A YANG Data model for ECA Policy Management
draft-wwx-netmod-event-yang-04

Abstract

RFC8328 defines a policy-based management framework that allow
definition of a data model to be used to represent high-level,
possibly network-wide policies. Policy discussed in RFC8328 are
classified into imperative policy and declarative policy, ECA policy
is an typical example of imperative policy. This document defines an
YANG data model for the ECA policy management. The ECA policy YANG
provides the ability for the network management function (within a
controller, an orchestrator, or a network element) to control the
configuration and monitor state change on the network element and
take simple and instant action when a trigger condition on the system
state is met.

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 https://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
1. Introduction

Network management consists of using one or multiple device-, technology-, service specific policies to influence management behavior within the system and make sure policies are enforced or executed correctly.
[RFC8328] defines a policy-based management framework that allow
definition of a data model to be used to represent high-level,
possibly network-wide policies. Policies discussed in [RFC8328] are
classified into imperative policy and declarative policy.
Declarative policy specifies the goals to be achieved but not how to
achieve those goals while imperative policy specifies when Events are
triggered and what actions must be performed on the occurrence of an
event. ECA policy is a typical example of imperative policy.

Event-driven management of states of managed objects across a wide
range of devices can be used to monitor state changes of managed
objects or resource and automatic trigger of rules in response to
events so as to better service assurance for customers and to provide
rapid autonomic response that can exhibit self-management properties
including self-configuration, self-healing, self-optimization, and
self-protection. Following are some of the use-cases where such ECA
Policy can be used:

- To filter out of objects underneath a requested a subtree, the
  subscriber may use YANG Push smart filter to request the network
  server to monitor specific network management data objects and
  send updates only when the value falls within a certain range.

- To filter out of objects underneath a requested a subtree, the
  subscriber may use YANG Push smart filter to request the network
  server to monitor specific network management data objects and
  send updates only when the value exceeds a certain threshold for
  the first time but not again until the threshold is cleared.

- To provide rapid autonomic response that can exhibit self-
  management properties, the management system delegate event
  response behaviors (e.g., auto-recover from network failure) to
  the network device so that the network can react to network change
  as quickly as the event is detected. The event response behaviors
  delegation can be done using ECA policy, e.g., to preconfigure
  protection/ restoration capability on the network device.

- To perform troubleshoot failures (i.e., fault verification and
  localization) and provide root cause analysis, the management
  system monitoring specific network management data objects may
  request the network device to export state information of a set of
  managed data objects when the value of monitored data object
  exceeds a certain threshold.

This document defines a ECA Policy management YANG data model. The
ECA Policy YANG provides the ability for the network management
function (within a controller, an orchestrator, or a network element)
to control the configurations and monitor state parameters on the
network element and take simple and instant action when a trigger condition on the system state is met.

The data model in this document is designed to be compliant with the Network Management Datastore Architecture (NMDA) [RFC8342].

2. Conventions used in this document

2.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

This document uses the following terms:

Error     A deviation of a system from normal operation [RFC3877].
Fault     Lasting error or warning condition [RFC3877].
Event     Something that happens which may be of interest or trigger the invocation of the rule. A fault, an alarm, a change in network state, network security threat, hardware malfunction, buffer utilization crossing a threshold, network connection setup, an external input to the system, for example [RFC3877].

2.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

3. Objectives

This section describes some of the design objectives for the ECA Policy management Data Model:

- Clear and precise identification of Event types in the ECA Policy.
- Clear and precise identification of managed object in the ECA Policy.
- Allow nested ECA policy, e.g., one event to be able to call another nested event.
Allow the NETCONF server send updates only when the value falls within a certain range.

Allow the NETCONF server send updates only when the value exceeds a certain threshold for the first time but not again until the threshold is cleared.

Provide rapid autonomic response in the network device that can exhibit self-management properties including self-configuration, self-healing, self-optimization, and self-protection.

4. Relationship to YANG Push

YANG-push mechanism provides a subscription service for updates from a datastore. And it supports two types of subscriptions which are distinguished by how updates are triggered: periodic and on-change.

