I2RS working group
Internet-Draft
Intended status: Standards Track
Expires: April 8, 2017

S. Hares
Q. Wu
Huawei
R. White
Ericsson

October 5, 2016

Filter-Based Packet Forwarding ECA Policy
draft-ietf-i2rs-pkt-eca-data-model-02.txt

Abstract

This document describes the yang data model for packet forwarding policy that filters received packets and forwards (or drops) the packets. Prior to forwarding the packets out other interfaces, some of the fields in the packets may be modified. If one considers the packet reception an event, this packet policy is a minimalistic Event-Match Condition-Action policy. This policy controls forwarding of packets received by a routing device on one or more interfaces on which this policy is enabled. The policy is composed of an ordered list of policy rules. Each policy policy rule contains a set of match conditions that filters for packets plus a set of actions to modify the packet and forward packets. The match conditions can match tuples in multiple layers (L1-L4, application), interface received on, and and other conditions regarding the packet (size of packet, time of day). The modify packet actions allow for setting things within the packet plus decapsulation and encapsulation packet. The forwarding actions include forwarding via interfaces, tunnels, or nexthops and dropping the packet. The policy model can be used with the session ephemeral (BGP Flow Specifications), reboot ephemeral state (I2RS ephemeral), and non-ephemeral routing/forwarding state (e.g. configuration state).

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 April 8, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document describes the yang data model for packet forwarding policy that filters received packets and forwards (or drops) the packets. Prior to forwarding the packets out other interfaces, some of the fields in the packets may be modified. If one considers the reception of a packet as an event, this minimalistic Event-Match Condition-Action policy. If one considers the reception of packets

containing Layer 1 to Layer 4 + application data a single packet, then this minimalistic policy can be called a packet-only ECA policy. This document will use the term packet-only ECA policy for this model utilizing the term "packet" in this fashion.

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

- content of packet headers for layer 1 to layer 4,
- application protocol data and headers,
- interfaces the packet was received on,
- time packet was received, and
- size of packet.

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

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

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

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

The high-level yang and the actual yang are not aligned. This is an interim-release of this document.

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:

Policy set:

Policy set is a set of policies

Policy:

A policy is a is an ordered set of rules .

