....       .....:     .    :.....
         :          :         :       :          :         :
    +----V---+  +---V----+ +--V---+ +-V------++--V-----++--V---+
    |  Match |  | match  | |match | | Action || action ||action|
    |Operator|  |Variable| |Value | |Operator||Variable|| Value|
    +--------+  +--------+ +------+ +--------++--------++------+
```

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
        +-------------------------------------+ (optional)
        |             Rule Group              |....
        +--------------------------------------+   :
          *      *                   *        ^    :
          |      |                   |        :....:
          |      |                   |
          |      |                   |
     +------+   +-------------------+
     | Name |   |     Rule_list     |
     |      |   |                   |
     +------+   +------|------------+
      +----------------|-----------------+
      |           rule                   |
      |-|------|----------|-----------|--+
        |      |          |           |
    +---|--+ +-|----+ +---|-------+ +-|----+
    | Name | |rule  | | ECA       | |rule  |
    +------+ |order | | match     | |status|
             |number| |qos-actions| +------+
             +------+ |fwd-actionx|
                      +-----------+
```

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
                    +-------------+
                    |  Match      |
                    |  Condition  |
                    +-------|-----+
                            |
            +-------------+-|-----------+-----------+
            |             |             |           |
            V             V             V           V

      ............  ............  ............ ...........
      :    L1    :  :    L2    :  :    L3    : : Service :  . . .
      :  match   :  :   match  :  :   match  : : match   :
      ''''''''''''  ''''''''''''  '''''''''''' '''''''''''
```

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:i2rs-eca-policy
     +--i2rs-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. i2rs-eca-policy Yang module

```
  //<CODE BEGINS> file "i2rs eca-policy@2015-10-18.yang"

   module i2rs-eca-policy
   {
    namespace "urn:ietf:params:xml:ns:yang:i2rs-eca-policy";
   // replace with iana namespace when assigned
   prefix "i2rs-eca";
  // import some basic inet types

  import ietf-inet-types { prefix "inet"; }  // RFC6991
  import ietf-interfaces { prefix "if"; }
  import i2rs-rib { prefix "i2rs-rib";}

  // meta
  organization
    "IETF I2RS WG";

  contact
     "email: shares@ndzh.com
          email: russ.white@riw.com
          email: jeff.tantsura@ericsson.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 "2015-10-18" {
           description "initial revision";
           reference "draft-hares-i2rs-bnp-eca-policy-dm-01";
         }

  // interfaces - no identity  matches


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

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

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

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

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

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

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

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

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

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


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

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

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

        identity l3-gre-tunnel {
     description
      " l3 header type for GRE tunnel match ";
         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 {
      description "L4 header for TCP";
          base l4-header-match-type;
         }

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

         identity l4-sctp-header {
          description "L4 header match for SCTP";
          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 {
            description "service header match for
                meta-match header";
            base service-header-match-type;
         }

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

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

           leaf match-if-name {
            description "match on interface name";
                type if:interface-ref;
           }
           // add description later
         }

         grouping interface-action {
           description
           "interface action up/down and
            enable/disable";
                leaf interface-up {
                 description
                  "action to put interface up";
                 type boolean;
                 }
                leaf interface-down {
                 description
                  "action to put interface down";
                  type boolean;
                  }
                leaf interface-enable {
                   description
                  "action to enable interface";
                  type boolean;
                  }
                leaf interface-disable {
                 description
                  "action to disable interface";
                  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 {
                    description " Include VLAN in header";
                    type boolean;
                        }
                  leaf qtag1-present {
                     description " This flag value indicates
                         inclusion of one 802.1Q tag in header";
                    type boolean;
                        }
                  leaf qtag2-present{
                      description "This flag indicates the
                          inclusion of second 802.1Q tag in header";
                    type boolean;
                    }

                  leaf dest-mac {
           description "IEEE destination MAC value
                      from the header";
                       type uint64;
                           //change to uint48
                   }
                  leaf src-mac {
                   description "IEEE source MAC
                    from the header";
                    type uint64;
                        //change to uint48
                        }
                  leaf vlan-tag {
                   description "IEEE VLAN Tag
                    from the header";
                        type uint16;
                        }
                  leaf qtag1 {
                    description "Qtag1 value
                         from the header";
                        type uint32;
                        }
                  leaf qtag2 {
                         description "Qtag1 value
                         from the header";
                     type uint32;
                  }
                  leaf L2-ethertype {
                    description "Ether type
                        from the header";
                        type uint16;
                  }
           }