The On-change Push allow receivers to receive updates whenever changes to target managed objects occur. This document specifies a mechanism that provides three trigger conditions:

- **Existence**: When a specific managed object appears, disappear or object change, the trigger fires, e.g. reserved ports are configured.

- **Boolean**: The user can set the type of boolean operator (e.g. unequal, equal, less, less-or-equal, greater, greater-or-equal, etc) and preconfigured threshold value (e.g. Pre-configured threshold). If the value of a managed object meet Boolean conditions, the trigger fires, e.g., when the boolean operator type is ‘less’, the trigger will be fired if the value of managed object is less than the pre-configured Boolean value.

- **Threshold**: The user can set the rising threshold, the falling threshold, the delta rising threshold, the delta falling threshold. A threshold test regularly compares the value of the monitored object with the threshold values, e.g., an event is triggered if the value of the monitored object is greater than or equal to the rising threshold or an event is triggered if the difference between the current measurement value and the previous measurement value is smaller than or equal to the delta falling threshold.

In these three trigger conditions, existence with type set to object change is similar to on Push change.

In addition, the model defined in this document provides a method for closed loop network management automation which allows automatic
trigger of rules in response to events so as to better service assurance for customers and to provide rapid autonomic response that can exhibit self-management properties including self-configuration, self-healing, self-optimization, and self-protection. The details of the usage example is described in Appendix A.

5. Model Overview

The YANG data model for the ECA Policy management has been split into two modules:

- The ietf-event-trigger.yang module defines a set of groupings for a generic trigger. It is intended that these groupings will be used by the policy based event management model or other models that require the trigger conditions. In this model, three trigger conditions are defined under the "test" choice node:

  * Existence: An existence test monitors and manages the absence, presence, and change of a data object, for example, interface status. When a monitored object is specified, the system reads the value of the monitored object regularly.

    + If the test type is Absent, the system triggers a network event and takes the specified action when the monitored object disappears.

    + If the test type is Present, the system triggers a network event and takes the specified action when the monitored object appears.

    + If the test type is Changed, the system triggers a network event and takes the specified action when the value of the monitored object changes.

  * Boolean: A Boolean test compares the value of the monitored object with the reference value and takes action according to the comparison result. The comparison types include unequal, equal, less, lessorequal, greater, and greaterorequal. For example, if the comparison type is equal, an event is triggered when the value of the monitored object equals the reference value. The event will not be triggered again until the value becomes unequal and comes back to equal.

  * Threshold: A Threshold trigger condition regularly compares compares the value of the monitored object with the threshold values.
+ A rising network event is triggered if the value of the monitored object is greater than or equal to the rising threshold.

+ A falling network event is triggered if the value of the monitored object is smaller than or equal to the falling threshold.

+ A rising network event is triggered if the difference between the current measurement value and the previous measurement value is greater than or equal to the delta rising threshold.

+ A falling network event is triggered if the difference between the current measurement value and the previous measurement value is smaller than or equal to the delta falling threshold.

+ A falling network event is triggered if the values of the monitored object, the rising threshold, and the falling threshold are the same.

+ A falling network event is triggered if the delta rising threshold, the delta falling threshold, and the difference between the current sampled value and the previous sampled value is the same.

If the value of the monitored object crosses a threshold multiple times in succession, the managed device triggers a network event only for the first crossing.

Editor-note: Three trigger conditions defined this document align with Complex condition and Simple condition defined in RFC3460? Are they sufficient to model ECA condition?

- The ietf-event.yang module defines four lists: trigger, target, event, and action. Triggers define the targets meeting some conditions that lead to events. Events trigger corresponding actions:

  * Each trigger can be seen as a logical test that, if satisfied or evaluated to be true, cause the action to be carried out. The ietf-event.yang module uses groupings defined in ietf-event-trigger.yang to present the trigger attributes.

  * The target list defines managed objects that can be added to logging or be set to a new value on the trigger, the trigger test type, or the event that resulted in the actions. The
target is also referred to as policy variable defined in [RFC3460].