Rule

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

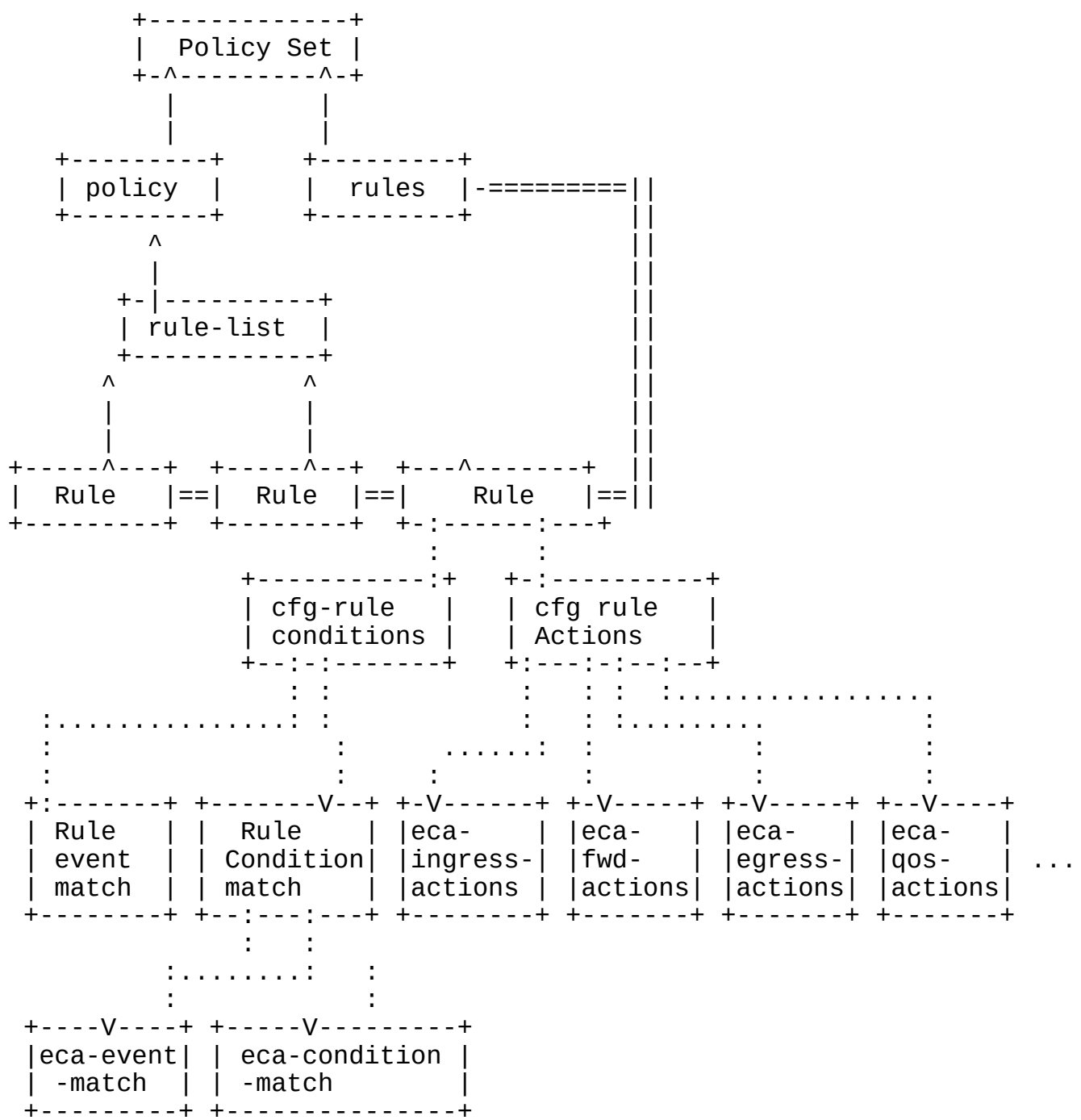

Figure 1: ECA rule structure

3. BNP Rule Groups

The pkt ECA policy is a policy whihc is an order set of pkt-ECA policy rules. The rules assume the event is the reception of a packet on the machine on a set of interfaces. This policy is

associated with a set of interfaces on a routing device (physical or virtual).

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

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

A policy group may have rulies at different order levels. For example, policy group 1 could have three policy rules at rule order 1 and four policy rules at rule order 5.

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

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

The generic match conditions are specific to a particular layer are refined by matches to a specific layer (as figure 2 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.

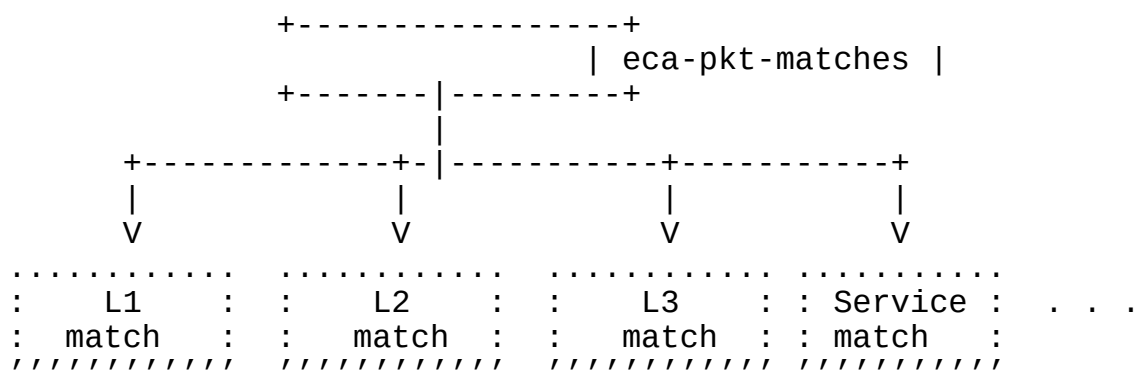

Figure 2 match logic

4. Packet ECA (event-condition-action) Filter Model in High Level Yang

The description of packet event-condition-action data model include:

module included in module

top level diagram

4.1. modules included

Below is the high level module inclusions.

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

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

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

Figure 3 - high level inclusion

4.2. top level description

Below is the high level yang diagram

```
module ietf-pkt-eca-policy
  +--rw pkt-eca-policy-cfg
  |  +--rw pkt-eca-policy-set
  |     +--rw policies* [policy-name]
  |     |  +--rw policy-name string
  |     |  +--rw vrf-name string
  |     |  +--rw address-family
  |     |  +--rw rule-list* [rule-name]
  |     |  |  +--rw rule-name
  |     |  |  +--rw rule-order-id uint16
  |     |  |  +--rw default-action-id integer
  |     |  |  +--rw default-resolution-strategy-id integer
  |     +--rw rules* [order-id rule-name]
  |        +--rw order-id uint16
  |        +--rw rule-name string
  |        +--rw policy-name string
  |        +--rw cfg-rule-conditions [rule-cnd-id]
  |        |  +--rw rule-cnd-id uint32
  |        |  +--rw support
  |        |  |  +--rw event-matches boolean
  |        |  |  +--rw pkt-matches boolean
  |        |  |  +--rw usr-context-matches boolean
  |        |  +--rw eca-events-match*  [rule-event-id]
  |        |  |  +--rw rule-event-it uint16
  |        |  |  | ... time-event match (see below)
  |        |  +--rw eca-condition-match
  |        |  |  +--rw eca-pkt-matches* [pkt-match-id]
  |        |  |  | ...(see packet matches below)
  |        |  |  | ... (address, packet header, packet payload)
  |        |  |  +--rw eca-user-context-matches* [usr-match-id]
  |        |  |  | ... (see user context match below)
  |        |  +--rw cfg-rule-actions [cfgr-action-id]
  |        |  |  +--rw cfgr-action-id
  |        |  |  +--rw eca-actions* [action-id]
  |        |  |  |  +--rw action-id uint32
  |        |  |  |  +--rw eca-ingress-actions*
  |        |  |  |  | ... (permit, deny, mirror)
  |        |  |  |  +--rw eca-fwd-actions*
  |        |  |  |  | ...  (invoke, tunnel encap, fwd)
  |        |  |  |  +--rw eca-egress-acttions*
  |        |  |  |  | .. .
  |        |  |  |  +--rw eca-qos-actions*
  |        |  |  |  | ...
```

```
|        |  |  +--rw eca-security-actions*
|        +--rw policy-conflict-resolution* [strategy-id]
|        |  +--rw strategy-id integer
|        |  +--rw filter-strategy identityref
|        |  |  .. FMR, ADTP, Longest-match
|        |  +--rw global-strategy identityref
|        |  +--rw mandatory-strategy identityref
|        |  +--rw local-strategy identityref
|        |  +--rw resolution-fcn uint32
|        |  +--rw resolution-value uint32
|        |  +--rw resolution-info  string
|        |  +--rw associated-ext-data*
|        |  |  +--rw ext-data-id integer
|        +--rw cfg-external-data* [cfg-ext-data-id]
|        |  +--rw cfg-ext-data-id integer
|        |  +--rw data-type integer
|        |  +--rw priority uint64
|        |  |  uses external-data-forms
|        |  ... (other external data)
|     +--rw pkt-eca-policy-opstate
|        +--rw pkt-eca-opstate
|           +--rw policies-opstat* [policy-name]
|           |  +--rw rules-installed;
|           |  +--rw rules_opstat* [rule-name]
|           |           |  +--rw strategy-used [strategy-id]
|           +--rw rules_opstate* [rule-order rule-name]
|           |  +--rw status
|           |  +--rw rule-inactive-reason
|           |  +--rw rule-install-reason
|           |  +--rw rule-installer
|           |  +--rw refcnt
|           +--rw rules_pktstats* [rule-order rule-name]
|           |  +--rw pkts-matched
|           |  +--rw pkts-modified
|           |  +--rw pkts-forward
|           |        +--rw op-external-data [op-ext-data-id]
|           |        |  +--rw op-ext-data-id integer
|           |        |  +--rw type identityref
|           |        |  +--rw installed-priority integer
|           |        |  |  (other details on external data )
```

