A YANG Data model for ECA Policy Management

draft-ietf-netmod-eca-policy-01

Qin Wu(bill.wu@Huawei.com)
Igor Bryskin (i_bryskin@yahoo.com)
Henk Birkholz (henk.birkholz@sit.fraunhofer.de)
Xufeng Liu (xufeng.liu.ietf@gmail.com)
Benoit Claise(benoit@claise.be)
Andy Bierman (andy@yumaworks.com)
Alexander Clemm(ludwig@clemm.org)
Qiufang Ma (maqiufang1@Huawei.com)
Diego R. Lopez(diego.r.lopez@telefonica.com)
Chongfeng Xie (xiechf@ctbri.com.cn)

Recap

- ECA Enables event-based management
  - provide a useful method to monitor state change of managed objects
- It uses YANG to express network policy and provides rapid autonomic responses to specific conditions,
  - enabling self-management behaviors, including, self-configuration, self-healing, self-optimization, and self-protection.
  - Four type events are discussed, i.e., server event, datastore event, timer event, diagnostic event;
- ECA May be realized in two ways:
  - Centralized network management has its limitations
    - Huge resource consumption due to massive data collection and processing
    - slow reaction to the network changes
    - Lack control on malfunction device
    - Scalability
  - Device Self Management: Move network management function to servers in the network
    - Provide continuous performance monitoring in the server
    - and detect defects and failures and take corrective action in the server.
    - Might require state management and **"Computational Logic"**

Status Update since IETF 109

- draft-ietf-netmod-eca-yang was adopted in January 20 2021
  - Two WG adoption calls was issued in the past, one was in Feb 18, 2020, the second was on December 07, 2020
  - Thanks for supporting and commenting…
    - Andy Bierman, Jonathan Hansford, Alex Clemm, Qiufang Ma, Chongfeng Xie, Diego R. Lopez, Xuefeng Liu, Adrian Farrel, Benoit Claise, Igor Bryskin, Henk Birkholz, Yunbo Yan, Peng Liu, Jan Lindblad, Lingli Deng, Chang Liu, Juergen, Tom Petch, Randy Preshun
  - Special thanks to Andy for provide foundation for this document and providing guidance for the direction of this document.
  - A few issues were raised during adoption call, and will require some further discussion
    - Policy based management definition?
    - What is an adequate abstraction level to express policies and intent?
    - Where are policies executed?
    - When to detect and resolve policy conflicts?
    - Who is interested in interoperable policy representations / languages?