* The event list defines what happens when an event is triggered, i.e., trigger the corresponding action, e.g., adding a logging (i.e., recording the triggered event), setting a value to the managed object or both. The group-id can be used to group a set of event that can be executed together, e.g., deliver a service or provide service assurance.

* Nested-event are supported by allowing one event’s trigger to reference other event’s definitions using the call-event configuration. Called events apply their triggers and actions before returning to the calling event’s trigger and resuming evaluation. If the called event is triggered, then it returns an effective boolean true value to the calling event. For the calling event, this is equivalent to a condition statement evaluating to a true value and evaluation of the event continues.

* The action list consists of updates or invocations on local managed object attributes and defines a set of actions which will be performed (e.g., logging, set value, etc) when the corresponding event is triggered. The value to be set can use many variations on rule structure.

The following tree diagrams [RFC8340] provide an overview of the data model for "ietf-event-trigger" module and the "ietf-event" module.

```
module: ietf-event-trigger
  grouping existences-trigger
    +- existences
    |  +- test-type?  enumeration
    |  +- target*    target
  grouping boolean-trigger
    +- boolean
    |  +- operator?  operator
    |  |  +- value?    match-value
    |  |  +- target*   target
  grouping threshold-trigger
    +- threshold
    |  +- rising-value?  match-value
    |  +- rising-target*  target
    |  +- falling-value?  match-value
    |  +- falling-target*  target
    |  +- delta-rising-value?  match-value
    |  +- delta-rising-target*  target
    |  +- delta-falling-value?  match-value
```
++-- delta-falling-target*   target
++-- startup?                enumeration
grouping trigger-grouping
++-- (test)?
++--:(existences)
|  ++-- existences
|     ++-- test-type?   enumeration
|     ++-- target*     target
++--:(boolean)
|  ++-- boolean
|     ++-- operator?   operator
|     ++-- value?      match-value
|     ++-- target*     target
++--:(threshold)
++-- threshold
++-- rising-value?           match-value
++-- rising-target*          target
++-- falling-value?          match-value
++-- falling-target*         target
++-- delta-rising-value?     match-value
++-- delta-rising-target*    target
++-- delta-falling-value?    match-value
++-- delta-falling-target*   target
++-- startup?                enumeration

module: ietf-event
++--rw events
++--rw event* [event-name type]
  ++--rw event-name           string
  ++--rw type                identityref
  ++--rw event-description?  string
  ++--rw group-id?           group-type
  ++--rw target*             trig:target
  ++--rw clear?              boolean
  ++--rw trigger* [name]
      ++--rw name               string
      ++--rw trigger-description?  string
      ++--rw call-event?        -> ../../event-name
      ++--rw frequency
          ++--rw type?         identityref
          ++--rw periodic
              |  ++--rw interval    uint32
              |  ++--rw start?      yang:date-and-time
              |  ++--rw end?        yang:date-and-time
              ++--rw scheduling
                  ++--rw month*     string
                  ++--rw day-of-month*  uint8
                  ++--rw day-of-week*  uint8
The relation between Event, Trigger, Target and Action is described as follows:
One event may trigger another event, i.e., the action output in the first event can be input to target in the second event and a set of events can be grouped together and executed in a coordinated manner, but if it does not trigger another event, the relation between two events should be ignored.
6. EVENT TRIGGER YANG Module

<CODE BEGINS> file "ietf-event-trigger@2019-10-28.yang"
module ietf-event-trigger {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-event-trigger";
  prefix trig;

  import ietf-yang-types {
    prefix yang;
  }
  organization
    "IETF xxx Working Group";
  contact
    "Zitao Wang: wangzitao@huawei.com
    Qin Wu: bill.wu@huawei.com"
  description
    "This module defines a reusable grouping for event trigger.";

  revision 2019-10-28 {
    description
      "Initial revision.";
    reference
      "foo";
  }

typedef match-value {
  type union {
    type yang:xpath1.0;
    type yang:object-identifier;
    type string;
  }
  description
    "This type is used to match resources of type ’target’. Since the type ’target’ is a union of different types, the ’match-value’ type is also a union of corresponding types.";
}

typedef target {
  type union {
    type instance-identifier;
    type yang:object-identifier;
    type yang:uuid;
    type string;
  }
  description
    "If the target is modelled in YANG, this type will
be an instance-identifier. If the target is an SNMP object, the type will be an object-identifier.
If the target is anything else, for example a distinguished name or a CIM path, this type will be a string.
If the target is identified by a UUID use the uuid type.
If the server supports several models, the precedence should be in the order as given in the union definition."