figure 4 - high-level yang for policy set

The three levels of policy are expressed as:

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

Operational State for Policy
=========================================
Policy level: pkt-eca-policy-opstate
group level:  pkt-eca-opstate:policies-opstat*
rule level:   pkt-eca-opstate:rules_opstat*
              pkt-eca-opstate:rules-pktstat*
external id:  pkt-eca-opstate:op-external-data*
```

figure 5

4.3. Conditional filters

The condition filters in the packet eca policy module included the following:

- event filters - time as an augment to reception of a packet.
- conditional matches on packet content or user-related content

The sections below provide the high-level yang for these sections of hte model.

```
module:i2rs-pkt-eca-policy
.....
+--rw pkt-eca-policy-cfg
   |  +--rw pkt-eca-policy-set
   |  +--rw pkt-eca-policy-set
   |  | ...
   |  +--rw rules* [order-id rule-name]
   |     +--rw order-id uint16
   |     +--rw rule-name string
   |     +--rw policy-name string
   |     +--rw cfg-rule-conditions [rule-cnd-id]
   |     |  +--rw rule-cnd-id integer
   |     |  +--rw eca-events-match*  [rule-event-id]
   |     |  |  +--rw rule-event-it uint16
   |     |  |  | ... time-event match (see below)
   |     |  +--rw eca-condition-match
   |     |  |  +--rw eca-pkt-matches* [pkt-match-id]
   |     |  |  |  ... (see L1-L4 matches below)
   |     |  |  +--rw eca-usr-context-matches* [usr-match-id]
   |     |  |  | (user, schedule, region, target,
   |     |  |  |     state, direction)
```

Figure 6

4.3.1. Event Match Filters

The default event is the event of receiving a packet. In addition, the events allow a time-event match. Time events are provided as a list which includes specific times or ranges of time.

```
|  +--rw pkt-eca-policy-set
|  +--rw pkt-eca-policy-set
|  | ...
|  +--rw rules* [order-id rule-name]
|     +--rw order-id uint16
|     +--rw rule-name string
|     +--rw policy-name string
|     +--rw cfg-rule-conditions [rule-cnd-id]
|     |  +--rw rule-cnd-id  uint32
|     |  +--rw support
|     |  |  +--rw event-matches boolean
|     |  |  +--rw pkt-matches boolean
|     |  |  +--rw usr-context-matches boolean
|     |  +--rw eca-events-match*  [rule-event-id]
|     |  |  +--rw rule-event-it  uint16
|     |  |  +--rw time-type identityref
|     |  |  +--(one-time)
|     |  |  |   +--rw event-time yang:date-and-time
|     |  |  +--(range-time)
|     |  |  +--rw event-start-time yang:date-and-time
|     |  |  +--rw event-end-time yang:date-and-time
```

figure 7

4.3.2. ECA Packet Condition Matches

The ECA condition matches are packet matches (eca)

4.3.2.1. Packet-match filter list (eca-pkt-match*)

The packet match content filters include: address filters and packet header content filters, and packet payload filters.

```
module:i2rs-pkt-eca-policy
+--pkt-eca-policy-set
   +--rw rules* [order-id rule-name]
   |  |....
   |  +--rw cfg-rule-conditions [rule-cnd-id]
   |  |  +--rw rule-cnd-id uint32
   |  |  +--rw support
   |  |  |  +--rw event-matches boolean
   |  |  |  +--rw pkt-matches boolean
   |  |  |  +--rw usr-context-matches boolean
   |  |  +--rw eca-event-match
   |  |  |  ...
   |  |  +--rw eca-condition-match
   |  |  +--rw eca-pkt-matches* [pkt-match-id]
   |  |  |   +--rw packet-match-id uint16
   |  |  |   +--rw packet-match-type identityref
   |  |  |   +--(packet-match-type)?
   |  |  |   |   +--:(address-pkt-match)
   |  |  |   |   | ...
   |  |  |   |   +--:(layer-pkt-match)
   |  |  |   |   | ...
   |  |  |   |   +--:(payload-pkt-match)
   |  |  |   |   | ...
   |  |  +--rw eca-user-matches [user-match-id]
```

Figure 8

4.3.2.1.1. Match filters for addresses in packet

The address matches match the L3, mpls, MAC and interface address scope. Figure x shows this match