Issue 1: Relationship with I2NSF YANG capability-data-model

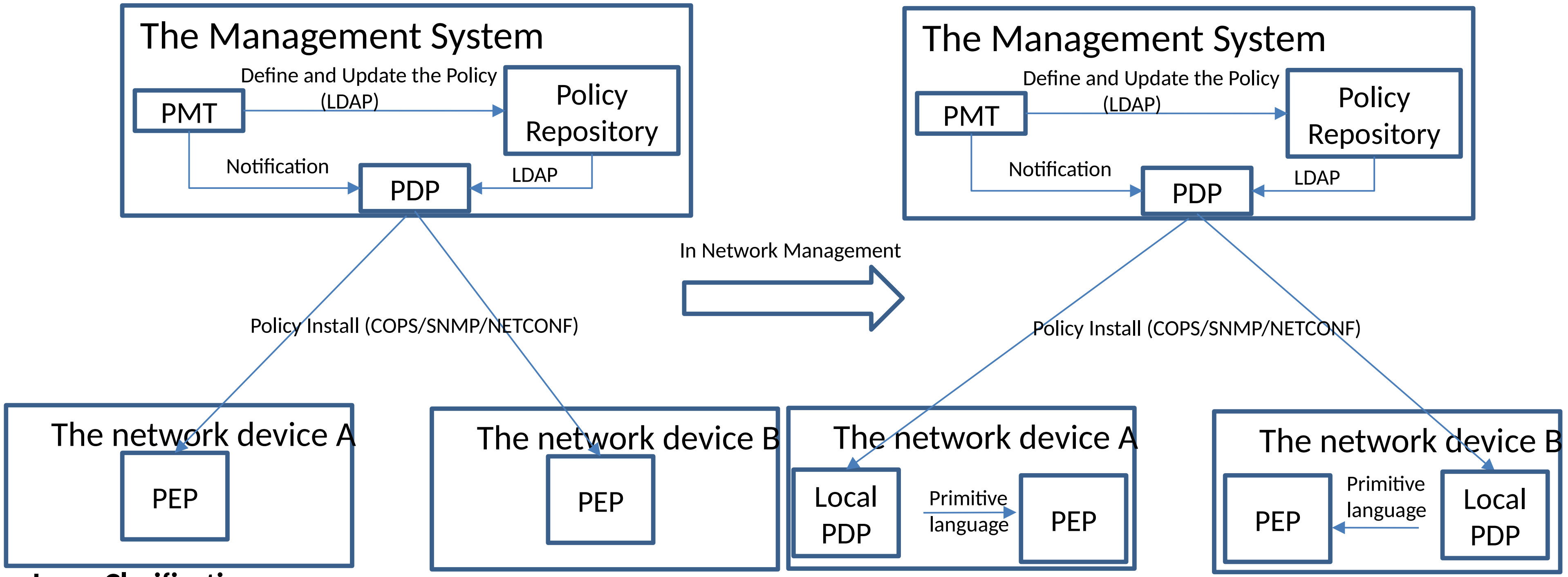
- In RFC8329, ECA is defined as Imperative paradigm related to data packet or data flow treatment:
  - An Event clause is used to trigger the evaluation of the Condition clause of the I2NSF Policy Rule.
  - A Condition clause is used to determine whether or not the set of Actions in the I2NSF Policy Rule can be executed or not.
  - An Action clause defines the type of operations that may be performed on this packet or flow.
- The "Event-Condition-Action" (ECA) policy model in [RFC8329] is used as the basis for the design of the capability model in draft-ietf-i2nsf-capability-data-model;
- In draft-ietf-netmod-eca-policy, similar imperative paradigm is defined
  - The event is defined as one related to datastore subscription or event stream subscription.
  - Condition: Condition can be seen as a logical test that, if satisfied or evaluated to be true, causes the action to be carried out.
  - Action: Update or invocation on local managed object attributes.
- Conclusion:
  - NSF can be an example use case for draft-ietf-netmod-eca-policy (once the technology-specific information is defined in a model)

Issue 2: Abstraction level to express policies and intent

- Policies need to be readable and hence be expressed at a high level of abstraction and in a suitable _language_
  - High level language we select for policy representation is YANG, expressed by the NMS or controller
  - Follow successful story of RMON vs SNMP
    - RMON, an extension of SNMP, provides traffic flow data for troubleshooting and the controls necessary to adjust for better performance from a central console.
    - Argument: do you see RMON as a language?
- High-level policy expression may be compiled down into more verbose primitive representations that are closer to an execution abstraction.
  - Primitive representation in the device is script language such as Python or TCL used in the device.
- A common pitfall is to start somewhere in the middle of several layers of abstraction and then getting stuck with something awkward to put a clean higher layer abstract onto and to compile things down to _efficient_ instrumentations.
  - YANG expression is capable for such a compilation;
  - At the current stage, YANG is used for abstraction and representation. YANG is both representative and implementable.

Issue 3: Where are policies executed

- The most well-known policy-based management was specified joint by IETF and DMTF, which consist of four functional elements: Policy Management Tool (PMT), Policy Decision Point(PDP), Policy Enforcement Point (PEP), Policy repository