         grouping L2-VXLAN-header {
            description
                "This VXLAN header may
                be replaced by actual VXLAN yang
                module reference";
                 container vxlan-header {
                 //vix outer mac header
                    uses i2rs-rib:ipv4-header;
                    leaf vxlan-network-id {
                      description "VLAN network id";
                      type uint32;
                     }
                  }
                  //fix inner header here
         }

         grouping L2-NVGRE-header {
            description
                "This NVGRE header may
                be replaced by actual NVGRE yang
                module reference";
                 container nvgre-header {
                    uses L2-802-1Q-header;
                    uses i2rs-rib:ipv4-header;
                  leaf gre-version {
                    description "L2-NVGRE GRE version";
                    type uint8;
                    }
                  leaf gre-proto {
                    description "L2-NVGRE protocol value";
                    type uint16;
                        }
                  leaf virtual-subnet-id {
                     description "L2-NVGRE subnet id value";
             type uint32;
             }
          leaf flow-id {
                     description "L2-NVGRE Flow id value";
             type uint16;
                  }
                    // 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 {
                            description
                                "Boolean flag to set VSID in packet";
                                type boolean;
                                }
                           leaf set-flowid {
                              description
                                   "Boolean flag to set VSID in packet";
                              type boolean;
                                }
                           leaf vsi {
                                 description
                                  "VSID value to set in packet";
                              type uint32;
                                }
                           leaf flow-id {
                             description
                                 "flow-id value to set in packet";
                           type uint16;
                           }
                         }
                         case l2-VXLAN-header {
                           leaf set-network-id {
                            description
                                 "flag to set network id in packet";
                              type boolean;}
                           leaf network-id {
                             description
                                 "network id value to set in packet";
                           type uint32;
                           }
                         }
```

```
                           case l2-mpls-header {
                             leaf pop {
                              description
                                   "Boolean flag to pop mpls header";
                             type boolean;
                             }
                             leaf push {
                              description
                                   "Boolean flag to push value into
                                   mpls header";
                             type boolean;
                             }
                             leaf mpls-label {
                                   description
                                   "mpls label to push in header";
                                   type uint32;
                                  }
                           }
                  }
           }
           grouping L3-header-match {
               description "match for L3 headers";
               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 {
               description "encapsulation actions for IPv4 headers";
               leaf encapsulate {
               description "flag to encapsulate headers";
                     type boolean;
                   }
               leaf ipv4-dest-address {
                          description "Destination Address for GRE header";
                          type inet:ipv4-address;
                  }
                   leaf ipv4-source-address {
                       description "Source Address for GRE header";
                           type inet:ipv4-address;
                  }
```

```
         }

         grouping l3-header-actions {
            description "actions that can
                 be performed on header";

                choice l3-header-act-type {
                  case l3-ipv4-hdr {
                      leaf set-ttl {
                          description "flag to set TTL";
                          type boolean;
                          }
                          leaf set-dscp {
                          description "flag to set DSCP";
                          type boolean;
                          }
                      leaf ttl-value {
                                description "TTL value to set";
                             type uint8;
                                 }
                          leaf dscp-val {
                                description "dscp value to set";
                type uint8;
                          }
            }
                   case l3-ipv6-hdr {
                     leaf set-next-header {
                           description
                           "flag to set next routing header in IPv6 header";
                           type boolean;
                           }
                         leaf set-traffic-class {
                           description
                           "flag to set traffic class in IPv6 header";
                           type boolean;
                           }
                         leaf set-flow-label {
                           description
                           "flag to set flow label in IPv6 header";
                           type boolean;}
                         leaf set-hop-limit {
                           type boolean;
                           }
                         leaf next-header {
                           description "value to set in next header";
                                type uint8;}
                          leaf traffic-class {
                           description "value for traffic class";
```

```
                              type uint8;
                              }
                      leaf flow-label {
                         description "value for flow label";
                              type uint16;
                              }
                      leaf hop-limit {
                         description "value for hop count";
                              type uint8;
                      }
                 }

                 case L3-gre-tunnel {
                   leaf decapsulate {
                 description "flag to decapsulate packet";
                        type boolean; }
                 }
               }
        }