```
 +--rw eca-pkt-matches* [pkt-match-id]
|  +--rw packet-match-id uint16
|  +--rw eca-pkt-match-type identityref
|  +--address-scope?
|  |  +--:(route-type)
|  |  |  +--: (ipv4)
|  |  |  |   ... src, dest, src-dest
|  |  |  +--: (ipv6)
|  |  |  |   ... src, dest, src-dest
|  |  |  +--: (mpls)
|  |  |  |    ... 32 bit label
|  |  |  +--: (mac)
|  |  |  |    ... src, dest, src-dest
|  |  |  +--:  (interface-route)
|  |  |  |    .... interface
```

Figure 9

4.3.2.1.2. Packet header matches

The packet header matches match interface, L1-L4 headers, service chain headers, and packet size. The L1 header expected to be a null match except if there is an advanced L1 technology such as l1 with a L1 identifier that can be detected in the packet. Figure x shows these matches.

```
|  +--rw (layer-type)
|  |  +--:(interface-match-type)
|  |  |  |...
|  |  +--:(L1-header-match)
|  |  |  |...
|  |  +--:(L2-header-match)
|  |  |  +--(802.1Q)
|  |  |  |....
|  |  |  +--(802.11)
|  |  |  | ...
|  |  |  +--(802.15)
|  |  |  | ...
|  |  |  +--(NVGRE)
|  |  |  |  ...
|  |  |  +--(VXLAN)
|  |  |  | ...
|  |  |  +--(MPLS )
|  |  |  | ..
|  |  +--:(L3-header-match)
|  |  |  +--(l3-ipv4-header)
|  |  |  |   ...
|  |  |  +--(l3-ipv6-header)
|  |  |  |      ...
|  |  |  +--(l3-gre-header)
|  |  |  |      ...
|  |  +--: L4-header-match
|  |  |  +--(l4-tcp-header)
|  |  |  |  ...
|  |  |  +--(l4-udp-header)
|  |  |  | ...
|  |  |  +--(l4-sctp-header)
|  |  |  |       ...
|  |  |  +--: Service-header-match
|  |  |     +--(sf-chain-meta-match)
|  |  |       ...
|  |  |     +--(sf-path-meta-match)
|  |  |       ..
|  |  +--:(packet-size)
|  |     +--l1-size-match uint32
|  |     +--l2-size-match uint32
|  |     +--l3-size-match uint32
|  |     +--l4-size-mtach uint32
|  |     +--service-meta-size uint32
|  |     +--leaf service-meta-payload  uint32
|  +---rw packet
```

Figure 10

4.3.2.1.3. Payload matches

The payload information is a stream of bytes to be found in the packet payload beyond the L4 or service-path header. The structure of this data is simply a list of byte strings as figure x shows.

```
|  |  |....
|  |  +--rw eca-pkt-matches* [pkt-match-id]
|  |  |   +--rw packet-match-id uint16
|  |  |   +--rw packet-match-type identityref
|  |  |   +--(packet-match-type)?
|  |  |   |   +--:(address-pkt-match)
|  |  |   |   | ...
|  |  |   |   +--:(layer-pkt-match)
|  |  |   |   | ...
|  |  |   |   +--:(payload-pkt-match)
|  |  |   |   |  +--rw packet-payload *[packet-payload-id]
|  |  |   |   |  +--rw packet-payload-id   uint16
|  |  |   |   |  +--rw payload-match-bytes uint16
|  |  |   |   |  +--rw packet-payload  string
```

Figure 11

4.3.2.2. Matches on User Context

The match on user context allows filtering for a packet plus a filter related to a user.

Since not all I2RS routers are access routers, the support for matches has a flag for user filter. It is expected that core routers may not support contextual matching.

One example of user filters is for is parental controls. During school hours, the teenager Joe is restrict from certain web sites from September 1 while Joe is at at school. This "school" filter is an example of a period filter which has a start time (8:00am) and end time (3:30pm), which is valid beginning September 1, 2016. This filter applies only to the region of school networks. The filter looks for specific entertainment (e.g. YouTube) web sites, social-media (e.g. facebook), and gaming sites. This block is for their mobile phone and tablet, but not the computer that says at home.

The following is the components

- user identifier (name, tenant id, virtual network),
- schedule for the user filter (once or periodic, time range (start/end), and weekly validity check),

o region this filter is valid.

o targeted services, applications, devices, and state.

