NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 12, 2020

M. Wang
Q. Wu
Huawei
I. Bryskin
Individual
X. Liu
Volta Networks
B. Claise
Cisco

December 10, 2019

# A YANG Data model for ECA Policy Management
**draft-wwx-netmod-event-yang-06**

## Abstract

RFC8328 defines a policy-based management framework that allows 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, Event Condition Action (ECA) policy is an typical example of imperative policy. This document defines a YANG data model for the ECA policy management. The ECA policy YANG provides the ability for the network management function (within a network element) to control the configuration and monitor state change and take simple and instant action on the server when a trigger condition on the system state is met.

## Status of This Memo

Table of Contents

## 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.  Event Condition Action (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:

o  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.

o  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.

o  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.

o  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.

o  To set up an LSP and reserve resources within the network via NETCONF protocol operation, Path Computation API RPC model can be

invoked to calculate a path meeting end-to-end network performance criteria.

This document defines a ECA Policy management YANG data model. The ECA Policy YANG provides the ability for the network management function (within a network element) to control the configurations and monitor state parameters and take simple and instant action on the server 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.

The following terms are defined in [RFC7950] [RFC3460] and are not redefined here:

- Server
- Client
- Policy variable
- Policy value
- Implicit policy variable
- explicit policy variable

This document uses the following terms:

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 untilization crossing a threshold, network connection setup, an external input to the system, for example.

Condition: Condition can be seen as a logical test that, if satisfied or evaluated to be true, cause the action to be carried out.

Action: Updates or invocations on local managed object attributes.

## 2.2. Tree Diagrams

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

## 3. 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:

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

o 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.

o 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 addtion, 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.

## 4. Overview of ECA YANG Data Model

A ECA policy rule is read as: when event occurs in a situation where condition is true, then action is executed. Therefore ECA comprises three key elements: event, associated conditions, and associated actions. These three elements should be pushed down and configured on the server by the client. If the action is rejected by the server duing ECA policy execution, the action should be rolled back and cleaned up.

### 4.1. ECA Policy Variable and Value

ECA policy variable (PV) generically represents information that changes (or "varies"), and that is set or evaluated by software. ECA policy Value is used for modeling values and constants used in policy conditions and actions. In policy, conditions and actions can abstract information as "policy variables" to be evaluated in logical expressions, or set by actions, e.g., the Policy Condition has the semantics "variable matches value" while Policy Action has the semantics "set variable to value".

In ECA, two type of policy variables are defined, implicit variable and explicit variable. Explicit variables are bound to exact data object instance in the model while implicit variables are defined and evaluated outside of a model. Each ECA policy variable has the following attributes:

o Name with Globally unique or ECA unique scope ;

o Type either implicit or explicit; The implicit or explicit type can be further broken down into global or local.

o Value data stored in the policy variable structured according to the PV type. This structure can be used to keep intermediate results/meta data during the execution of an ECA policy.

The following operations are allowed with/on a PV:

o initialize (with a constant/enum/identity);

o  set (with contents of another same type PV);

o  read (retrieve datastore contents pointed by the specified same type XPath/sub-tree);

o  write (modify configuration data in the datastore with the PV's content/value);

o  insert (PV's content into a same type list);

o  iterate (copy into PV one by one same type list elements)

o  function calls in a form of F(arg1,arg2,...), where F is an identity of a function from extendable function library, arg1,arg2,etc are PVs respectively, the function's input parameters, with the result returned in result policy variable.

PVs could be used as input/output of an ECA invoked RPC and policy argument in the func calls.  PVs could also be a source of information sent to the client in notification messages.

PVs could be used in condition expressions

The model structure for the Policy Variable is shown below:

```
  +--rw policy-variables
  |  +--rw policy-variable* [name]
  |     +--rw name                  string
  |     +--rw type?                 identityref
  |     +--rw explict-variable?   yang:xpath1.0
  |     +--rw implict-variable?   identityref
  |     +--rw policy-value?       union
```

## 4.2.  ECA Event

The ECA event are used to keep track of state of changes associated with one of multiple operational state data objects in the network device.  Typical examples of ECA event include a fault, an alarm, a change in network state, network security threat, hardware malfunction, buffer utilization crossing a threshold, network connection setup, and an external input to the system.

Each ECA Event has the following attributes:

o  name, the name of ECA event;

o  type, either one time or peridic scheduling;

o  group-id, which can be used to group a set of events that can be executed together,e.g., deliver a service or provide service assurance;

o  scheduled-time,configuration scheduling - scheduling one time or periodic.

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.

All events specified in the ECA policy model are continuously monitored by the server.

