NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2021

A. Bierman
YumaWorks
Q. Wu
Huawei
I. Bryskin
Individual
H. Birkholz
Fraunhofer SIT
X. Liu
Volta Networks
B. Claise
Cisco

July 13, 2020

A YANG Data model for ECA Policy Management
draft-wwx-netmod-event-yang-09

Abstract

This document defines a YANG data model for the Event Condition Action (ECA) policy management. The ECA policy YANG provides the ability to delegate the network management function to the server and 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

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 time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 14, 2021.

Table of Contents

1. Introduction

Traditional approaches for network to automatically perform corrective actions in response to network events have been largely built on centralized policy based management [RFC3198]. With centralized network mangement, the managed object state or operational state spanning across the devices needs to be retrieved by the client from various different servers. However there are issues associated with centrallized network management:

- Centralized network management incurs massive data collection and processing, the resource consumption (e.g., network bandwidth usage, the state to be maintained) is huge.

- Centralized network management leads to slow reaction to the network changes when large amount of managed object state from devices needs to collected and correlated at the central point where decisions about resource adjustment are made;

- Centralized network management can not control or influence management behavior within the server if the server is disconnected from any network or the existing configuration on the server has major errors;

- Centralized network management doesn't scale well when thousands of devices needs to send hundeds of event notifications or millions of managed data objects need to be polled by the client;

A more effective alternative to centralized network management is to delegate network management function to servers in the network and allow each server monitor state changes of managed objects. Accordingly there is a need for a service to provide continuous peformance monitoring and detect defects and failures and take corrective action.

This document defines a ECA Policy management YANG data model. The ECA Policy YANG provides the ability to move the network management task to the server for self monitoring and self healing and 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: An event is something that happens that may be of interest - a configuration change, a fault, a change in status, crossing a threshold, or an external input to the system, for example. Often, this results in an asynchronous message, sometimes referred to as a notification or event notification, being sent to interested parties to notify them that this event has occurred [RFC5277].

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.

ECA Event: The input to the ECA logic that initiates the processing Derived from extensible list of platform event types.

Server Event: An event that happens in the server for which a Notification could be generated in an Event Stream subscription.

Datastore Event: An event that happens within a datastore within the server for a Notification could be generated in a datastore subscription.

Timer Event: A pseudo-event in the server that allows ECA logic to be invoked periodically.

Diagnostic Event: A pseudo-event initiated by the client to test ECA logic.

Self Monitoring: Automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements.

Self Healing: Automatic discovery, and correction of faults; automatically applying all necessary actions to bring system back to normal operation.

Policy Variable (PV): Represents datastore states that change (or "vary"), and that is set or evaluated by software.

PV-Source: Represents an XPath result, which contains one of four data types: Boolean, Number, String, and Node Set.

PV-Result: Represents the value of the result of an Policy Variable evaluation.

2.2. Tree Diagrams

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

3. 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 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.

3.1. ECA Policy Variable and Value

ECA policy variable (PV) generically represents datastore states that change (or "vary"), and that is set or evaluated by software. The value of ECA policy variable 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, pv-source variable and pv-result variable. pv-source variable represents an XPath result, which contains one of four data types: Boolean, Number, String, and Node Set while pv-result variable represents the value of the result of an Policy Variable evaluation.

- A pv-source is always config=true.
- A pv-result is always config=false.
- A single anydata cannot be used for all values since it is only allowed to contain child nodes. Separate scalar and nodeset values are needed.

Each ECA policy variable has the following two attributes:

- Name with Globally unique or ECA unique scope ;
- Type either pv-source or pv-result;

The following operations are allowed with/on a PV:

- initialize (with a constant/enum/identity);
- set (with contents of another same type PV);
- read (retrieve datastore contents pointed by the specified same type XPath/sub-tree);
- write (modify configuration data in the datastore with the PV's content/value);
- insert (PV's content into a same type list);
- iterate (copy into PV one by one same type list elements)

- 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 also be a source of information sent to the client in notification messages.

PVs could be also 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 (xpath-value-choice)?
     |        +--:(policy-source)
     |        |  +--rw (pv-source)
     |        |     +--:(xpath-expr)
     |        |     |  +--rw xpath-expr?         yang:xpath1.0
     |        |     +--:(scalar-constant)
     |        |     |  +--rw scalar-constant?    string
     |        |     +--:(nodeset-constant)
     |        |        +--rw nodeset-constant?   <anydata>
     |        +--:(policy-result)
     |           +--rw (pv-result)
     |              +--:(scalar-value)
     |              |  +--rw scalar-value?       string
     |              +--:(nodeset-value)
     |                 +--rw nodeset-value?      <anydata>
```

3.2. ECA Event

The ECA Event is any subscribable event notification either explicitly defined in a YANG module (e.g., interface management model) supported by the server or a event stream conveyed to the server via YANG Push subscription. 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.

Each ECA Event has the following attributes:

- event-name, the name of ECA event;

- event-type, typical examples of ECA event type include server event, datastore event, timer event and diagnostic event.

- event-stream,in case of server event.

- event-module, in case of server event.
- event-name, in case of server event.
- event, it is event stream conveyed to the server in case of server event.
- datastore, in case of datastore event.
- data-path, in case of datastore event.
- data, it is event notification defined in a YANG module, in case of datastore event.
- period, in case of timer event.

A client may define an event of interest by making use of YANG PUSH subscription. Specifically, the client may configure an ECA event according to the ECA model specifying the event's name, as well as the name of corresponding PUSH subscrition. In this case, the server is expected to:

- Register the event recording its name and using the referred PUSH subsription trigger as definition of the event firing trigger;
- Auto-configure the event's ECA input in the form of local PVs using the PUSH subscription's filters;
- At the moment of event firing intercept the notifications that would be normally sent to the PUSH subscription's client(s); copy the data store states pointed by the PUSH subscription's filters into the auto-configured ECA's local PVs and execute the ECA's condition-action chain.

All events (specified in at least one ECA pushed to the server) are required to be constantly monitored by the server. One way to think of this is that the server subscribes to its own publications with respect to all events that are associated with at least one ECA.

The model structure for the ECA Event is shown below:

```
     +--rw events
     |  +--rw event* [event-name]
     |     +--rw event-name               string
     |     +--rw event-type?              identityref
     |     +--rw policy-variable*         -> /gncd/policy-variables/policy-
variable/name
     |     +--rw local-policy-variable*   -> /gncd/ecas/eca/policy-variable/
name
     |     +--rw (type-choice)?
     |        +--:(server-event)
     |        |  +--rw event-stream?      string
     |        |  +--rw event-module?      string
     |        |  +--rw event?             <anydata>
     |        +--:(datastore-event)
     |        |  +--rw datatore?          string
     |        |  +--rw data-path?         string
     |        |  +--rw data?              <anydata>
     |        +--:(timer-event)
     |        |  +--rw time-schedule!
     |        |     +--rw period?   centiseconds
     |        |     +--rw count?    uint16
     |        +--:(diagnostics-event)
```

3.3. ECA Condition

The ECA Condition is the logical expression that is specified in a form of Xpath expression and evaluated to TRUE or FALSE. The XPath expression specifies an arbitrary logical/mathematical expression; The elements of the ECA Condition expression are referred by the XPaths pointing to referred datastore states.

The ECA Condition expression in the form of XPath expression allows for specifying a condition of arbitrary complexity as a single string with an XPath expression, in which pertinent PVs and datastore states are referred to by their respective positions in the YANG tree.

ECA Conditions are associated with ECA Events and evaluated only within event threads triggered by the event detection.

When an ECA Condition is evaluated to TRUE, the associated ECA Action is executed.

The model structure for the condition is shown below:

```
     +--rw conditions
     |  +--rw condition* [name]
     |     +--rw name                     string
     |     +--rw (expression-choice)?
```

```
        |        +--:(xpath)
        |           +--rw condition-xpath?   string
```

3.3.1. Mapping Policy Variables to XPath Variables

Policy variables are mapped to XPath variable bindings so they can be referenced in the XPath expression for a Condition.

- The 'name' leaf value for the policy variable is mapped to the local-name of the XPath variable. No namespace is used for ECA varaibles. Eg., the policy variable named 'foo' would be accessible with a variable refernece '$foo'.

- The local-name 'USER' is reserved and defined in NACM. The server SHOULD provide the USER variable as NACM is implemented.

- The values of all available policy variables are updated by the server (if required) before the XPath expression is evaluated. The variable binding value MUST NOT change while the XPath expression is being evaluated. If multiple references to the same variable exist in an XPath expression, they MUST resolve to the same value in each instance.

Example: "/test1[name=$badfan] and /test2[name=$badfan]"
The same value of 'badfan' is expected in each instance.

- If a variable reference cannot be resolved because no policy variable with that name is accessible to the ECA under evaluation, then an eca-exception notification SHOULD be generated, and the XPath evaluation MUST be terminated with an error.

- Example:

```
[TBD: Need to determine what XPath parsers support.
 Need to support simple expressions like
    PV(x) = $A
    PV(x) = $A + $B

May need to wrapper in function calls

    PV(x) = number($A)
    PV(x) = number($A) + number($B)

TBD: How to do conditional assignments

   if nmda-supported()
     PV(top) = /interfaces
   else
     PV(top) = /interfaces-state
   end

Then an XPath expression can use

   $top/interface/name
```

3.3.2. ECA XPath Context

All XPath expressions used in ECA share the following XPath context definition.

- The set of namespace declarations is the set of all modules loaded into the server at the moment. Prefix bindings can reference the set of namespace URIs for this set of modules.

- All names SHOULD be namespace-qualified. There is no default namespace to use if no namespace is specified. If no namespace is used then the XPath step matches the local-name in all namespaces.

- The function library is the core function library defined in [XPATH], the functions defined in Section 10 of [RFC7950], and the ECALIB functions defined in this document Section 3.5.1.

- The set of variable bindings is set to all policy variables that are visible to the ECA under evaluation. This includes the local-policy-variable and policy-variable entries configured for the 'eca' entry. Since pv-source values can reference other policy variables, the order that these fields are set is significant.

- The accessible tree is all state data in the server, and the running configuration datastore. The root node has all top-level data nodes in all modules as children.

- The context node for all ECA XPath evaluation is the root node.

3.3.3. ECA Evaluation Exceptions

Not all errors can be detected at configuration time. Error that occur while ECA logis is being evaluated will cause the server to generate an eca-exception notification.

TBD: Does an exception cause the ECA entry to be disabled automatically?