```
module:i2rs-pkt-eca-policy
.....
+--rw pkt-eca-policy-cfg
   |  +--rw pkt-eca-policy-set
   |  +--rw pkt-eca-policy-set
   |  | ...
   |  +--rw rules* [order-id rule-name]
   |     +--rw order-id uint16
   |     +--rw rule-name string
   |     +--rw policy-name string
   |     +--rw cfg-rule-conditions [rule-cnd-id]
   |     |  +--rw rule-cnd-id uint32
   |     |  | ...
   |     |  +--rw eca-event-match* [rule-event-id]
   |     |  |  ..
   |     |  +--rw eca-pkt-matches* [pkt-match-id]
   |     |  |  ....
   |     |  +--rw eca-usr-context-matches* [usr-match-id]
   |     |  |  +--rw user* [user-id]
   |     |  |  |  +--rw user-id uint32
   |     |  |  |  +--rw user-name string
   |     |  |  |  +--rw user-type identityref
   |     |  |  |  |  +--(user-type)?
   |     |  |  |  |  |   +--:(tenant)
   |     |  |  |  |  |   |  +--rw tenant-id uint16
   |     |  |  |  |  |   +--:(vn-id)
   |     |  |  |  |  |   |  +--rw vn-id uint16
   |     |  |  +--rw schedule* [schedule-name]
   |     |  |  | .....
   |     |  |  +--rw target
   |     |  |  |  +--rw protocol
   |     |  |  |  | ... (UDP, TCP, ICMP, ICMPv6, IP, IPv6)
   |     |  |  |  +--rw transport-ports
   |     |  |  |  |  +--rw src-port inet:port-number
   |     |  |  |  |  +--rw dest-port intent:port-number
   |     |  |  |  +--service [service-name]
   |     |  |  |  |...
   |     |  |  |  +--rw application
   |     |  |  |  |...
   |     |  |  |  +--rw device
   |     |  |  |  | ..
```

Figure 12

Schedule filters allow a time for the filter. Continuing our parental control filters for school, the schedule an be a list of weekly filters for Monday-Friday of the school week. The first filter (School-Monday) would have a of start time of 8:00am GMT September 5, 2016 an end time of 4:00pm GMT September 5, 2016. The schedule type woudl be weekly. The validity-until time would be December 20, 2016. The region impacted by this schedule would be AS20999 which is the service provider of the school’s network.

```
+--rw pkt-eca-policy-cfg
   |  +--rw pkt-eca-policy-set
   |  +--rw pkt-eca-policy-set
   |  | ...
   |  +--rw rules* [order-id rule-name]
   |     +--rw order-id uint16
   |     +--rw rule-name string
   |     +--rw policy-name string
   |     +--rw cfg-rule-conditions [rule-cnd-id]
   |     |  +--rw rule-cnd-id uint32
   |     |  +--rw eca-usr-context-matches* [usr-match-id]
   |     |  |  +--rw schedule* [schedule-name]
   |     |  |  |  +--rw schedule-name
   |     |  |  |  +--rw schedule-type identityref /* one-time, weekly, 2 wee
ks, monthly */
   |     |  |  |  +--rw start-type? yang:date-and-time
   |     |  |  |  +--rw end-type?  yang:date-and-time
   |     |  |  |  +--rw validity-until yang:date-and-time /* valid until */
   |     |  |  +--rw region *[as-4byte]
   |     |  |  |  +--rw as-4byte uint32 /* region */
```

figure 13

The target for this service filtering is specified by protocols, applications, and devices. The figure below shows the filtering for a target protocol and port number or an application.

```
+--rw pkt-eca-policy-cfg
   |  +--rw pkt-eca-policy-set
   |  +--rw pkt-eca-policy-set
   |  | ...
   |  +--rw rules* [order-id rule-name]
   |     +--rw order-id uint16
   |     +--rw rule-name string
   |     +--rw policy-name string
   |     +--rw cfg-rule-conditions [rule-cnd-id]
   |     |  +--rw rule-cnd-id uint32
   |     |  +--rw eca-usr-context-matches* [usr-match-id]
   |     |  |  +--rw target
   |     |  |  |  +--rw service* [svc-id svc-name]
   |     |  |  |  |  +--rw svc-id uint16
   |     |  |  |  |  +--rw svc-name string
   |     |  |  |  |  +--rw protocol-support
   |     |  |  |  |  |  +--rw TCP boolean
   |     |  |  |  |  |  +--rw UDP boolean
   |     |  |  |  |  |  +--rw ICMP boolean
   |     |  |  |  |  |  +--rw ICMPv6 boolean
   |     |  |  |  |  |  +--rw IP boolean
   |     |  |  |  |  +--rw src-port? inet:port-number
   |     |  |  |  |  +--rw dest-port? inet_port-number
   |     |  |  |  +--rw application* [app-name]
   |     |  |  |  |  +--rw app-name string
   |     |  |  |  |  +--rw app-id uint16
   |     |  |  |  |  +--rw app-category
   |     |  |  |  |       /* business, educational, internet */
   |     |  |  |  |  +--rw app-subcategory
   |     |  |  |  |       /* finance, email, game, social-net, web */
   |     |  |  |  |  +--rw app-data-transmission
   |     |  |  |  |     /* client-server, web-brower, p2p, network */
   |     |  |  |  |  +--rw app-risk-level
   |     |  |  |  |    /* exploitable, evasive, data-lost, malware-vehicle,
tun
   |     |  |  |  +--rw device
   |     |  |  |  |  +--rw pc boolean
   |     |  |  |  |  +--rw mobile-phone boolean
   |     |  |  |  |  +--rw tablet boolean
   |     |  |  |  |  +--rw voip-phone boolean
```

Figure 14

4.4. ECA Packet Actions

The packet actions list includes ingress actions,egress actions, Qos actions that modify the packet, and security actions. The High level Yang that shows where the action fit is in figure 15, and the details

are shown in figure 16. The QoS actions per header is shown in figure 17.