The model structure for the ECA Event is shown below:

```
     +--rw event* [name type]
        +--rw name                 string
        +--rw type                 identityref
        +--rw event-description?   string
        +--rw group-id?            group-type
        +--rw explict-variable*    leafref
        +--rw clear?               boolean
        +--rw scheduled-time
        |  +--rw type?                  identityref
        |  +--rw periodic
        |  |  +--rw interval    uint32
        |  |  +--rw start?      yang:date-and-time
        |  |  +--rw end?        yang:date-and-time
        |  +--rw calendar-time
        |     +--rw month*          string
        |     +--rw day-of-month*   uint8
        |     +--rw day-of-week*    uint8
        |     +--rw hour*           uint8
        |     +--rw minute*         uint8
        |     +--rw second*         uint8
        |     +--rw start?          yang:date-and-time
        |     +--rw end?            yang:date-and-time
        +--ro last-event?          -> /eca/event/name
```

## 4.3. ECA Condition

Condition can be seen as a logical test that, if satisfied or evaluated to be true, cause the action to be carried out. In this model, condition can be specified as logical combinations of the following three condition expressions:

- Existence: An existence condition 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 existence test type is Absent, the system triggers a network event and takes the specified action when the monitored object disappears.

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

  * If the existence 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 comparision hierarchy is logical hierarchies specified in a form of:

```
<policy-variable> <relation> <policy-value> or
<policy-variable1> <relation> <policy-variable2>
```

relation is one of the comparison operations from the set:
==, !=, >, <, >=, <=

- The operation 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 the value of the monitored object with the threshold values , with one of the following mechanisms:

  * A rising event is triggered if the value of the monitored object is greater than or equal to the rising threshold.

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

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

* A falling 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 an event only for the first crossing.

In addition, logical operation type can be used to describe complex logical operations between different condition lists under the same event, for example, (condition A & condition B) or condition C.

The model structure for the condition is shown below:

```
          +--rw condition* [name]
          |  +--rw name                      string
          |  +--rw condition-description?    string
          |  +--rw logical-operation-type?   identityref
          |  +--rw call-event?               -> ../../name
          |  +--rw (test)?
          |     +--:(existences)
          |     |  +--rw existences
          |     |     +--rw type?              enumeration
          |     |     +--rw policy-variable?   leafref
          |     +--:(boolean)
          |     |  +--rw boolean
          |     |     +--rw operator?          operator
          |     |     +--rw policy-value
          |     |     |  +--rw policy-argument
          |     |     |     +--rw (argument)?
          |     |     |        +--:(explict-variable)
          |     |     |        |  +--rw explict-variable?   leafref
          |     |     |        +--:(implict-variable)
          |     |     |        |  +--rw implict-variable?   leafref
          |     |     |        +--:(value)
          |     |     |           +--rw policy-value?       leafref
          |     |     +--rw policy-variable
          |     |        +--rw policy-argument
          |     |           +--rw (argument)?
          |     |              +--:(explict-variable)
          |     |              |  +--rw explict-variable?   leafref
          |     |              +--:(implict-variable)
          |     |                 +--rw implict-variable?   leafref
          |     +--:(threshold)
          |        +--rw threshold
          |           +--rw rising-value?                   leafref
          |           +--rw rising-policy-variable*         leafref
          |           +--rw falling-value?                  leafref
          |           +--rw falling-policy-variable*        leafref
          |           +--rw delta-rising-value?             leafref
          |           +--rw delta-rising-policy-variable*   leafref
          |           +--rw delta-falling-value?            leafref
          |           +--rw delta-falling-policy-variable*  leafref
          |           +--rw startup?                        enumeration
```

## 4.4. ECA Action

The action list consists of updates or invocations on local managed object attributes and a set of actions are defined as follows, which will be performed when the corresponding event is triggered:

- o  sending one time log notification

- (-re)configuration - modifying a configuration data in the conventional configuration datastore.

- adding/removing event notify subscription (essentially, the same action as performed when a client explicitly adds/removes a subscription)

- executing an RPC defined by a YANG module supported by the server (the same action as performed when a client interactively calls the RPC);

- performing operations and function calls on PVs (such as assign, read, insert, iterate, etc);

Multiple ECA Actions could be triggered by a single ECA event.

Any given ECA Condition or Action may appear in more than one ECAs.

The model structure for the actions is shown below:

```
        +--rw actions
           +--rw action* [name]
              +--rw name                        string
              +--rw (action-type)?
                 +--:(set)
                 |  +--rw set
                 |     +--rw policy-variable?   leafref
                 |     +--rw value?             <anydata>
                 +--:(logging)
                 |  +--rw logging
                 |     +--rw type?              logging-type
                 |     +--rw policy-variable?   leafref
                 +--:(function-call)
                 |  +--rw function-call
                 |     +--rw function-type?     identityref
                 |     +--rw policy-argument* [name]
                 |     |  +--rw name                        string
                 |     |  +--rw (argument)?
                 |     |     +--:(explict-variable)
                 |     |     |  +--rw explict-variable?   leafref
                 |     |     +--:(implict-variable)
                 |     |     |  +--rw implict-variable?   leafref
                 |     |     +--:(value)
                 |     |        +--rw policy-value?       leafref
                 |     +--rw result
                 |        +--rw (argument)?
                 |           +--:(explict-variable)
                 |           |  +--rw explict-variable?   leafref
```

```
                    |           +--:(implict-variable)
                    |           |  +--rw implict-variable?   leafref
                    |           +--:(value)
                    |              +--rw policy-value?       leafref
                    +--:(rpc-call)
                       +--rw rpc-call
                          +--rw name?     string
                          +--rw input
                          |  +--rw policy-argument* [name]
                          |     +--rw name
                          |     |       string
                          |     +--rw (argument)?
                          |        +--:(explict-variable)
                          |        |  +--rw explict-variable?   leafref
                          |        +--:(implict-variable)
                          |        |  +--rw implict-variable?   leafref
                          |        +--:(value)
                          |           +--rw policy-value?       leafref
                          +--rw output
                             +--rw policy-argument* [name]
                                +--rw name
                                |       string
                                +--rw (argument)?
                                   +--:(explict-variable)
                                   |  +--rw explict-variable?   leafref
                                   +--:(implict-variable)
                                   |  +--rw implict-variable?   leafref
                                   +--:(value)
                                      +--rw policy-value?       leafref
```