```
identity eca-exception-reason {
  description
    "Base of all values for the 'reason' leaf in the
    eca-exception notification.";
}

identity varbind-unknown {
  base eca-exception-reason;
  description
    "The requested policy variable binding is not defined.
     The variable binding cannot be resolved in the XPath
     evaluation.";
}

// TBD: define exceptions as needed

 notification eca-exception {
   description
     "This notification is sent when some error occurs
      while the server is processing ECA logic.
      [TBD: lots more detail and parameters]";
    leaf reason {
      type eca-exception-reason;
    }
 }
```

3.4. ECA Action

The ECA 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:

- sending one time notification

- (re-)configuration scheduling - scheduling one time or periodic (re-)configuration in the future

- stopping current ECA;
- invoking another ECA;

Three points are worth noting:

- When a "Send notification" action is configured as an ECA Action, the notification message to be sent to the client may contain not only elements of the data store (as, for example, YANG PUSH or smart filter notifications do), but also the contents of global and local PVs, which store results of arbitrary operations performed on the data store contents (possibly over arbitrary period of time) to determine, for example, history/evolution of data store changes, median values, ranges and rates of the changes, results of configured function calls and expressions, etc. - in short, any data the client may find interesting about the associated event with all the logic to compute said data delegated to the server. Importantly, ECA notifications are the only ECA actions that directly interact with and hence need to be unambiguously understood by the client. Furthermore, the same ECA may originate numerous single or repetitive semantically different notifications within the same or separate event firings. In order to facilitate for the client the correlation of events and ECA notifications received from the server, the ECA model requires each notification to carry mandatory information, such as event and (event scope unique) notification names.
- 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-element* [name]
        |     |  +--rw name                       string
        |     |  +--rw action-type?               identityref
        |     |  +--rw (action-operation)?
        |     |     +--:(notify-operation)
        |     |        +--rw notify-operation
        |     |           +--rw name?               string
        |     |           +--rw policy-variable* [name]
        |     |              +--rw name    string
        |     +--rw time-schedule!
        |     |  +--rw period?   centiseconds
        |     |  +--rw count?    uint16
```

3.5. ECA

An ECA container includes:

- ECA name.
- List of local PVs and global PVs. As mentioned, These PVs could be configured as dynamic (their instances appear/disappear with start/stop of the ECA execution) or as static (their instances exist as long as the ECA is configured). Global PV will be shared by multiple ECA instances while local PVs are within the scope of a specific ECA instance.
- Normal CONDITION-ACTION list: configured conditions each with associated actions to be executed if the condition is evaluated to TRUE

TBD: how different ECAs do not impact each other if they share PVs and other components is not in the scope of this document at this moment.

3.5.1. ECA XPath Function Library (ECALIB)

A set of common event PVs need to be set for every invocation of condition or action logic:

```
   $event-type      (string)
   $event-name    (string)

For event-type = "server-event"

  $event-stream   (string)
  $event-module   (string)
  $event-name     (string)
  $event          (node-set)

The condition can use these PVs directly in an expression
An expression can access client-configured PVs of course

   $event/child[name=$some-global-var] > 10

For event-type = "datastore"

  $datastore  (string)
  $data-path  (string)
  $data       (node-set)

The data is defined to be a container with the requested data as child nodes

  $data/interface[type=$gigabit-eth] // (node-set is an array of data nodes,
usually siblings)

A standard sustained-event func call should be defined to specify how many
seconds the
 XPath expression needs to be true to consider the function result true

// check every 5 seconds up to 60 seconds
  sustained-event("$event/child[name=$some-global-var] > 10", 5, 12)

function boolean sustained-event (string expr, number interval, number count)
        test expression 'expr' once per 'interval'. Keep testing once per
interval until
        false result reached or 'count' number of interval on specific
interface has been
        tested true.Return true if condition tested true for count intervals;
Returns
        false otherwise
```

The ECA XPath function library is expected to grow over time and additional standard or vendor function libraries should be possible. The server should provide a read-only list of ECA function libraries supported.

```
        +--rw eca-func-libs
           +--rw eca-function* [func-name]
```

```
            +--rw func-name    string
            +--rw eca* [eca-name]
               +--rw eca-name    -> /gncd/ecas/eca/name
```

TBD: How can ECA access specific datastores? Currently no NMDA support for config=true values in <operational> is provided. Access to <candidate> datastore is not possible.