```
module ietf-pkt-eca-policy
  +--rw pkt-eca-policy-cfg
  |  +--rw pkt-eca-policy-set
  |     +--rw policies* [policy-name]
  |     |  +--rw policy-name string
  |     |  +--rw vrf-name string
  |     |  +--rw address-family
  |     |  +--rw rule-list* [rule-name]
  |     |  |  +--rw rule-name
  |     |  |  +--rw rule-order-id uint16
  |     |  |  +--rw default-action-id integer
  |     |  |  +--rw default-resolution-strategy-id integer
  |     +--rw rules* [order-id rule-name]
  |        +--rw order-id uint16
  |        +--rw rule-name string
  |        +--rw cfg-rule-conditions [rule-cnd-id]
  |        | ...
  |        +--rw cfg-rule-actions* [cfgr-action-id]
  |        |  +--rw cfgr-action-id
  |        |  +--rw eca-actions* [action-id]
  |        |  |  +--rw action-id uint32
  |        |  |  +--rw eca-ingress-actions
  |        |  |  | ... (permit, deny, mirror)
  |        |  |  +--rw eca-fwd-actions*
  |        |  |  | ...  (invoke, tunnel encap, fwd)
  |        |  |  +--rw eca-egress-actions*
  |        |  |  |  ()
  |        |  |  +--rw eca-qos-actions*
  |        |  |  | ...
  |        |  |  +--rw eca-security-actions*
```

Figure 15

This figure shows the details for each action section (ingress, egress, qos, and security).

```
|        +--rw eca-ingress-actions
|        |  +--rw num-fwd-actions
|        |  +--rw fwd-actions
|        |  |  +--rw permit boolean
|        |  |  +--rw mirror boolean
|        |  |  +--rw interface-fwd ip:interface-ref
|        |  |  +--uses i2rs:rib-nexthop
|        |  |  +--uses ip-next-fwd;
|        +--rw eca-egress-actions
|        |  +--rw packet-rate uint32
|        |  +--rw byte-rate uint32
|        |  +--rw tunnel-encap boolean
|        |  +--rw exit-fwding boolean
|        |  +--rw interface-egress ip:interface-ref
|        |  +--uses i2rs:rib-nexthop
|        |  +--uses ip-next-fwd;
|        +--rw eca-qos-actions
|        |  ... (see figure x below )
|        +--rw eca-security
|
|        |  +--rw security-action-type identityref
|        |  +--(security-action-type)?
|        |  +--:(content-security-action) ANYXML
|        |  | ...
|        |  +--:(attack-mitigation-type) ANYXML
|        |  | ..
|        |  +--:(single-packet-type) ANYXML
```

figure 16 - forwarding

> The QOS actions modify the headers are shown below.

```
|        +--rw ecq-qos-actions
|        |  +--rw cnt-actions uint8 /* modifying actions */
|        |  +--rw mod-actions
|        |  |  +--case interface-actions
|        |  |  | ..
|        |  |  +--case L1-action
|        |  |  | ..
|        |  |  +--case L2-action
|        |  |  | ..
|        |  |  +--case L3-action
|        |  |  | ..
|        |  |  +--case L4-action
|        |  |  | ..
|        |  |  +--case service-action
|        |  |  | ..
```

Figure 17

4.5. Policy Conflict Resolution strategies

Some policies within the filter-base policy will conflict. For example, a global strategy may conflict with a local node strategy. This portion of the filter-based data model provides this support.

```
|        +--rw pc-resolution-strategies* [strategy-id]
|        |  +--rw strategy-id integer
|        |  +--rw pc-resolution-supported boolean
|        |  +--rw filter-strategy identityref
|        |  |  .. FMR, ADTP, Longest-match
|        |  +--rw global-strategy identityref
|        |  +--rw mandatory-strategy identityref
|        |  +--rw local-strategy identityref
|        |  +--rw resolution-fcn uint32
|        |  +--rw resolution-value uint32
|        |  +--rw resolution-info  string
|        |  +--rw associated-ext-data*
|        |  |  +--rw ext-data-id integer
```

Figure 18

4.6. External Data

External data may be used to set the policy.

```
     |       +--rw cfg-external-data* [cfg-ext-data-id]
     |        |  +--rw cfg-ext-data-id integer
     |        |  +--rw data-type integer
     |        |  +--rw priority uint64
     |        |  +--rw external-data-forms anyxml /* mount point */
```

Figure 19

5. i2rs-eca-policy Yang module

```
<CODE BEGINS> file "ietf-pkt-eca-policy@2016-02-09.yang"
 module ietf-pkt-eca-policy {
    namespace "urn:ietf:params:xml:ns:yang:ietf-pkt-eca-policy";
   // replace with iana namespace when assigned
    prefix "pkt-eca-policy";

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

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

  // meta
    organization "IETF I2RS WG";

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

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

        revision "2016-06-26" {
           description "sec ond revision";
           reference "draft-ietf-i2rs-pkt-eca-policy-dm-03";
```

```
        }


  // interfaces - no identity  matches


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

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

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

        // L4 header match identities

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

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

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

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

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


         // 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.";
    }

   identity rule-cr-type {
```

```
     description "status
         values for rule: FMR (0), ADTP (1),
         Longest-match (2)";
    }

        identity rule-cr-FMR {
           base rule-cr-type;
           description "first match resolution.";
    }

        identity rule-cr-ADTP {
       base rule-cr-type;
           description "ADTP resolution.";
    }


        identity rule-cr-longest {
       base rule-cr-type;
           description "longest match resolution.";
    }