## 5.  ECA YANG Model (Tree Structure)

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

```
  grouping start-end-grouping
    +-- start?   yang:date-and-time
    +-- end?     yang:date-and-time
  grouping existences-trigger
    +-- existences
       +-- type?              enumeration
       +-- policy-variable?
               -> /policy-variables/policy-variable/name
  grouping boolean-trigger
    +-- boolean
       +-- operator?          operator
       +-- policy-value
       |  +-- policy-argument
```

```
          |     +-- (argument)?
          |        +--:(explict-variable)
          |        |  +-- explict-variable?   leafref
          |        +--:(implict-variable)
          |        |  +-- implict-variable?   leafref
          |        +--:(value)
          |           +-- policy-value?       leafref
          +-- policy-variable
             +-- policy-argument
                +-- (argument)?
                   +--:(explict-variable)
                   |  +-- explict-variable?   leafref
                   +--:(implict-variable)
                      +-- implict-variable?   leafref
     grouping threshold-trigger
       +-- threshold
          +-- rising-value?
          |       -> /policy-variables/policy-variable/policy-value
          +-- rising-policy-variable*
          |       -> /policy-variables/policy-variable/name
          +-- falling-value?
          |       -> /policy-variables/policy-variable/policy-value
          +-- falling-policy-variable*
          |       -> /policy-variables/policy-variable/name
          +-- delta-rising-value?
          |       -> /policy-variables/policy-variable/policy-value
          +-- delta-rising-policy-variable*
          |       -> /policy-variables/policy-variable/name
          +-- delta-falling-value?
          |       -> /policy-variables/policy-variable/policy-value
          +-- delta-falling-policy-variable*
          |       -> /policy-variables/policy-variable/name
          +-- startup?                         enumeration
     grouping trigger-grouping
       +-- (test)?
          +--:(existences)
          |  +-- existences
          |     +-- type?              enumeration
          |     +-- policy-variable?
          |             -> /policy-variables/policy-variable/name
          +--:(boolean)
          |  +-- boolean
          |     +-- operator?          operator
          |     +-- policy-value
          |     |  +-- policy-argument
          |     |     +-- (argument)?
          |     |        +--:(explict-variable)
          |     |        |  +-- explict-variable?   leafref
```

```
          |     |        +--:(implict-variable)
          |     |        |  +-- implict-variable?   leafref
          |     |        +--:(value)
          |     |           +-- policy-value?       leafref
          |     +-- policy-variable
          |        +-- policy-argument
          |           +-- (argument)?
          |              +--:(explict-variable)
          |              |  +-- explict-variable?   leafref
          |              +--:(implict-variable)
          |                 +-- implict-variable?   leafref
          +--:(threshold)
             +-- threshold
                +-- rising-value?                   leafref
                +-- rising-policy-variable*
                |       -> /policy-variables/policy-variable/name
                +-- falling-value?                  leafref
                +-- falling-policy-variable*
                |       -> /policy-variables/policy-variable/name
                +-- delta-rising-value?             leafref
                +-- delta-rising-policy-variable*
                |       -> /policy-variables/policy-variable/name
                +-- delta-falling-value?            leafref
                +-- delta-falling-policy-variable*
                |       -> /policy-variables/policy-variable/name
                +-- startup?                        enumeration

   module: ietf-eca
     +--rw policy-variables
     |  +--rw policy-variable* [name]
     |     +--rw name                 string
     |     +--rw type?                identityref
     |     +--rw explict-variable?   yang:xpath1.0
     |     +--rw implict-variable?   identityref
     |     +--rw policy-value?       union
     +--rw eca
        +--rw event* [name type]
           +--rw name                  string
           +--rw type                  identityref
           +--rw event-description?   string
           +--rw group-id?             group-type
           +--rw explict-variable*    leafref
           +--rw clear?                boolean
           +--rw scheduled-time
           |  +--rw type?                   identityref
           |  +--rw periodic
           |  |  +--rw interval    uint32
           |  |  +--rw start?      yang:date-and-time
```

```
           |  |  +--rw end?        yang:date-and-time
           |  +--rw calendar-time
           |     +--rw month*          string
           |     +--rw day-of-month*   uint8
           |     +--rw day-of-week*    uint8
           |     +--rw hour*           uint8
           |     +--rw minute*         uint8
           |     +--rw second*         uint8
           |     +--rw start?          yang:date-and-time
           |     +--rw end?            yang:date-and-time
           +--ro last-event?          -> /eca/event/name
           +--ro last-condition?      -> /eca/event/condition/name
           +--ro last-action?
           |       -> /eca/event/actions/action/name
           +--rw condition* [name]
           |  +--rw name                      string
           |  +--rw condition-description?    string
           |  +--rw logical-operation-type?   identityref