typedef operator {
    type enumeration {
        enum unequal {
            description
            "Indicates that the comparison type is unequal to.";
        }
        enum equal {
            description
            "Indicates that the comparison type is equal to.";
        }
        enum less {
            description
            "Indicates that the comparison type is less than.";
        }
        enum less-or-equal {
            description
            "Indicates that the comparison type is less than or equal to.";
        }
        enum greater {
            description
            "Indicates that the comparison type is greater than.";
        }
        enum greater-or-equal {
            description
            "Indicates that the comparison type is greater than or equal to.";
        }
    }
    description
    "definition of the operator";
}

grouping existences-trigger {
    description
    "A grouping that provides existence trigger";
    container existences {
leaf test-type {
  type enumeration {
    enum absent {
      description "If the test type is Absent, the system triggers an event and takes the specified action when the monitored object disappears.";
    }
    enum present {
      description "If the test type is Present, the system triggers an event and takes the specified action when the monitored object appears.";
    }
    enum changed {
      description "If the test type is Changed, the system triggers an alarm event and takes the specified action when the value of the monitored object changes.";
    }
  }
  description "test type.";
}
leaf-list target {
  type target;
  description "List for target objects";
  description "Container for existence";
}

grouping boolean-trigger {
  description "A grouping that provides boolean trigger";
  container boolean {
    leaf operator {
      type operator;
      description "Comparison type.";
    }
    leaf value {
      type match-value;
      description "Comparison value which is static threshold value.";
    }
    leaf-list target {
      type target;
      description "List for target management objects.";
    }
  }
  description "Container for existence";
}

grouping boolean-trigger {
  description "A grouping that provides boolean trigger";
  container boolean {
    leaf operator {
      type operator;
      description "Comparison type.";
    }
    leaf value {
      type match-value;
      description "Comparison value which is static threshold value.";
    }
    leaf-list target {
      type target;
      description "List for target management objects.";
    }
  }
  description "Container for existence";
}


}  

description
  "Container for boolean test.";

}

}


grouping threshold-trigger {
  description
    "A grouping that provides threshold trigger";

container threshold {
  leaf rising-value {
    type match-value;
    description
      "Sets the rising threshold to the specified value,
        when the current sampled value is greater than or equal to
        this threshold, and the value at the last sampling interval
        was less than this threshold, the event is triggered. ";
  }

  leaf-list rising-target {
    type target;
    description
      "List for target objects.";
  }

  leaf falling-value {
    type match-value;
    description
      "Sets the falling threshold to the specified value."
  }

  leaf-list falling-target {
    type target;
    description
      "List for target objects.";
  }

  leaf delta-rising-value {
    type match-value;
    description
      "Sets the delta rising threshold to the specified value.";
  }

  leaf-list delta-rising-target {
    type target;
    description
      "List for target objects.";
  }

  leaf delta-falling-value {
    type match-value;
    description
      "Sets the delta falling threshold to the specified value.";
  }

}
leaf-list delta-falling-target {
  type target;
  description
    "List for target objects."
}
leaf startup {
  type enumeration {
    enum rising {
      description
        "If the first sample after this
        managed object becomes active is greater than or equal
        to ‘rising-value’ and the ‘startup’ is equal to
        ‘rising’ then one threshold rising event is
        triggered for that managed object."
    }
    enum falling {
      description
        "If the first sample after this managed object becomes
        active is less than or equal to ‘falling-value’ and
        the ‘startup’ is equal to ‘falling’ then one
        threshold falling event is triggered for that managed
        object."
    }
    enum rising-or-falling {
      description
        "That event may be triggered when the
        ‘startup’ is equal to ‘rising-or-falling’.
        ‘rising-or-falling’ indicate the state value of the
        managed object may less than or greater than the
        specified threshold value."
    }
  }
  description
    "Startup setting."
}
description
  "Container for the threshold trigger condition.
  Note that the threshold here may change over time
  or the state value changes in either ascend order
  or descend order."
}

grouping trigger-grouping {
  description
    "A grouping that provides event trigger."
  choice test {
    description