4. ECA YANG Model (Tree Structure)

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

```
module: ietf-eca
  +--rw gncd
     +--rw policy-variables
     |  +--rw policy-variable* [name]
     |     +--rw name                              string
     |     +--rw (xpath-value-choice)?
     |        +--:(policy-source)
     |        |  +--rw (pv-source)
     |        |     +--:(xpath-expr)
     |        |     |  +--rw xpath-expr?          yang:xpath1.0
     |        |     +--:(scalar-constant)
     |        |     |  +--rw scalar-constant?     string
     |        |     +--:(nodeset-constant)
     |        |        +--rw nodeset-constant?    <anydata>
     |        +--:(policy-result)
     |           +--rw (pv-result)
     |              +--:(scalar-value)
     |              |  +--rw scalar-value?        string
     |              +--:(nodeset-value)
     |                 +--rw nodeset-value?       <anydata>
     +--rw events
     |  +--rw event* [event-name]
     |     +--rw event-name               string
     |     +--rw event-type?              identityref
     |     +--rw policy-variable*         -> /gncd/policy-variables/policy-
variable/name
     |     +--rw local-policy-variable*   -> /gncd/ecas/eca/policy-variable/
name
     |     +--rw (type-choice)?
     |        +--:(server-event)
     |        |  +--rw event-stream?      string
     |        |  +--rw event-module?      string
     |        |  +--rw event?             <anydata>
     |        +--:(datastore-event)
     |        |  +--rw datatore?          string
     |        |  +--rw data-path?         string
     |        |  +--rw data?              <anydata>
     |        +--:(timer-event)
     |        |  +--rw time-schedule!
```

```
     |        |     +--rw period?   centiseconds
     |        |     +--rw count?    uint16
```

```
|        +--:(diagnostics-event)
+--rw conditions
|  +--rw condition* [name]
|     +--rw name                      string
|     +--rw (expression-choice)?
|        +--:(xpath)
|           +--rw condition-xpath?   string
+--rw actions
|  +--rw action* [name]
|     +--rw name              string
|     +--rw action-element* [name]
|     |  +--rw name                       string
|     |  +--rw action-type?               identityref
|     |  +--rw (action-operation)?
|     |     +--:(notify-operation)
|     |        +--rw notify-operation
|     |           +--rw name?               string
|     |           +--rw policy-variable* [name]
|     |              +--rw name    string
|     +--rw time-schedule!
|        +--rw period?   centiseconds
|        +--rw count?    uint16
+--rw ecas
|  +--rw eca* [name]
|     +--rw name                string
|     +--rw username            string
|     +--rw event-name          string
|     +--rw policy-variable* [name]
|     |  +--rw name                              string
|     |  +--rw (xpath-value-choice)?
|     |  |  +--:(policy-source)
|     |  |  |  +--rw (pv-source)
|     |  |  |     +--:(xpath-expr)
|     |  |  |     |  +--rw xpath-expr?         yang:xpath1.0
|     |  |  |     +--:(scalar-constant)
|     |  |  |     |  +--rw scalar-constant?    string
|     |  |  |     +--:(nodeset-constant)
|     |  |  |        +--rw nodeset-constant?   <anydata>
|     |  |  +--:(policy-result)
|     |  |     +--rw (pv-result)
|     |  |        +--:(scalar-value)
|     |  |        |  +--rw scalar-value?       string
|     |  |        +--:(nodeset-value)
|     |  |           +--rw nodeset-value?     <anydata>
|     |  +--rw is-static?                      boolean
|     +--rw condition-action* [name]
|     |  +--rw name         string
|     |  +--rw condition?   -> /gncd/conditions/condition/name
```

```
     |     |  +--rw action?      -> /gncd/actions/action/name
     |     +---x start
     |     +---x stop
     |     +---x next-action
     +--rw eca-func-libs
        +--rw eca-function* [func-name]
           +--rw func-name    string
           +--rw eca* [eca-name]
              +--rw eca-name    -> /gncd/ecas/eca/name

  notifications:
    +---n eca-exception
       +--ro reason?   identityref
```

5. 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 gnca;

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC8341: Network Configuration Access Control Model";
  }

  organization
    "IETF Network Configuration (NETCONF) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>
     Editor:   Qin Wu
               <mailto:bill.wu@huawei.com>
     Editor:   Igor Bryskin
               <mailto:Igor.Bryskin@huawei.com>
     Editor:   Henk Birkholz
               <mailto:henk.birkholz@sit.fraunhofer.de>
     Editor:   Xufeng Liu
               <mailto:xufeng.liu.ietf@gmail.com>
     Editor:   Benoit Claise
               <mailto:bclaise@cisco.com>
     Editor:   Andy Bierman
```

```
              <mailto:andy@yumaworks.com>
     Editor:  Alexander Clemm
              <mailto:ludwig@clemm.org>";
  description
    "Event Condition Action (ECA) model.";

  revision 2018-06-22 {
    description
      "Initial revision";
    reference
      "RFC XXXX";
  }

  identity argument-type {
    description
      "Possible values are:
       constant, variable, or datastore state.";
  }

  identity comparison-type {
    description
      "Possible values are:
       equal, not-equal, greater, greater-equal, less, less-equal.";
  }

  identity logical-operation-type {
    description
      "Possible values are:
       not, or, and.";
  }

  identity function-type {
    description
      "Possible values are:
       plus, minus, mult, divide, sustained-event.";
  }

  identity sustained-event {
    description
      "Identity for standard sustained-event function call,
       the input variables for sustained-event include string
       expr, number interval, number count. Keep testing
       expression 'expr'once per interval until false result
       reached or 'count' number of interval on specific interface
       has been tested true.Return true if condition tested true
       for count intervals; Returns false otherwise.";
  }
```

```
  identity plus {
    description
      "Identity for standard plus function call, the input
       variables for plus function call include src policy argument
       and dst policy arugment.";
  }

  identity mius {
    description
      "Identity for standard minus function call, the input
       variables for plus function call include src policy argument
       and dst policy arugment.";
  }

  identity multiply {
    description
      "Identity for standard multiply function call, the input
       variables for multiply function call include src policy argument
       and dst policy arugment.";
  }

  identity divide {
    description
      "Identity for standard divide function call, the input
       variables for multiply function call include src policy argument
       and dst policy arugment.";
  }

  identity content-moving-operation-type {
    description
      "Possible values are:
       copy, iterate, insert.";
  }

  identity action-type {
    description
      "Possible values are:
       action, content-move, function-call, rpc, notify.";
  }