           |  +--rw call-event?               -> ../../name
           |  +--rw (test)?
           |     +--:(existences)
           |     |  +--rw existences
           |     |     +--rw type?              enumeration
           |     |     +--rw policy-variable?   leafref
           |     +--:(boolean)
           |     |  +--rw boolean
           |     |     +--rw operator?          operator
           |     |     +--rw policy-value
           |     |     |  +--rw policy-argument
           |     |     |     +--rw (argument)?
           |     |     |        +--:(explict-variable)
           |     |     |        |  +--rw explict-variable?   leafref
           |     |     |        +--:(implict-variable)
           |     |     |        |  +--rw implict-variable?   leafref
           |     |     |        +--:(value)
           |     |     |           +--rw policy-value?       leafref
           |     |     +--rw policy-variable
           |     |        +--rw policy-argument
           |     |           +--rw (argument)?
           |     |              +--:(explict-variable)
           |     |              |  +--rw explict-variable?   leafref
           |     |              +--:(implict-variable)
           |     |                 +--rw implict-variable?   leafref
           |     +--:(threshold)
           |        +--rw threshhold
           |           +--rw rising-value?                  leafref
           |           +--rw rising-policy-variable*          leafref
           |           +--rw falling-value?                 leafref
```

```
           |           +--rw falling-policy-variable*         leafref
           |           +--rw delta-rising-value?              leafref
           |           +--rw delta-rising-policy-variable*    leafref
           |           +--rw delta-falling-value?             leafref
           |           +--rw delta-falling-policy-variable*   leafref
           |           +--rw startup?
           |                   enumeration
           +--rw actions
              +--rw action* [name]
                 +--rw name                   string
                 +--rw (action-type)?
                    +--:(set)
                    |  +--rw set
                    |     +--rw policy-variable?   leafref
                    |     +--rw value?             <anydata>
                    +--:(logging)
                    |  +--rw logging
                    |     +--rw type?              logging-type
                    |     +--rw policy-variable?   leafref
                    +--:(function-call)
                    |  +--rw function-call
                    |     +--rw function-type?     identityref
                    |     +--rw policy-argument* [name]
                    |     |  +--rw name                         string
                    |     |  +--rw (argument)?
                    |     |     +--:(explict-variable)
                    |     |     |  +--rw explict-variable?   leafref
                    |     |     +--:(implict-variable)
                    |     |     |  +--rw implict-variable?   leafref
                    |     |     +--:(value)
                    |     |        +--rw policy-value?       leafref
                    |     +--rw result
                    |        +--rw (argument)?
                    |           +--:(explict-variable)
                    |           |  +--rw explict-variable?   leafref
                    |           +--:(implict-variable)
                    |           |  +--rw implict-variable?   leafref
                    |           +--:(value)
                    |              +--rw policy-value?       leafref
                    +--:(rpc-call)
                       +--rw rpc-call
                          +--rw name?     string
                          +--rw input
                          |  +--rw policy-argument* [name]
                          |     +--rw name
                          |     |       string
                          |     +--rw (argument)?
                          |        +--:(explict-variable)
```

```
                          |        |  +--rw explict-variable?   leafref
                          |        +--:(implict-variable)
                          |        |  +--rw implict-variable?   leafref
                          |        +--:(value)
                          |           +--rw policy-value?       leafref
                          +--rw output
                             +--rw policy-argument* [name]
                                +--rw name
                                |       string
                                +--rw (argument)?
                                   +--:(explict-variable)
                                   |  +--rw explict-variable?   leafref
                                   +--:(implict-variable)
                                   |  +--rw implict-variable?   leafref
                                   +--:(value)
                                      +--rw policy-value?       leafref
```

## 6.  ECA YANG Module

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

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

  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NETMOD Working Group";
  contact
    "Editor:  Zitao Wang
              <mailto:wangzitao@huawei.com>
     Editor:  Qin Wu
              <mailto:bill.wu@huawei.com>
     Editor:  Igor Bryskin
              <mailto:Igor.Bryskin@huawei.com>
     Editor:  Xufeng Liu
              <mailto:xufeng.liu.ietf@gmail.com>
     Editor:  Benoit Claise
               <mailto:bclaise@cisco.com>";
  description
    "  This module contains YANG specifications for ECA Policy management.
       Copyright (c) 2019 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.
```