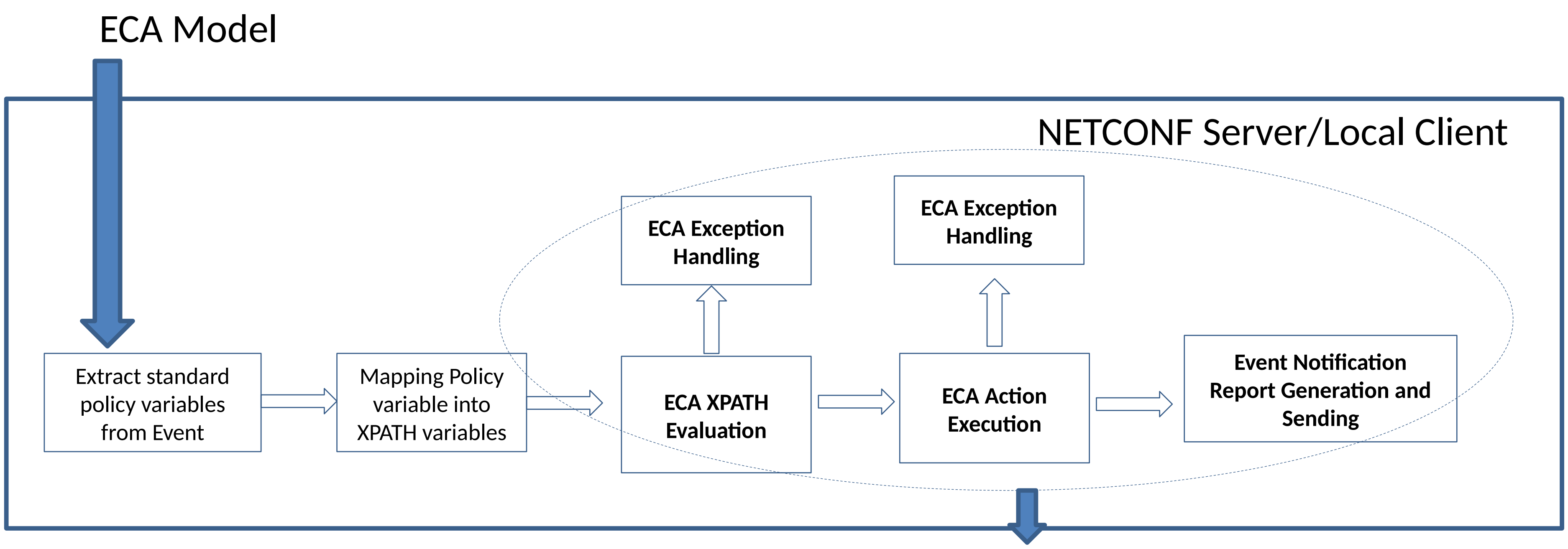
Issue Clarification:

1. In the in-Network management, Management Function is in, or close, to the network element
2. Whether it is traditional network management or In-network management, both policy are enforced or executed In the network element.

What is in the scope:

Our ECA policies are executed by the NETCONF/RESCONF server. This document defines the NETCONF/RESTCONF interface, but not the implementation framework and details.

Issue 4: When to detect and resolve policy conflicts?


ECA Model
NETCONF Server/Local Client
Extract standard policy variables from Event
Mapping Policy variable into XPATH variables
ECA Exception Handling
ECA XPATH Evaluation
ECA Exception Handling
ECA Action Execution
Event Notification Report Generation and Sending

- ECA exception is handled at two phases, one is in the ECA XPATH Evaluation, the second is in the ECA Action execution
- Policy conflict will be detected in the second phase (ECA Action execution) in the device.
- Policy conflict can also be detected in the Policy design/definition stage, i.e., Policy management tool needs to define a policy which has no conflict with existing policy in the Policy repository in the management system.
- Last not the least, Diagnostic event can be used to debug the policy conflict before the policy can be enforced in the network device.

Issue 5: Implementation Experience or Interest?

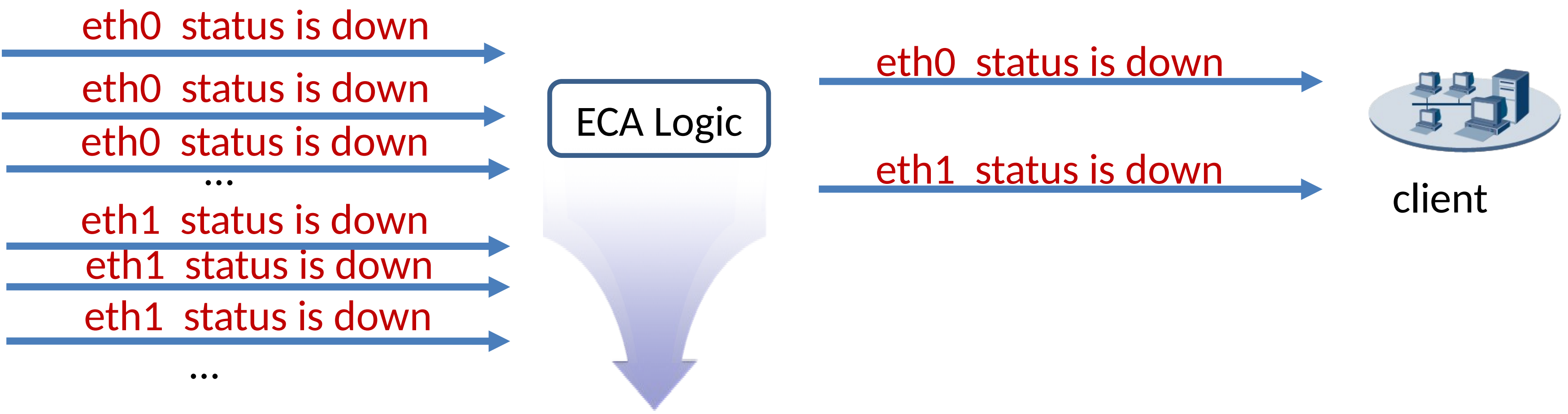
- ECA business model?
  - Embedded Pre-processing capability in the device product
  - Provide telemetry automation in the telemetry data source
- Consumer of the ECA interoperability solution
  - Self management application, Service assurance application, network visibility application, FAPS application, etc., which require both the NMS and the device to support ECA solution
- Is there existing implementation which provide a good basis?
  - Cisco IOS: Embedded event management (EEM)
  - Huawei CE Product: Open Programmability System (OPS)
  - Amazon Simple Notification Service (SNS), S3
- Who is planning to implement?
  - Huawei, we are working on implementing ECA model
  - Potential Open-Source project “Teraflow” (https://5g-ppp.eu/teraflow/) is investigating.