  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 server-event {
    base event-type;
    description
      "Identity for server event.";
  }

  identity datastore-event {
    base event-type;
    description
      "Identity for datastore event.";
  }

  identity timer-event {
    base event-type;
    description
      "Identity for timer event.";
  }

  identity diagnostics-event {
    base event-type;
    description
      "Identity for diagnostics event.";
  }

  identity eca-exception-reason {
    description
      "Base of all values for the 'reason' leaf in the
       eca-exception notification.";
  }

  identity varbind-unknown {
    base eca-exception-reason;
    description
      "The requested policy variable binding is not defined.
       The variable binding cannot be resolved in the XPath
       evaluation.";
  }

  typedef centiseconds {
    type uint32;
    description
      "A period of time, measured in units of 0.01 seconds.";
  }

  typedef oper-status {
```

```
    type enumeration {
      enum completed {
        description
          "Completed with no error.";
      }
      enum running {
        description
          "Currently with no error.";
      }
      enum sleeping {
        description
          "Sleeping because of time schedule.";
      }
      enum stoped {
        description
          "Stopped by the operator.";
      }
      enum failed {
        description
          "Failed with errors.";
      }
      enum error-handling {
        description
          "Asking the operator to handle an error.";
      }
    }
    description
      "The operational status of an ECA execution.";
  }

  grouping scalar-value {
    leaf scalar-value {
      type string;
      description
        "Represents an XPath simple value that has an
         XPath type of Boolean, String, or Number.
         This value will be converted to an XPath type,
         as needed.

         A YANG value is encoded as a string using the same
         rules as the 'default' value for the data type.

         An eca-exception notification is generated if a scalar
         XPath value is used in a path expression, where a
         node-set is expected. Normally XPath will treat this result
         as an empty node-set, but this is an ECA programming error.";
    }
  }
```

```
  grouping nodeset-value {
    anydata nodeset-value {
      description
        "Represents an XPath node set. A 'node-set' anydata node
         with no child data nodes represents an empty node-set.
         Each child node in within this anydata structure
         represents a subtree that is present in the XPath
         node-set.

         An XPath node-set is not required to contain a top-level
         YANG data node.  It is not required to contain an entire
         complete subtree.

         It is am implementation-specific manner how a
         representation of YANG 'anydata' nodes are mapped
         to specific YANG module schema definitions.";
    }
  }

  grouping scalar-constant {
    leaf scalar-constant {
      type string;
      description
        "Represents an XPath simple value that has an
         XPath type of Boolean, String, or Number.
         This value will be converted to an XPath type,
         as needed.

         A YANG value is encoded as a string using the same
         rules as the 'default' value for the data type.

         An eca-exception notification is generated if a scalar
         XPath value is used in a path expression, where a
         node-set is expected. Normally XPath will treat this result
         as an empty node-set, but this is an ECA programming error.";
    }
  }

  grouping nodeset-constant {
    anydata nodeset-constant {
      description
        "Represents an XPath node set. A 'node-set' anydata node
         with no child data nodes represents an empty node-set.
         Each child node in within this anydata structure
         represents a subtree that is present in the XPath
         node-set.

         An XPath node-set is not required to contain a top-level
```

```
        YANG data node.  It is not required to contain an entire
        complete subtree.

        It is am implementation-specific manner how a
        representation of YANG 'anydata' nodes are mapped
        to specific YANG module schema definitions.";
    }
  }

  grouping pv-source {
    choice pv-source {
      mandatory true;
      description
        "A PV source represents an XPath result, which contains
         one of four data types: Boolean, Number, String,
         and Node Set.  XPath defines mechanisms to covert
         values between these four types.

         The 'xpath-expr' leaf is used to assign the PV source
         to the result of an arbitrary XPath expression.
         The result of this expression evaluation is used
         internally as needed. The result may be any one of
         the XPath data types.

         The 'scalar-constant' leaf is used to represent a Boolean,
         String, or Number XPath constant value.

         The 'nodeset-constant' anydata structure is used to
         represent a constant XPath node-set.";
      leaf xpath-expr {
        type yang:xpath1.0;
        description
          "Contains an XPath expression that must be evaluated
           to produce an XPath value. [section X.X] describes
           the XPath execution environment used to process this
           object.";
      }
      case scalar-constant {
        uses scalar-constant;
      }
      case nodeset-constant {
        uses nodeset-constant;
      }
    }
  }

  grouping pv-result {
    choice pv-result {
```

```
      mandatory true;
      description
        "Represents the value of the result of an
         Policy Variable evaluation.

         The 'scalar-value' leaf is used to represent a Boolean,
         String, or Number XPath result value.