```
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject to
        the license terms contained in, the Simplified BSD License set
        forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (https://trustee.ietf.org/license-info).
       This version of this YANG module is part of RFC xxxx; see the
       RFC itself for full legal notices.";

  revision 2019-10-28 {
    description
      "Initial revision.";
    reference
      "RFC xxxx";
  }
  identity variable-type {
   description
   "base variable type";
  }
  identity global-explict {
   base variable-type;
   description
   "Identity for global explict variable";
  }
  identity global-implict {
   base variable-type;
   description
   "Identity for global explict variablae";
  }
  identity local-explict {
   base variable-type;
   description
   "Identity for local explict variable";
  }
  identity local-implict {
   base variable-type;
   description
   "Identity for local implict variable";
  }
  identity function-type {
    description
      "Possible values are:
       plus, minus, mult, divide, remain.";
  }

  identity logical-operation-type {
    description
      "Possible values are:
```

```
         not, or, and.";
    }

    identity policy-variable-type {
      description
        "Possible values are:
         boolean, int32, int64, uint32, uint64, string, etc.";
    }

    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.";
    }
  }

  typedef operator {
    type enumeration {
      enum unequal {
        description
          "Indicates that the comparision type is unequal to.";
      }
      enum equal {
        description
          "Indicates that the comparision type is equal to.";
      }
      enum less {
```

```
            description
              "Indicates that the comparision type is less than.";
          }
          enum less-or-equal {
            description
              "Indicates that the comparision type is less than
               or equal to.";
          }
          enum greater {
            description
              "Indicates that the comparision type is greater than.";
          }
          enum greater-or-equal {
            description
              "Indicates that the comparision 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 type {
            type enumeration {
              enum match {
              description "march";
              }
              enum mismatch {
              description "mismatch";
              }
            }
            description
              "existence type, variable match the value or variable mismatch the
value";
          }
          leaf policy-variable {
            type leafref {
              path "/policy-variables/policy-variable/name";
            }
            description
              "Policy variable";
          }
          description
            "Container for existence";
```

```
    }
```

```
  }

  grouping boolean-trigger {
    description
      "A grouping that provides boolean trigger";
    container boolean {
      leaf operator {
        type operator;
        description
          "Comparison type.";
      }
      container policy-value {
        container policy-argument {
          choice argument {
            case explict-variable {
              leaf explict-variable {
                type leafref {
                  path "/policy-variables/policy-variable/explict-variable";
                }
                description
                  "explict variable";
              }
            }
            case implict-variable {
              leaf implict-variable {
                type leafref {
                  path "/policy-variables/policy-variable/implict-variable";
                }
                description
                  "implict variable";
              }
            }
            case value {
              leaf policy-value {
                type leafref {
                  path "/policy-variables/policy-variable/policy-value";
                }
                description
                  "policy value";
              }
            }
            description
              "Choice one argument format";
          }
          description
            "Cotainer for policy argument";
        }
        description
```

```
          "Container for policy value";
      }
      container policy-variable {
        container policy-argument {
          choice argument {
            case explict-variable {
              leaf explict-variable {
                type leafref {
                  path "/policy-variables/policy-variable/explict-variable";
                }
                description
                  "explict variable";
              }
            }
            case implict-variable {
              leaf implict-variable {
                type leafref {
                  path "/policy-variables/policy-variable/implict-variable";
                }
                description
                  "implict variable";
              }
            }
            description
              "Choice one argument format";
          }
          description
            "Cotainer for policy argument";
        }
        description
          "Container for policy variable";
      }
      description
        "Container for boolean test.";
    }
  }

  grouping threshold-trigger {
    description
      "A grouping that provides threshold trigger";
    container threshold {
      leaf rising-value {
        type leafref {
          path "/policy-variables/policy-variable/policy-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-policy-variable {
        type leafref {
          path "/policy-variables/policy-variable/name";
        }
        description
          "List for target variable.";
      }
      leaf falling-value {
        type leafref {
          path "/policy-variables/policy-variable/policy-value";
        }
        description
          "Sets the falling threshold to the specified value.";
      }
      leaf-list falling-policy-variable {
        type leafref {
          path "/policy-variables/policy-variable/name";
        }
        description
          "List for target variable.";
      }
      leaf delta-rising-value {
        type leafref {
          path "/policy-variables/policy-variable/policy-value";
        }
        description
          "Sets the delta rising threshold to the specified value.";
      }
      leaf-list delta-rising-policy-variable {
        type leafref {
          path "/policy-variables/policy-variable/name";
        }
        description
          "List for target variable.";
      }
      leaf delta-falling-value {
        type leafref {
          path "/policy-variables/policy-variable/policy-value";
        }
        description
          "Sets the delta falling threshold to the specified value.";
      }
      leaf-list delta-falling-policy-variable {
        type leafref {
          path "/policy-variables/policy-variable/name";
```