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

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

```
                description
               "action to enable interface";
               }
             leaf interface-disable {
         type boolean;
              description
               "action to disable interface";
             }
        }


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

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

         grouping L2-802-1Q-header {
```

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

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

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

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

```
                leaf L2-ethertype {
                      type uint16;
                      description "Ether type
                      from the header";
                }
          }


        grouping L2-VXLAN-header {
           container vxlan-header {
                 uses i2rs-rib:ipv4-header;
                 leaf vxlan-network-id {
                     type uint32;
                     description "VLAN network id";
                    }
                 description " choices for
                   L2-VLAN header matches.
                       Outer-header only.
                       Need to fix inner header. ";
          }
           description
               "This VXLAN header may
               be replaced by actual VXLAN yang
               module reference";
        }

        grouping L2-NVGRE-header {

                container nvgre-header {
                   uses L2-802-1Q-header;
                   uses i2rs-rib:ipv4-header;
                  leaf gre-version {
                   type uint8;
                   description "L2-NVGRE GRE version";
                   }
                  leaf gre-proto {
                   type uint16;
                   description "L2-NVGRE protocol value";
                       }
                  leaf virtual-subnet-id {
             type uint32;
                      description "L2-NVGRE subnet id value";
             }
          leaf flow-id {
             type uint16;
                      description "L2-NVGRE Flow id value";
                }
                  // uses L2-802-1Q-header;
```

```
            description
                  "This NVGRE header may
                  be replaced by actual NVGRE yang
                   module reference";
            }
                description "Grouping for
                   L2 NVGRE header.";
         }


         grouping L2-header-match {

            choice l2-header-match-type {
                     case l2-802-1Q {
                           uses L2-802-1Q-header;
                        }
                        case l2-802-11 {
                          // matches for 802.11 headers
                        }
                        case l2-802-15 {
                       // matches for 802.1 Ethernet
                      }
                        case l2-NVGRE {
                         // matches for NVGRE
                         uses L2-NVGRE-header;
                        }
                        case l2-VXLAN-header {
                         uses L2-VXLAN-header;
                        }
                        case l2-mpls-header {
                          uses i2rs-rib:mpls-header;
                        }
                 description "Choice of L2
                    headers for L2 match";
                }
        description
               " The layer 2 header match includes
                  any reference to L2 technology";
         }

         grouping L2-NVGRE-mod-acts {
          // actions for NVGRE
           leaf set-vsid {
        type boolean;
              description
                  "Boolean flag to set VSID in packet";
                }
           leaf set-flowid {
```

```
            type boolean;
            description
                 "Boolean flag to set VSID in packet";
              }
          leaf vsi {
           type uint32;
               description
                 "VSID value to set in packet";
               }
          leaf flow-id {
                type uint16;
            description
                "flow-id value to set in packet";
          }
          description "L2-NVRE Actions";
        }

        grouping L2-VXLAN-mod-acts {
          leaf set-network-id {
             type boolean;
             description
                "flag to set network id in packet";
          }
          leaf network-id {
            type uint32;
                description
                "network id value to set in packet";
          }
          description "VXLAN header
            modification actions.";
        }

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

```
            }
            description "MPLS modify
             header actions";
         }

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

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

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


        grouping L3-header-match {

            choice L3-header-match-type {
                   case l3-ipv4-hdr {
                     uses i2rs-rib:ipv4-header;
                   }
                   case l3-ipv6-hdr {
                     uses i2rs-rib:ipv6-header;
                   }
                   case L3-gre-tunnel {
                     uses i2rs-rib:gre-header;
                   }
                     description "match for L3
                         headers for IPv4, IPv6,
                          and GRE tunnels";
                }
        description "match for L3 headers";
         }
```

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

        grouping L3-header-actions {
          choice l3-header-act-type {
                case l3-ipv4-hdr {
                    leaf set-ttl {
                          type boolean;
                          description "flag to set TTL";
                    }
                        leaf set-dscp {
                          type boolean;
                          description "flag to set DSCP";
                        }
                    leaf ttl-value {
                          type uint8;
                          description "TTL value to set";
                        }
                        leaf dscp-val {
              type uint8;
                          description "dscp value to set";
                          }
            }
                case l3-ipv6-hdr {
                    leaf set-next-header {
                          type boolean;
                          description
                           "flag to set next routing
                           header in IPv6 header";
                        }
                        leaf set-traffic-class {
                          type boolean;
                          description
                           "flag to set traffic class
                           in IPv6 header";
```

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

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

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

```
          description "actions to
           be performed on L3 header";
        }


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

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

           leaf tcp-s-port {
             type uint16;
             description "source port match value";
                  }
           leaf tcp-d-port {
             type uint16;
         description "dest port value
                  to match";
```

```
            }
            leaf seq-num {
                  type uint32;
                  description "sequence number
                   value to match";
                 }
                 leaf ack-num {
                 type uint32;
                 description "action value to
                  match";
                 }
           description "Actions to
              modify TCP header";
        }

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

        }

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

```
                type uint16;
              description "UDP Destination
                    port match value";
                }