"Choice test";
case existences {
    uses existences-trigger;
}
case boolean {
    uses boolean-trigger;
}
case threshold {
    uses threshold-trigger;
}
}
</CODE ENDS>

7. EVENT YANG Module

<CODE BEGINS> file "ietf-event@2019-10-28.yang"

module ietf-event {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-event";
    prefix evt;

    import ietf-yang-types {
        prefix yang;
    }
    import ietf-event-trigger {
        prefix trig;
    }

    organization
        "IETF xxx Working Group";
    contact
        "Zitao Wang: wangzitao@huawei.com
        Qin Wu: bill.wu@huawei.com";
    description
        "This module defines a model for the service topology.";

    revision 2019-10-28 {
        description
            "Initial revision.";
        reference
            "foo";
    }

    identity event-type {
        description


"Base identity for event type";
}

identity frequency {
  description
  "Base identity for frequency";
}

identity periodic {
  base frequency;
  description
  "Identity for periodic trigger";
}

identity scheduling {
  base frequency;
  description
  "Identity for scheduling trigger";
}

identity logging {
  description
  "Base identity for logging action";
}

identity logging-notification {
  base logging;
  description
  "Logging for event notification";
}

identity logging-set {
  base logging;
  description
  "Logging for reset values";
}

typedef logging-type {
  type identityref {
    base logging;
  }
  description
  "Logging types";
}

typedef group-type {
  type string;
  description

"Group type";
}

grouping start-end-grouping {
  description
    "A grouping that provides start and end times for
    Event objects.";
  leaf start {
    type yang:date-and-time;
    description
      "The date and time when the Event object
      starts to create triggers.";
  }
  leaf end {
    type yang:date-and-time;
    description
      "The date and time when the Event object
      stops to create triggers.
      It is generally a good idea to always configure
      an end time and to refresh the end time as needed
      to ensure that agents that lose connectivity to
      their Controller do not continue executing Schedules
      forever.";
  }
}

container events {
  list event {
    key "event-name type";
    leaf event-name {
      type string;
      description
        "Event name";
    }
    leaf type {
      type identityref {
        base event-type;
      }
      description
        "Type of event";
    }
    leaf event-description {
      type string;
      description
        "Event description";
    }
    leaf group-id {
      type group-type;
    }
  }
}
description
    "Group Identifier";
}
leaf-list target {
    type trig:target;
    description
        "targeted objects";
}
leaf clear {
    type boolean;
    default "false";
    description
        "A flag indicate whether the event be closed";
}
list trigger {
    key "name";
    leaf name {
        type string;
        description
            "Trigger name";
    }
    leaf trigger-description {
        type string;
        description
            "Trigger description";
    }
    leaf call-event {
        type leafref {
            path "../../../event-name";
        }
        description
            "This leaf call sub-event.";
    }
}
container frequency {
    leaf type {
        type identityref {
            base frequency;
        }
        description
            "Type of trigger frequency";
    }
    container periodic {
        when "derived-from-or-self(../type, 'periodic')";
        description
            "A periodic timing object triggers periodically
             according to a regular interval.";
        leaf interval {
            type uint32 {
range "1..max";
}  
units "seconds";
mandatory true;
description  
"The number of seconds between two triggers  
generated by this periodic timing object.";
}  
uses start-end-grouping;
}
container scheduling {
  when "derived-from-or-self(../type, 'scheduling')";
description  
"A scheduling timing object triggers.";
leaf-list month {
  type string;
description  
"A set of months at which this scheduling timing  
will trigger.";
}
leaf-list day-of-month {
  type uint8 {
    range "0..59";
  }
description  
"A set of days of the month at which this  
scheduling timing will trigger.";
}
leaf-list day-of-week {
  type uint8 {
    range "0..59";
  }
description  
"A set of weekdays at which this scheduling timing  
will trigger.";
}
leaf-list hour {
  type uint8 {
    range "0..59";
  }
description  
"A set of hours at which the scheduling timing will  
trigger.";
}
leaf-list minute {
  type uint8 {
    range "0..59";
  }
description
   "A set of minutes at which this scheduling timing
    will trigger.";
}
leaf-list second {
    type uint8 {
        range "0..59";
    }
    description
    "A set of seconds at which this calendar timing
     will trigger.";
}
uses start-end-grouping;
}
description
   "Container for frequency";
}
uses trig:trigger-grouping;
description
   "List for trigger";
}
container actions {
    list action {
        key "name";
        leaf name {
            type string;
            description
            "Action Name";
        }
        container set {
            leaf target {
                type trig:target;
                description
                "The target objects";
            }
            anydata value {
                description
                "Inline set content.";
            }
            description
            "Set a value to the target";
        }
        container logging {
            leaf type {
                type logging-type;
                description
                "Specifies the log action";
            }
        }
    }
}
8. Security Considerations

