Generalized Network Control Automation YANG Model

draft-bryskin-netconf-automation-yang-01

Igor Bryskin (Huawei Technologies)
Xufeng Liu (Jabil)
Alex Clemm (Huawei Technologies)
Henk Birkholz (Fraunhofer SIT)
Tianran Zhou (Huawei Technologies)
Objectives

- **Purpose**: to manipulate network close loop automation via configuration of standardized Event-Condition-Action (ECA) containers

- **ECA** – a set of NETCONF style requests/primitives (e.g. get data, edit-config, call-rpc, etc), whose execution on the server is triggered by a specified event, and whose order of execution is conditioned by current and/or historical network states and/or their derivatives

- Explicit **non-goal**: introducing a new interpreter/language/scripting environment
ECAs, when and why

- Reaction to events could be articulated to the network server in advance
- To enhance network responsiveness to events
- To improve scalability of network control
- To configure on the server programmable by a client logic
Policy Variables

• **Policy Variable (PV)** is an ECA state, i.e. a structure to keep results of the ECA execution for immediate or future use

• PV **types**: **global** (shared between ECAs), **local** (ECA scope, static or dynamic)

• PV **content structure**:  
  of a common type (e.g. integer, uint64, etc.)
  Or  
  of an existing YANG node pointed by XPath (e.g. TE_Topologies/links/te_link)
What could be done with PVs?

- **read** from/**write** to YANG data store
- Used as input/output when calling **YANG RPCs**
- Used to generate **notification** messages;
- Used as input/output for **function** calls, for example Fmult(a, 0.75) to calculate 0.75*a
- Used in XPath expressions with PVs referred to by their respective positions in the YANG tree
ECA Events

- Subscribable events:
  - explicitly defined by YANG modules
  - YANG Push or/and smart filter subscriptions

- Timers
ECA Conditions

• Logical expressions with YANG data store nodes and/or PVs
• A condition could be configured as:
  - a single XPath expression
  - a hierarchy of comparisons and logical combinations of thereof
  
  (e.g. (X == Y || A<B) && (C<=D || E>F))
ECA Actions

• NETCONF style primitives:
  - get data, edit-config, etc.
  - calling YANG defined **RPCs** (e.g. TE_TunnelPathComputation RPC defined by YANG TE Tunnel model)
  - sending **notification** messages to the client
  - adding/removing event notification **subscriptions**

• Starting/stopping **timers**

• Calling other **ECAs**

• Performing **operations on PVs** (e.g. function calls)
ECA Structure

- **Event** name
- List of local PVs
- **Normal** Condition-Action list
- **Cleanup** Condition-Action list (to undo actions from the normal Condition-Action list in case one of the normal actions was rejected by the server)