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

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

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

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

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

```
            description "sctp header match
                  verification tag value";
               }
               description "SCTP header
                    match values";
        }

        grouping sctp-header-action {
            leaf set-stcp-src-port {
                type boolean;
             description "set source port in sctp header";
               }
               leaf set-stcp-dst-port {
                type boolean;
             description "set destination port in sctp header";
               }
               leaf set-stcp-chunk1 {
                type boolean;
             description "set chunk value in sctp header";
               }
               leaf chunk-type-value  {
                 type uint8;
                 description "sctp chunk type value";
               }
               leaf chunk-flag-value {
                 type uint8;
                 description "sctp chunk type
                  flag value";
            }

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

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


        grouping L4-header-match {
             choice l4-header-match-type {
                   case l4-tcp-header {
                     uses tcp-header-match;
```

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


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

   grouping service-header-match {
              choice service-header-match-type {
                case sf-chain-meta-match {
                 description "uses
                 sfc-sfc:service-function-chain-grouping:
                 + sfc-sfc:service-function-chain";
                }
                 case sf-path-meta-match {
                  description "uses
                      sfc-spf:service-function-paths:
                      + sfc-spf:service-function-path";
                 }
              description "SFC header match
               choices";
              }
          description "SFC header and path
               matches";
      }

      grouping sfc-header-actions {
              choice service-header-match-type {
                case sf-chain-meta-match {
                  leaf set-chain {
                        type boolean;
                     description "flag to set
                         chain in sfc.  Should
```

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


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

```
           "rule operational status";
         }

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

// links between rule to group

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

```
           }
           description "link between
             group and link";
          }
          description "rule-name to
           group link";
        }

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

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

```
                    to be modified";
                  }
                       leaf pkts-dropped {
                    type uint64;
                    description "number of
                     packets that filter caused
                     to be modified";
                  }
               leaf bytes-dropped {
                    type uint64;
                    description "number of
                     packets that filter caused
                     to be modified";
                  }
                  leaf pkts-forwarded {
                    type uint64;
                    description "number of
                     packets that filter caused
                     to be forwarded.";
                    }
                    leaf bytes-forwarded {
                    type uint64;
                    description "number of
                     packets that filter caused
                     to be forwarded.";
                    }

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



   grouping packet-size-match {
         leaf l1-size-match {
            type uint32;
            description "L1 packet match size.";
                 }
         leaf l2-size-match {
            type uint32;
                description "L2 packet match size.";
                }
```

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

        grouping time-day-match {

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

        description "matches for
          time of day.";

        }


        grouping user-event-match {
       leaf user-name {
             type string;
```

```
            description "name of user
                 event";
           }
           leaf match-string {
            type string;
                description "user match
                string";
           }

        description "matches for
          time of day.";

        }


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

        }

    grouping eca-pkt-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;
          description "ECA matches";
        }

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

```
          }

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

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

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

        grouping eca-qos-actions {
          leaf cnt-actions {
            type uint32;
            description "count of ECA actions";
           }
          list qos-actions {
             key "action-id";
             leaf action-id {
             type uint32;
             description "action id";
             }
             uses interface-actions;
             uses L1-header-actions;
             uses l2-header-mod-actions;
             uses L3-header-actions;
             uses L4-header-actions;

            description "ECA set or change
             packet Actions. Actions may be
                 added here for interface,
                 L1, L2, L3, L4 nad service forwarding
                 headers.";
           }
          description "eca- qos actions";
         }

         grouping ip-next-fwd {
                leaf rib-name {
                  type string;
```

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

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

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

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

```
  }

  grouping eca-egress-actions {
    leaf packet-rate {
          type uint32;
          description "maximum packet-rate";
    }
   leaf byte-rate {
          type uint64;
          description "maximum byte-rate ";
    }
    description "packet security actions";
  }


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

           }
           leaf global-strategy {
             type boolean;
             description "global strategy";
          }
          leaf mandatory-strategy {
           type boolean;
           description "required strategy";
          }
          leaf local-strategy {
           type boolean;
           description "local strategy";
          }
          leaf resolution-fcn {
           type uint64;
           description "resolution function id ";
          }
          leaf resolution-value {
```

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


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

```
        description "external data list";
    }


     grouping pkt-eca-policy-set {
         list groups {
            key "group-name";
                leaf group-name {
                  type string;
                  description
                   "name of group of rules";
                 }
                leaf vrf-name {
                  type string;
                  description "VRF name";
                }
                uses rt:address-family;
                list group-rule-list {
                  key "rule-name";
                  leaf rule-name {
                   type string;
                   description "name of rule";
                   }
                  leaf rule-order-id {
                    type uint16;
                        description "rule-order-id";
                  }
                   description "rules per group";
                }
                description "pkt eca rule groups";
          }
          list eca-rules {
                key "order-id";
                ordered-by user;
                leaf order-id {
                   type uint16;
                   description "Number of order
                    in ordered list (ascending)";
                }
                leaf eca-rule-name {
                    type string;
                        description "name of rule";
                }
                 leaf installer {
                   type string;
                   description
                         "Id of I2RS client
                          that installs this rule.";
```

```
                }
                 uses eca-event-matches;
                 uses eca-ingress-actions;
                 uses eca-qos-actions;
                 uses eca-security-actions;
                 uses eca-fwd-actions;
                 uses eca-egress-actions;
                 uses cfg-external-data;
                 uses policy-conflict-resolution;

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

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


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

 }


 <CODE ENDS>
```

6. IANA Considerations

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

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

associated prefix "pkt-eca";

7. Security Considerations

These generic filters are 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-rib-info-model]
Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-09 (work in progress), July 2016.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-17 (work in progress), September 2016.

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-08 (work in progress), July 2016.

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

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

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

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

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

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