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.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
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 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:

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

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.
4. BNP Generic Info Model in High Level Yang

Below is the high level inclusion

```
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 string
         +--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
   +--ro eca-status
      +--group_status* [name]
         +--ro name string
            +--ro rule-status* [rule-name]
               +--ro rule-name string
               +--ro rule-status
               +--ro rule-inactive-reason
               +--ro rule-installer
               +--ro refcnt unit16

5. i2rs-eca-policy Yang module

//>CODE BEGINS< file "ietf-bnp-eca-policy@2016-01-03.yang"

   module ietf-bnp-eca-policy {

// replace with iana namespace when assigned
prefix "bnp-eca";

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 "2016-01-03" {
    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 {
        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 ";
}

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

identity 12-VXLAN {
base l2-header-match-type;
   description
   "12 header type for VXLAN match ";
 }

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

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

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

identity l3-gre-tunnel {
   base l3-header-match-type;
   description
   "13 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-action {
    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 {
  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
    }
description "Choices for L1 header match types";
  }
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...";
  }

  leaf qtag2-present {
    type boolean;
    description "This flag value indicates...";
  }

leaves "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";
        }
    }
}

grouping L2-header-match {

    choice 12-header-match-type {
        case 12-802-1Q {
            // uses L2-802-1Q-header;
            }
    }

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-header-actions {
  choice 12-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;
        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";
}
}
case l2-VXLAN-header {
  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";
  }
}
case l2-mpls-header {
  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 "L2 header choices for 802.1Q, 802.11, 802.15, NVGRE, VXLAN, MPLS";

description
  " The layer 2 header match includes
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 13-header-actions {
    choice 13-header-act-type {
        case l3-ipv4-hdr {
            leaf set-ttl {
                type boolean;
                description "flag to set TTL";
            }
        }
    }
}

any reference to L2 technology";
}
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 next-header {
    type uint8;
    description "value to
    set in next IPv6 header";
  }

  leaf traffic-class {
    type uint8;
    description "value to set
in traffic class;)

leaf flow-label {
  type uint16;
  description "value to set in IPOv6 flow label";
}

leaf 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";

declaration "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 {
    uses tcp-header-match;
    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";
    }
    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 {
    uses udp-header-match;
    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";
}
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
sctp chunk data";
  }
}

description "sctp chunk
header match fields";
}

grouping sctp-header-match {
  uses sctp-chunk;
  leaf sctp-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 {
    uses sctp-header-match;
    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";
    }
    description "sctp header actions";
}

grouping L4-header-match {
    choice 14-header-match-type {
        case 14-tcp-header {
            uses tcp-header-match;
        }
        case 14-udp-header {
            uses udp-header-match;
        }
        case 14-sctp {
            uses sctp-header-match;
        }
    }
    description "L4 match header choices";
}

grouping L4-header-action {
    choice L4-header-match-type {
        case 14-tcp-header {
            uses tcp-header-action;
        }
        case 14-udp-header {

uses udp-header-action;
}
case l4-sctp {
    uses sctp-header-action;
}
description "L4 header
match action choices";
}
description "L4 header matches";
}


grouping service-header-match {
    choice service-header-match-type {
        case sf-chain-meta-match {
            description "uses
sfc:service-function-chain-grouping:
+ 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 service-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-sfcservice-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 "refernce count on rule. ";
  }
  description
    "rule operational status";
}

  grouping group_status {
    leaf name {
      type string;
      description "eca group name";
    }
    list rules_status {
      uses rule_status;
      description "eca rule list";
    }

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;
}


```yaml
description "ECA matches";
}

grouping eca-qos-actions {
  leaf cnt-actions {
    type uint32;
    description "count of ECA 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-name;
    description "name of next hop";
  }
  description "ECA set or change packet Actions";
}

grouping eca-fwd-actions {
  leaf interface-fwd {
    type i2rs-rib: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";
}

container bnp-eca-policy-set {
  container policy-groups {
    list rule-group {
      key "group-name";
    }
  }
}
```
leaf group-name {
    type string;
    description
    "name of group of rules";
}

list rule {
    key "rule-name";
    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 eca-matches;
    uses eca-qos-actions;
    uses eca-fwd-actions;
    description "ECA rules";
} // end of rule
} // end of group list

description "Policy groups.";

description "Policy sets.";

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.ietf-i2rs-architecture]

[I-D.ietf-i2rs-rib-info-model]

[I-D.ietf-netconf-restconf]

[I-D.ietf-netmod-acl-model]


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