The YANG modules defined in this document MAY be accessed via the RESTCONF protocol ([RFC8040]) or NETCONF protocol ([RFC6241]). The lowest RESTCONF or NETCONF layer requires that the transport-layer protocol provides both data integrity and confidentiality, see Section 2 in [RFC8040] and [RFC6241]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) ([RFC6242]). The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS ([RFC5246]).

The NETCONF access control model ([RFC6536]) provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /events/event/event-name
- /events/event/target
IANA Considerations

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made:

```
+-----------------------------------------------+-----------------------------------------------+
| XML: N/A, the requested URI is an XML namespace.     |                                               |
| URI: urn:ietf:params:xml:ns:yang:ietf-event      | Registrant Contact: The IESG.                 |
| XML: N/A, the requested URI is an XML namespace.     |                                               |
```

This document registers two YANG modules in the YANG Module Names registry [RFC6020].

```
+-------------------+-------------------+-------------------+-------------------+
| Name:             | Namespace:        | Prefix:           | Reference:        |
```

Acknowledges

This work has benefited from the discussions of ECA Policy over the years. In particular, the SUPA project [https://datatracker.ietf.org/wg/supa/about/] provided approaches to express high-level, possibly network-wide policies to a network management function (within a controller, an orchestrator, or a network element).

Igor Bryskin, Xufeng Liu, Alexander Clemm, Henk Birkholz, Tianran Zhou contributed to an earlier version of [GNCA]. We would like to thank the authors of that document on event response behaviors.
11. Contributors

Nicola Sambo  
Scuola Superiore Sant’Anna  
Via Moruzzi 1  
Pisa  56124  
Italy  
Email: nicola.sambo@sssup.it

Giuseppe Fioccola  
Huawei Technologies  
Riesstrasse, 25  
Munich  80992  
Germany  
Email: giuseppe.fioccola@huawei.com

12. Normative References


Appendix A. Usage Example of ECA model Working with YANG PUSH

The relation between Event and YANG PUSH is described as follow: YANG Push Notification may trigger one event, i.e. one trigger conditions of the Event A can be set to "receiver received a yang push notification", and it can associates with other conditions of the Event A. When these conditions are met, the event A is triggered.
For Example:

The receiver received a push-change-update notification and learned that the "oper-status" of interface[name='eth0'] changed.