```
        }
        description
          "List for target variable.";
      }
      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 thrshold 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;
        }
      }
    }

    container policy-variables {
      list policy-variable {
        key "name";
        leaf name {
          type string;
          description
            "Policy variable name";
        }
        leaf type {
          type identityref {
          base variable-type;
          }
          description
            "Policy variable type";
        }
        leaf explict-variable {
          type yang:xpath1.0;
          description
            "Explict policy variable";
        }
        leaf implict-variable {
          type identityref {
            base policy-variable-type;
          }
          description
            "A common policy variable type, defined as an
             identity.";
        }
        leaf policy-value {
          type union {
            type yang:xpath1.0;
            type yang:object-identifier;
            type yang:uuid;
            type string;
            type boolean;
            type int32;
```

```
          type int64;
          type uint32;
          type uint64;
        }
        description
          "Policy value";
      }
      description
        "List for policy variable";
    }
    description
      "Policy variables";
  }
  container eca {
    list event {
      key "name type";
      leaf 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 explict-variable {
        type leafref {
          path "/policy-variables/policy-variable/explict-variable";
        }
        description
          "Explict variable";
      }
      leaf clear {
        type boolean;
        default "false";
```

```
        description
          "A flag indicate whether the event be closed";
      }
      container scheduled-time {
        leaf type {
          type identityref {
            base frequency;
          }
          description
            "Type of scheduled-time";
        }
        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 calendar-time {
          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";
      }
      leaf last-event {
        type leafref {
          path "/eca/event/name";
        }
        config false;
        description
          "The reference to a event last executed
           or being executed.";
      }
      leaf last-condition {
        type leafref {
          path "/eca/event/condition/name";
        }
```

```
          config false;
          description
            "The reference to a condition last executed or being
             executed.";
        }
        leaf last-action {
          type leafref {
            path "/eca/event/actions/action/name";
          }
          config false;
          description
            "The reference to aa action last executed or being
             executed.";
        }
        list condition {
          key "name";
          leaf name {
            type string;
            description
              "Trigger name";
          }
          leaf condition-description {
            type string;
            description
              "Trigger description";
          }
          leaf logical-operation-type {
            type identityref {
              base logical-operation-type;
            }
            description
              "The logical operation type.";
          }
          leaf call-event {
            type leafref {
              path "../../name";
            }
            description
              "This leaf call sub-event.";
          }
          uses trigger-grouping;
          description
            "List for trigger";
        }
        container actions {
          list action {
            key "name";
            leaf name {
```

```
            type string;
            description
              "Action Name";
          }
          choice action-type {
            description
              "Choice one action type";
            case set {
              container set {
                leaf policy-variable {
                  type leafref {
                    path "/policy-variables/policy-variable/name";
                  }
                  description
                    "The target objects";
                }
                anydata value {
                  description
                    "Inline set content.";
                }
                description
                  "Set a value to the target";
              }
            }
            case logging {
              container logging {
                leaf type {
                  type logging-type;
                  description
                    "Specifies the log action";
                }
                leaf policy-variable {
                  type leafref {
                    path "/policy-variables/policy-variable/name";
                  }
                  description
                    "The target objects";
                }
                description
                  "Specifies the log action";
              }
            }
            case function-call {
              container function-call {
                description
                  "The operation is to call a function, which is of one of
                   a few basic predefined types, such as plus, minus,
                   multiply, devide, or remainder.";
```

```
                leaf function-type {
                  type identityref {
                    base function-type;
                  }
                  description
                    "One of the predefined basic function types, such as
                     plus, minus, multiply, devide, or remainder.";
                }
                list policy-argument {
                  key "name";
                  leaf name {
                    type string;
                    description
                      "Policy argument name";
                  }
                  choice argument {
                    case explict-variable {
                      leaf explict-variable {
                        type leafref {
                          path "/policy-variables/policy-variable/explict-
variable";
                        }
                        description
                          "explict variable";
                      }
                    }
                    case implict-variable {
                      leaf implict-variable {
                        type leafref {
                          path "/policy-variables/policy-variable/implict-
variable";
                        }
                        description
                          "implict variable";
                      }
                    }
                    case value {
                      leaf policy-value {
                        type leafref {
                          path "/policy-variables/policy-variable/policy-
value";
                        }
                        description
                          "policy value";
                      }
                    }
                    description
                      "Choice one argument format";
                  }
                  description
                    "List for policy argument";
```