         The 'nodeset-value' anydata structure is used to represent
         an XPath node-set result.";
      case scalar-value {
        uses scalar-value;
      }
      case nodeset-value {
        uses nodeset-value;
      }
    }
  }

  grouping policy-variable-attributes {
    description
      "Defining the policy variable attributes, including name, type
       and value. These attributes are used as part of the Policy
       Variable (PV) definition.";
    leaf name {
      type string;
      description
        "A string to uniquely identify a Policy Variable (PV), either
         globally for a global PV, or within the soope of ECA for a
         local PV.";
    }
    choice xpath-value-choice {
      description
        "The type of a policy variable may be either a common
         primative type like boolean or a type from existing
         schema node referenced by an XPath string.";
      case policy-source {
        uses pv-source;
      }
      case policy-result {
        uses pv-result;
      }
    }
  }

  grouping action-element-attributes {
    description
      "Grouping of action element attributes.";
```

```
    leaf action-type {
      type identityref {
        base action-type;
      }
      description
        "Identifies the action type.";
    }
    choice action-operation {
      description
        "The operation choices that an ECA Action can take.";
      case notify-operation {
        container notify-operation {
          description
            "The operation is to send a YANG notification.";
          leaf name {
            type string;
            description
              "Name of the subscribed YANG notification.";
          }
          list policy-variable {
            key "name";
            description
              "A list of policy arguments carried in the notification
               message.";
            leaf name {
              type string;
              description
                "A string name used as the list key to form a list
                 of policy arguments.";
            }
          }
        }
      }
    }
  }

  grouping time-schedule-container {
    description
      "Grouping to define a container of a time schedule.";
    container time-schedule {
      presence "Presence indicates that the timer is enabled.";
      description
        "Specifying the time schedule to execute an ECA Action, or
         trigger an event.";
      leaf period {
        type centiseconds;
        description
          "Duration of time that should occur between periodic
```

```
           push updates, in units of 0.01 seconds.";
      }
      leaf count {
        type uint16;
        description
          "specify the count number of interval that has to pass before
           successive adaptive periodic push update records for the same
           subscription are generated for a receiver.";
      }
    }
  }

  container gncd {
    nacm:default-deny-all;
    description
      "Top level container for Generalized Network Control Automation
       (gncd).";
    container policy-variables {
      description
        "Container of global Policy Variables (PVs).";
      list policy-variable {
        key "name";
        description
          "A list of global Policy Variables (PVs), with a string
           name as the entry key.";
        uses policy-variable-attributes;
      }
    }
    container events {
      description
        "Container of ECA events.";
      list event {
        key "event-name";
        description
          "A list of events used as the triggers of ECAs.";
        leaf event-name {
          type string;
          description
            "The name of the event.";
        }
        leaf event-type {
          type identityref {
            base event-type;
          }
          description
            "The type of the event.";
        }
        leaf-list policy-variable {
```

```
          type leafref {
            path "/gncd/policy-variables/policy-variable/name";
          }
          description
            "global policy variables, which
             are shared by all ECA scripts.";
        }
        leaf-list local-policy-variable {
          type leafref {
            path "/gncd/ecas/eca/policy-variable/name";
          }
          description
            "local policy variables, which
             are kept within an ECA instance, and appears/
             disappears with start/stop of the ECA execution.";
        }
        choice type-choice {
          description
            "The type of an event, including server event and datastore
event.";
          case server-event {
            leaf event-stream {
              type string;
              description
                "The name of a subscribed stream .";
            }
            leaf event-module {
              type string;
              description
                "The name of YANG data module associated with the subscribed
                 stream.";
            }
            anydata event {
              description
                "This anydata value MUST Contain the absolute XPath
                 expression identifying the element path to the node that is
                 associated with subscribed stream.";
            }
          }
          case datastore-event {
            leaf datatore {
              type string;
              description
                "The name of a datatore from which applications
                 subscribe to updates.";
            }
            leaf data-path {
              type string;
```

description

```
                "The absolute XPath expression identifying the
                 element path to the node that is associated with
                 subscribed stream..";
              }
              anydata data {
                description
                  "This anydata value MUST Contain the node that is
                              associated with the data path.";
              }
            }
            case timer-event {
              uses time-schedule-container {
                description
                  "Specifying the time schedule to trigger the event.
                   If not specified, the event is not triggered.";
              }
            }
            case diagnostics-event;
          }
        }
      }
      container conditions {
        description
          "Container of ECA Conditions.";
        list condition {
          key "name";
          description
            "A list of ECA Conditions.";
          leaf name {
            type string;
            description
              "A string name to uniquely identify an ECA Condition
               globally.";
          }
          choice expression-choice {
            description
              "The choices of expression format to specify a condition,
               which can be either a XPath string.";
            case xpath {
              leaf condition-xpath {
                type string;
                description
                  "A XPath string, representing a logical expression,
                   which can contain comparisons of datastore values
                   and logical operations in the XPath format.";
              }
            }
          }
```

```
      }
    }
    container actions {
      description
        "Container of ECA Actions.";
      list action {
        key "name";
        description
          "A list of ECA Actions.";
        leaf name {
          type string;
          description
            "A string name to uniquely identify an ECA Action
             globally.";
        }
        list action-element {
          key "name";
          description
            "A list of elements contained in an ECA Action. ";
          leaf name {
            type string;
            description
              "A string name to uniquely identify the action element
               within the scope of an ECA action.";
          }
          uses action-element-attributes;
        }
        uses time-schedule-container {
          description
            "Specifying the time schedule to execute this ECA
             Action.
             If not specified, the ECA Action is executed immediately
             when it is called.";
        }
      }
    }
    container ecas {
      description
        "Container of ECAs.";
      list eca {
        key "name";
        description
          "A list of ECAs";
        leaf name {
          type string;
          description
            "A string name to uniquely identify an ECA globally.";
        }
```