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

         grouping tcp-header-action {
           leaf set-source-port {
           description "flag to set source port value";
            type boolean;}
           leaf set-dest-port {
           description "flag to set source port value";
            type boolean;
           }
           uses tcp-header-match;
           }

         grouping udp-header-match {
            leaf source-port {
             description "UDP source port match value";
                 type uint16;
                 }
                leaf dest-port {
           description "UDP Destination port match value";
                 type uint16;
                 }
        }

        grouping udp-header-action {
            leaf set-source-port {
             description "flag to set UDP source port match value";
                 type boolean;
                 }
                leaf set-dest-port {
             description
                   "flag to set UDP destination port match value";
                 type boolean;
                 }
           uses udp-header-match;
        }

        grouping sctp-chunk {
          leaf chunk-type {
           description "sctp chunk type value";
            type uint8;
                }
          leaf chunk-flag {
           description "sctp chunk type flag value";
            type uint8;
                }
          leaf chunk-length {
           description "sctp chunk length";
            type uint16;
           }
          leaf chunk-data-0 {
           description "byte zero of stcp chunk data";
           type uint32;
           }
        }

         grouping sctp-header-match {
            leaf source-port {
             description "sctp header match source port value";
                 type uint16;
                 }
                leaf dest-port {
             description "sctp header match destination port value";
                 type uint16;
                 }
                 leaf verification-tag {
             description "sctp header match verification tag value";
                 type uint32;
                 }
                  uses sctp-chunk;
        }
        grouping sctp-header-action {
            leaf set-source-port {
             description "set source port in sctp header";
                 type boolean;
                 }
                leaf set-dest-port {
             description "set destination port in sctp header";
                 type boolean;
                 }
                 leaf set-chunk1 {
             description "set chunk value in sctp header";
                  type boolean;
                 }
                  uses sctp-header-match;
        }


         grouping L4-header-match {
             choice l3-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;
                   }
                 }
         }


         grouping l4-header-action {
             choice L3-header-match-type {
                   case l4-tcp-header {
```

```
                   uses 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-f
unction-chain;
                  }
                   case sf-path-meta-match {
                    // uses sfc-spf:service-function-paths:service-function-
path;
                   }
                }
        }
        grouping service-header-actions {
                choice service-header-match-type {
                  case sf-chain-meta-match {
                    leaf set-chain {
                     description "flag to set chain in sfc";
                         type boolean;}
                  //uses sfc-sfc:service-function-chain-grouping:service-fun
ction-chain;
                  }
                   case sf-path-meta-match {
                    leaf set-path {
                     description "flag to set path in sfc header";
                         type boolean;}
               // uses sfc-spf:service-function-paths:service-function-path;
                   }
                }
        }




    grouping rule_status {
      description
          "rule operational status";
           leaf rule-status {
             type string;
           }
           leaf rule-status-inactive {
```

```
            description "description of why rule is inactive";
            type string;
          }
          leaf rule-status-installer {
             description "response on rule installed";
            type string;
          }
          leaf refcnt {
             description "refernce count on rule. ";
           type uint64;
      }
        }

   grouping packet-size-match {
     description "packet size by layer
         only non-zero values are matched";
         leaf l1-size-match {
             description "L1 packet match size.";
             type uint32;
                 }
         leaf l2-size-match {
                description "L2 packet match size.";
                type uint32;
                }
         leaf l3-size-match {
            description "L3 packet match size.";
                type uint32;
                }
         leaf l4-size-match {
            description "L4 packet match size.";
                type uint32;
                }
         leaf service-meta-size {
            description "service meta info match size.";
                type uint32;
                }
         leaf service-meta-payload {
            description "service meta-play match size";
           type uint32;
           }
        }

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


    grouping 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;
          uses time-day-match;
        }

        grouping eca-qos-actions {
         description "ECA set or change
         packet Actions";
          leaf cnt-actions {
             description "count of ECA actions";
                 type uint32;
          }
          // actions may be added for interface,
     // L1, L2, L3, and L4 and service forwarding.
         }

         grouping ip-next-fwd {
          leaf rib-name {
           description "name of RIB";
           type string;
            }
          leaf next-hop-name {
            description "name of next hop";
            type string;
                }
         }

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

  container bnp-ecap-policy-set {
   description
   " main bnp ecap policy";

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

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

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

     //<CODE ENDS>
```

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., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-07 (work in progress), September 2015.

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

[RFC5511] Farrel, A., "Routing Backus-Naur Form (RBNF): A Syntax Used to Form Encoding Rules in Various Routing Protocol Specifications", RFC 5511, DOI 10.17487/RFC5511, April 2009, <http://www.rfc-editor.org/info/rfc5511>.

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