```
                }
                container result {
                  choice argument {
                    case explict-variable {
                      leaf explict-variable {
                        type leafref {
                          path "/policy-variables/policy-variable/explict-
variable";
                        }
                        description
                          "explict variable";
                      }
                    }
                    case implict-variable {
                      leaf implict-variable {
                        type leafref {
                          path "/policy-variables/policy-variable/implict-
variable";
                        }
                        description
                          "implict variable";
                      }
                    }
                    case value {
                      leaf policy-value {
                        type leafref {
                          path "/policy-variables/policy-variable/policy-
value";
                        }
                        description
                          "policy value";
                      }
                    }
                    description
                      "Choice one argument format";
                  }
                  description
                    "Container for result";
                }
              }
            }
            case rpc-call {
              container rpc-call {
                leaf name {
                  type string;
                  description
                    "The name of the YANG RPC or YANG action to be
                     called.";
                }
                container input {
                  list policy-argument {
                    key "name";
                    leaf name {
                      type string;
                      description
                        "Policy argument name";
                    }
                    choice argument {
                      case explict-variable {
                        leaf explict-variable {
                          type leafref {
                            path "/policy-variables/policy-variable/explict-
variable";
                          }
                          description
                            "explict variable";
                        }
                      }
                      case implict-variable {
                        leaf implict-variable {
                          type leafref {
                            path "/policy-variables/policy-variable/implict-
variable";
                          }
                          description
                            "implict variable";
                        }
                      }
                      case value {
                        leaf policy-value {
                          type leafref {
                            path "/policy-variables/policy-variable/policy-
value";
                          }
                          description
                            "policy value";
                        }
                      }
                      description
                        "Choice one argument format";
                    }
                    description
                      "List for policy argument";
                  }
                  description
                    "Container for input";
                }
                container output {
                  list policy-argument {
                    key "name";
                    leaf name {
                      type string;
                      description
                        "Policy argument name";
                    }
                    choice argument {
                      case explicit-variable {
                        leaf explicit-variable {
                          type leafref {
                            path "/policy-variables/policy-variable/explict-
variable";
                          }
                          description
                            "explict variable";
                        }
                      }
                      case implicit-variable {
                        leaf implicit-variable {
                          type leafref {
                            path "/policy-variables/policy-variable/implict-
variable";
                          }
                          description
                            "implict variable";
                        }
                      }
                      case value {
                        leaf policy-value {
                          type leafref {
                            path "/policy-variables/policy-variable/policy-
value";
                          }
                          description
                            "policy value";
                        }
                      }
                      description
                        "Choice one argument format";
                    }
                    description
                      "List for policy argument";
                  }
                  description
                    "Container for output";
                }
                description
                  "Container for rpc call";
              }
            }
          }
          description
            "List for actions";
        }
        description
          "Container for Actions";
      }
      description
        "List for Events";
    }
    description
      "YANG data module for defining event triggers and actions for
       network management purposes";
  }
}
```

<CODE ENDS>

## 7. 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:

o /eca/event/name

o /eca/policy-variables/policy-variable/name

o /eca/event/actions/action/name

o /eca/event/condition/name

## 8.  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:

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

This document registers one YANG module in the YANG Module Names registry [RFC6020].

```
---------------------------------------------------------------------
   Name:         ietf-eca
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-eca
   Prefix:       eca
   Reference:    RFC xxxx
---------------------------------------------------------------------
```

## 9.  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 delegation for material that assisted in thinking that helped improve this document.

## 10.  Objectives for existing and possible future extension

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

o  Clear and precise identification of Event types in the ECA Policy.

o  Clear and precise identification of managed object (i.e., policy variable) in the ECA Policy.

- Allow nested ECA policy,e.g, one event to be able to call another nested event.

- Allow the client use NETCONF/RESTCONF protocol or any other management protocol to configure ECA Policy.

- Allow the server send updates only when the value falls within a certain range.

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

- Allow the client optimize the system behavior across the whole network to meet objectives and provide some performance guarantees for network services.

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

- Allow the ECA execution thread in the server use YANG Push/YANG Push extension to communicate with the client.

## 11. Contributors

Chongfeng Xie
China Telecom
Email: xiechf@ctbri.com.cn

Xiaopeng Qin
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing  100095
China
qinxiaopeng@huawei.com

Alexander Clemm
Futurewei
Email: ludwig@clemm.org

Henk Birkholz
Fraunhofer SIT
Email: henk.birkholz@sit.fraunhofer.de

Tianran Zhou
Huawei
Email: zhoutianran@huawei.com

Aihua Guo
Individual
aihguo1@gmail.com

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

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

[RFC2981] Kavasseri, R., Ed., "Event MIB", RFC 2981, DOI 10.17487/RFC2981, October 2000, <https://www.rfc-editor.org/info/rfc2981>.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6370] Bocci, M., Swallow, G., and E. Gray, "MPLS Transport Profile (MPLS-TP) Identifiers", RFC 6370, DOI 10.17487/RFC6370, September 2011, <https://www.rfc-editor.org/info/rfc6370>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

[RFC8328]  Liu, W., Xie, C., Strassner, J., Karagiannis, G., Klyus, M., Bi, J., Cheng, Y., and D. Zhang, "Policy-Based Management Framework for the Simplified Use of Policy Abstractions (SUPA)", RFC 8328, DOI 10.17487/RFC8328, March 2018, <https://www.rfc-editor.org/info/rfc8328>.

## Appendix A.  ECA Model Usage Example