```
        leaf username {
          type string;
          mandatory true;
          description
            "Name of the user for the session.";
        }
        leaf event-name {
          type string;
          mandatory true;
          description
            "The name of an event that triggers the execution of
             this ECA.";
        }
        list policy-variable {
          key "name";
          description
            "A list of ECA local Policy Variables (PVs), with a
             string name as the entry key.";
          uses policy-variable-attributes;
          leaf is-static {
            type boolean;
            description
              "'true' if the PV is static; 'false' if the PV is
               dynamic.
               A dynamic PV appears/disappears with the start/stop
               of the ECA execution; a static PV exists as long as
               the ECA is configured.";
          }
        }
        list condition-action {
          key "name";
          ordered-by user;
          description
            "A list of Condition-Actions, which are configured
             conditions each with associated actions to be executed
             if the condition is evaluated to TRUE.
             [TBD Does the server do all the actions where the condition
             is true? Does it stop after one condition-action is completed?
             How is it possible to require multiple conditions to be true
             in order to do 1 action? How will conditions be reusable
             and not giant cut-and-paste combination of other entries?]";
          leaf name {
            type string;
            description
              "A string name uniquely identify a Condition-Action
               within this ECA.";
          }
          leaf condition {
```

```
            type leafref {
              path "/gncd/conditions/condition/name";
            }
            description
              "The reference to a configured condition.";
          }
          leaf action {
            type leafref {
              path "/gncd/actions/action/name";
            }
            description
              "The reference to a configured action.";
          }
        }
        action start {
          description
            "Start to execute this ECA.";
        }
        action stop {
          description
            "Stop the execution of this ECA.";
        }
        action next-action {
          description
            "Resume the execution of this ECA to complete the next
             action.";
        }
      }
    }
    container eca-func-libs {
      description
        "Container of ECA Function Libraries.";
      list eca-function {
        key "func-name";
        description
          "A list of ECA standard function.";
        leaf func-name {
          type string;
          description
            "A string name to uniquely identify an ECA standard function.";
        }
        list eca {
          key "eca-name";
          description
            "A list of ECAs contained in this ECA function libraries.";
          leaf eca-name {
            type leafref {
              path "/gncd/ecas/eca/name";
```

```
            }
            description
              "The reference to a configured ECA.";
          }
        }
      }
    }
  }

  notification eca-exception {
    description
      "This notification is sent when some error occurs
       while the server is processing ECA logic.
       [TBD: lots more detail and parameters]";
    leaf reason {
      type identityref {
        base eca-exception-reason;
      }
    }
  }
}
```

<CODE ENDS>

6. 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:

- /gnca:gncd/gnca:policy-variables/gnca:policy-variable/gnca:name
- /gnca:gncd/gnca:events/gnca:event/gnca:name
- /gnca:gncd/gnca:conditions/gnca:condition/gnca:name
- /gnca:gncd/gnca:actions/gnca:action/gnca:name
- /gnca:gncd/gnca:ecas/gnca:eca/gnca:name
- /gnca:gncd/gnca:ecas/gnca:eca/gnca:username
- /gnca:gncd/gnca:eca-func-libs/gnca:eca-function/gnca:func-name

7. 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:       gnca
   Reference:    RFC xxxx
---------------------------------------------------------------------
```

8. Acknowledges

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.

9. Contributors

Alexander Clemm
Futurewei
Email: ludwig@clemm.org

Michale Wang
Huawei
Email:wangzitao@huawei.com

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

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

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

10. References

10.1. Normative References

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

[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>.

[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>.

10.2. Informative References

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. ECA Condition Expression Examples

Here are two examples of Condition Expression:

(a) a condition that only includes data store states and constants, for example:

TE metric of Link L in Topology T greater than 100, it can be expressed as follows:

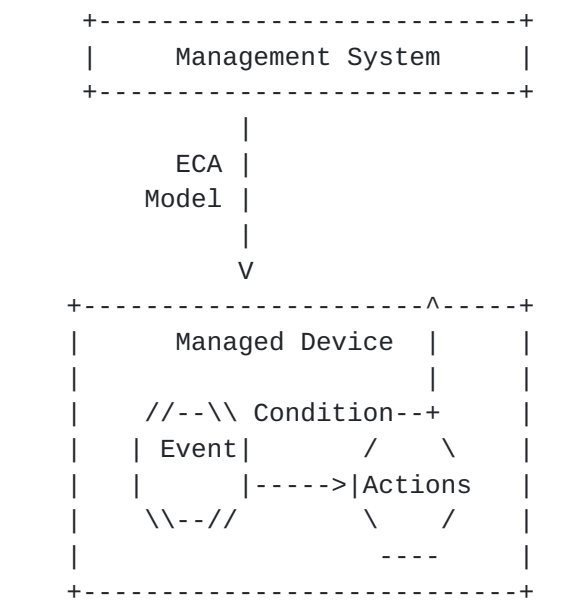
```
"/nw:networks/nw:network[network-id='T']/nt:link[link-id='L']/tet:te\
/tet:te-link-attributes/tet:te-delay-metric > 100"
```

(b) a condition that also includes a Policy Variable, for example:

Allocated bandwidth of Link L in Topology T greater than 75% of what is stored in Policy Variable B, it can be expressed as follows:

```
"/nw:networks/nw:network[network-id='T']/nt:link[link-id='L']/tet:te\
/tet:te-link-attributes/tet:max-resv-link-bandwidth\
> (ietf-eca:policy-variables/policy-variable[name='B']/value) * 0.75"
```

Appendix B. ECA Model Self Monitoring Usage Example



The management system designs a new ECA policy based on monitored objects in ietf-interfaces module that support threshold checking and pushes down the ECA policy to control interface behavior in the managed device that supports NETCONF/RESTCONF protocol operation, i.e.,scan all interfaces for a certain type every 5 seconds up to 60 seconds and check the counters or status, return an array of interface entries (XPath node-set) that match the search. The XML example snippet is shown as below:

```
<gnca>
 <policy-variables>
  <policy-variable>
    <name>event-name</name>
    <scalar-constant>interface-self-monitoring</scalar-constant>
  </policy-variable>
  <policy-variable>
    <name>event-type</name>
    <scalar-constant>server-event</scalar-constant>
  </policy-variable>
  <policy-variable>
    <name>event-stream</name>
    <scalar-constant>NETCONF</scalar-constant>
  </policy-variable>
  <policy-variable>
    <name>event-module</name>
    <scalar-constant>ietf-interfacs</scalar-constant>
  </policy-variable>
  <policy-variable>
    <name>event</name>
    <xpath-expr>if:interfaces/if:interface[if:type=if:gigabitEthernet]</xpath-
```

expr>

```
  </policy-variable>
 </policy-variables>
 <events>
    <event>
     <event-name>interface-self-monitoring</name>
     <event-type>server-event</event-type>
     <event-stream>NETCONF</event-stream>
     <event-module>ietf-interfacs</event-module>
     <event>if:interfaces/if:interface[if:type=if:gigabitEthernet]</event>
   </event>
 </events>
 <conditions>
  <condition>
    <name>if-monitoring-condition</name>
    <condition-xpath>event/statistics/in-errors > 1000 </condition-xpath>
  </condition>
 </conditions>
 <actions>
  <action>
    <name>if-matched-statistics</name>
    <action-element>
    </action-element>
    <time-schedule>
      <period>5</period>
      <count>12</count>
    </time-schedule>
  </action>
 </actions>
 <ecas>
  <eca>
    <name>interface-eca-handling</name>
    <event-name>interface-self-monitoring</event-name>
    <condition-action>
     <name>sustained-event</name>
     <condition>if-monitoring-condition</condition>
     <action>if-matched-statistics</action>
   </condition-action>
 </eca>
 </ecas>
<eca-func-libs>
<eca-function>
<func-name>sustained-event</func-name>
<eca>
<eca-name>interface-eca-handling</eca-name>
</eca>
</eca-function>
</eca-func-libs>
</gnca>
```

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-11-21T13:51:00Z</eventTime>
  <eca-report>
          <interfaces
           xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
           xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
         <interface>
           <name>GE0</name>
           <type>ianaift:gigabitEthernet</type>
           <enabled>false</enabled>
         </interface>

         <interface>
           <name>GE1</name>
           <type>ianaift:gigabitEthernet</type>
           <enabled>true</enabled>
         </interface>

         <interface>
           <name>GE2</name>
           <type>ianaift:gigabitEthernet</type>
           <enabled>true</enabled>
         </interface>
  </eca-report>
</notification>

<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-11-21T13:53:00Z</eventTime>
  <eca-execution>
  <oper-status>completed</oper-status>
     <event-name>interface-self-monitoring</name>
     <event-type>server-event</event-type>
     <event-stream>NETCONF</event-stream>
     <event-module>ietf-interfacs</event-module>
     <period>5</period>
     <count>12</count>
  </eca-execution>
</notification>
```

In this example, the event name is set to 'interface-self-monitoring', the event type is set to 'server-event', the function name of ECA function libraries is set to 'sustained-event', the name of 'condition-action' is corresponding to standard function call 'sustained-event'.

Appendix C. Changes between Revisions

v08 - v09

- Add ECA function libraries list in the ECA model.
- Subtree and data node path fixing in the security section.

v07 - v08

Replace ECA model usage example with self monitoring usage example in the appendix.

Clean up references.

Add a new section to discuss Mapping Policy Variables to XPath Variables.

Add a new section to discuss ECA XPath Context.

Add a new section to discuss ECA Evaluation Exceptions.

Rewrite Introduction to highlight elevator pitch.

Replace implicit variable and explicit variable with pv-source variable and pv-result variable.

Take out function-call, cleanup-condition-action list, execution list, policy argument container, eca-script list at this moment.

v06 - v07

- Reuse alarm notification event received on an event stream (RFC 8639) in ECA logic;
- Represent ECA condition expression only in the form of Xpath expression;
- Add ECA condition expression example in the appendix;
- Add ECA model usage example in the appendix;
- Remove the section to discuss the relation with YANG push;
- Remove the dependency to SUPA framework draft;
- Remove smart filter extension example in the Appendix.

- Bind ECA script with condition expression in the model.

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

Andy Bierman
YumaWorks

Email: andy@yumaworks.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

Henk Birkholz
Fraunhofer SIT

Email: henk.birkholz@sit.fraunhofer.de

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

Benoit Claise
Cisco

Email: bclaise@cisco.com