The target of Event "interface-state-monitoring" is set to "/if:interfaces/if:interface[if:name='eth0']", the trigger list contains two conditions: 1) receiver received a push-change-update notification; 2) the value of "in-errors" of interface[name='eth0'] exceeded the pre-configured threshold. When these conditions are met, corresponding action will be performed, i.e. disable interface[name='eth0']. The XML examples are shown as below:

```xml
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2017-10-25T08:22:33.44Z</eventTime>
  <push-change-update
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <id>89</id>
    <datastore-changes>
      <yang-patch>
        <patch-id>0</patch-id>
        <edit>
          <edit-id>edit1</edit-id>
```
<operation>replace</operation>
<target>/ietf-interfaces:interfaces</target>
[value>
  <interfaces
    xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
      <name>eth0</name>
      <oper-status>up</oper-status>
    </interface>
  </interfaces>
</value>
</edit>
</yang-patch>
</push-change-update>
</notification>

<event>
  <event-name>interface-state-monitoring</event-name>
  <type>interface-exception</type>
  <target>/if:interfaces/if:interface[if:name='eth0']</target>
  <trigger>
    <name>state-push-change</name>
    <trigger-description>received yang push \changed notification</trigger-description>
    <test>
    </test>
  </trigger>
  <trigger>
    <name>evaluate-in-errors</name>
    <call-event>interface-state-chang</call-event>
    <trigger-description>evaluate the number of the packets that contained errors</trigger-description>
    <frequency>10m</frequency>
    <test>
      <boolean>
        <operator>greater-or-equal</operator>
        <value>100</value>
        <target>/if:interfaces/if:interface[if:name='eth0']/if:statistic/if:in-errors</target>
      </boolean>
    </test>
  </trigger>
</event>

Appendix B. Usage Example of Reusing Trigger-Grouping

The "ietf-event-trigger.yang" module defines a set of groupings for a generic trigger. It is intended that these groupings can be reused by other models that require the trigger conditions, for example, in some subscription and notification cases, many applications do not require every update, only updates that are of certain interest. The following example describe how to reuse the "ietf-event-trigger" module to define the subscription and notification smarter filter.

```yaml
import ietf-subscribed-notifications {
  prefix sn;
}
import ietf-event-trigger {
  prefix trig;
}
augment "/sn:subscriptions/sn:subscription" {
  description "add the smart filter container";
  container smart-filter {
    description "It concludes filter configurations";
    uses trig:trigger-grouping;
  }
}
```

The tree diagrams:
module: ietf-smart-filter
augment /sn:subscriptions/sn:subscription:
  +--rw smart-filter
    +--rw (test)?
      +--:(existences)
        |  +--rw existences
        |     +--rw target* target
        +--:(boolean)
          |  +--rw boolean
          |     +--rw operator? operator
          |     +--rw value? match-value
          |     +--rw target* target
          +--:(variation)
            +--rw variation
              +--rw rising-value? match-value
              +--rw rising-target* target
              +--rw falling-value? match-value
              +--rw falling-target* target
              +--rw delta-rising-value? match-value
              +--rw delta-rising-target* target
              +--rw delta-falling-value? match-value
              +--rw delta-falling-target* target
              +--rw startup? enumeration

Appendix C. Changes between Revisions

v03 - v04

- Add text in introduction section to clarify the usage examples of ECA policy
- Update objective section to align with use cases.
- Clarify the relationship between target and policy variable.
- Change variation trigger condition back into threshold trigger condition and clarify the usage of three trigger conditions.
- Remove Event MIB related section.
- Add new coauthors.

v02 - v03

- Usage Example Update: add an usage example to introduce how to reuse the ietf-event-trigger module to define the subscription-notification smarter filter.
v01 - v02

- Introduce the group-id which allow group a set of events that can be executed together.
- Change threshold trigger condition into variation trigger condition to further clarify the difference between boolean trigger condition and variation trigger condition.
- Module structure optimization.
- Usage Example Update.

v00 - v01

- Separate ietf-event-trigger.yang from Event management model and ietf-event.yang and make it reusable in other YANG models.
- Clarify the difference between boolean trigger condition and threshold trigger condition.
- Change evt-smp-min and evt-smp-max into min-data-object and max-data-object in the data model.

Authors’ Addresses

Michael Wang
Huawei Technologies, Co., Ltd
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: wangzitao@huawei.com

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

Email: bill.wu@huawei.com

Chongfeng Xie
China Telecom

Email: xiechf@ctbri.com.cn