```
     +---------------------------+
     |     Management System     |
     +---------------------------+
               |
           ECA |
         Model |
               |
               V
    +----------------------^-----+
    |      Managed Device  |     |
    |                      |     |
    |    //--\\ Condition--+     |
    |   | Event|       /    \    |
    |   |      |----->|Actions   |
    |    \\--//        \    /    |
    |                   ----     |
    +----------------------------+
```

For Example:

The management system push down one ECA policy to control interface behavior in the managed device that supports NETCONF protocol operation.

The explicit policy variable of Event "interface-state-monitoring" is set to "/if:interfaces/if:interface[if:name='eth0']", the trigger list contains two conditions: 1)The publisher sends 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:

```
  <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>
     </datastore-changes>
   </push-change-update>
  </notification>

    <eca>
     <event>
     <name>interface-state-monitoring</name>
     <type>interface-exception</type>
     <scheduled-time>
     <periodic>
     <interval>10m</interval>
     </periodic>
     </scheduled-time>
     <condition>
      <name>state-push-change</name>
      <condition-description>sent a yang push
       \changed notification</condition-description>
      <test>
       <existence>
       <policy-variable>/yp:notification/yp:push-change-update/yp:id[id=89]\
      /yp:datastore-changes/.../yp:target="/ietf-interfaces:interfaces='eth0'\
     </policy-variable>
        </existence>
      </test>
     </explict-variable>
     <condition>
      <name>evaluate-in-errors</name>
      <condition-description>evaluate the number of
       the error packets</condition-description>
      <test>
       <boolean>
        <operator>greater-or-equal</operator>
```

```
        <policy-value>
         <policy-argument>
          <policy-value>100</policy-value>
         </policy-argument>
        </policy-value>
        <policy-variable>
         <policy-argument>
         <explict-variable>/if:interfaces/if:interface[if:name='eth0']\
        /if:statistic/if:in-errors</explict-variable>
         </policy-argument>
        </policy-variable>
       </boolean>
      </test>
     </condition>
     <action>
      <name>foo</name>
       <set>
        <policy-variable>/if:interfaces/if:interface[if:name='eth0']</policy-
variable>
        <value>
         <interfaces>
          <interface>
           <name>eth0</name>
           <enable>false</enable>
          </interface>
         </interfaces>
        </value>
       </set>
     </action>
    </event>
   </eca>
```

## Appendix B.  Usage Example of Reusing Trigger-Grouping in smarter filter

The "ietf-eca.yang" module defines a set of groupings for a generic condition expression.  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-eca" module to define the subscription and notification smarter filter.

```
     import ietf-subscribed-notifications {
        prefix sn;
     }
     import ietf-eca {
      prefix eca;
     }

     augment "/sn:subscriptions/sn:subscription" {
       description "add the smart filter container";
       container smart-filter {
          description "It concludes filter configurations";
          uses eca:trigger-grouping;
       }
     }
```

The tree diagrams:

```
   module: ietf-smart-filter
   augment /sn:subscriptions/sn:subscription:
     +--rw smart-filter
       +-- (test)?
          +--:(existences)
          |  +-- existences
          |     +-- type?              enumeration
          |     +-- policy-variable?
          |             -> /policy-variables/policy-variable/name
          +--:(boolean)
          |  +-- boolean
          |     +-- operator?          operator
          |     +-- policy-value
          |     |  +-- policy-argument
          |     |     +-- (argument)?
          |     |        +--:(explict-variable)
          |     |        |  +-- explict-variable?   leafref
          |     |        +--:(implict-variable)
          |     |        |  +-- implict-variable?   leafref
          |     |        +--:(value)
          |     |           +-- policy-value?       leafref
          |     +-- policy-variable
          |        +-- policy-argument
          |           +-- (argument)?
          |              +--:(explict-variable)
          |              |  +-- explict-variable?   leafref
          |              +--:(implict-variable)
          |                 +-- implict-variable?   leafref
          +--:(threshold)
             +-- threshold
                +-- rising-value?                    leafref
                +-- rising-policy-variable*
                |       -> /policy-variables/policy-variable/name
                +-- falling-value?                   leafref
                +-- falling-policy-variable*
                |       -> /policy-variables/policy-variable/name
                +-- delta-rising-value?              leafref
                +-- delta-rising-policy-variable*
                |       -> /policy-variables/policy-variable/name
                +-- delta-falling-value?             leafref
                +-- delta-falling-policy-variable*
                |       -> /policy-variables/policy-variable/name
                +-- startup?                         enumeration
```

## Appendix C.  Changes between Revisions

v05 - v06

- Decouple ECA model from NETCONF protocol and make it applicable to other network mangement protocols.

- Move objective section to the last section with additional generic objectives.

v04 - v05

- Harmonize with draft-bryskin and add additional attributes in the models (e.g., policy variable, func call enhancement, rpc execution);

- ECA conditions part harmonization;

- ECA Event, Condition, Action, Policy Variable and Value definition;

- Change ietf-event.yang into ietf-eca.yang and remove ietf-event-trigger.yang

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 modeland 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  210012
China

Email: wangzitao@huawei.com

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

Email: bill.wu@huawei.com

Igor Bryskin
Individual

Email: i_bryskin@yahoo.com

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

Benoit Claise
Cisco

Email: bclaise@cisco.com