Follow Up and Next Steps

- Address remaining comments raised during adoption and in today's meeting.
  - Issues 1-5
- Add Implementation Status Section
  - Include current examples of ECA usage
  - More examples would be most welcome

ECA Usage Example

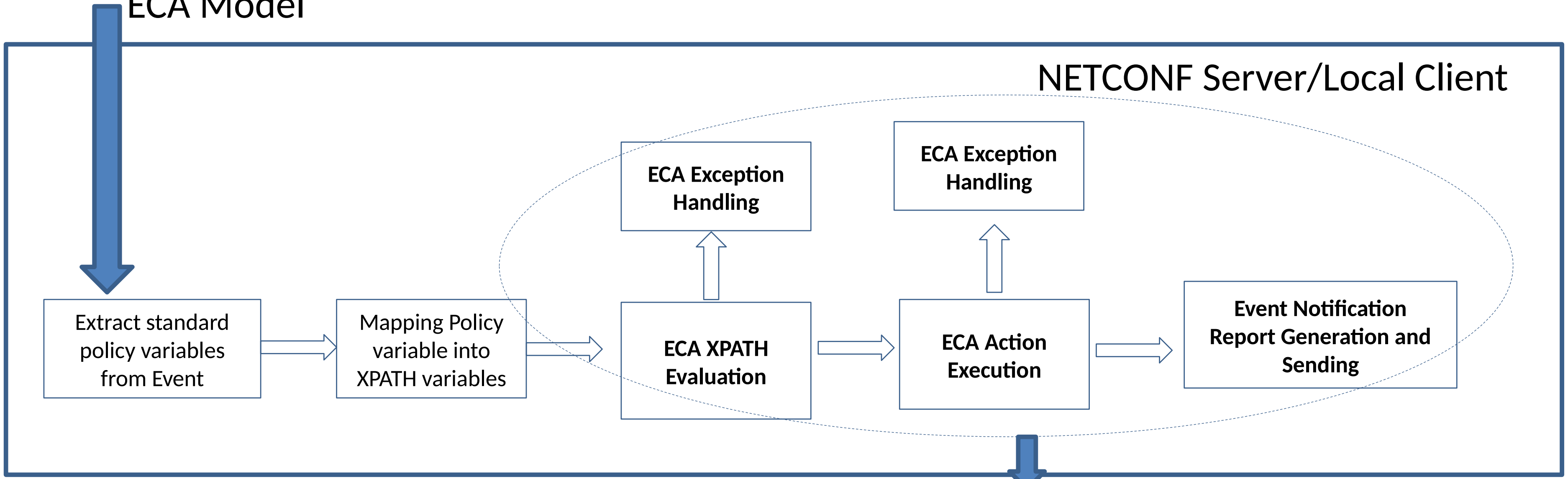
- **A smart filter to suppress duplicated alarm event notification**



ECA Logic Design

1. Subscribe the server event and Scan all Ethernet interfaces and check whether the interface status is down
2.Create event occurrence counter to count the occurrence of the same event
3. If the occurrence time exceeds preconfigured threshold, suppress the event
4. If the occurrence time is below the preconfigured threshold, send the same event notification as one defined in RFC8639.

ECA Logic in the self management Device



ECA Model
NETCONF Server/Local Client
Extract standard policy variables from Event
Mapping Policy variable into XPATH variables
ECA Exception Handling
ECA XPATH Evaluation
ECA Exception Handling
ECA Action Execution
Event Notification Report Generation and Sending

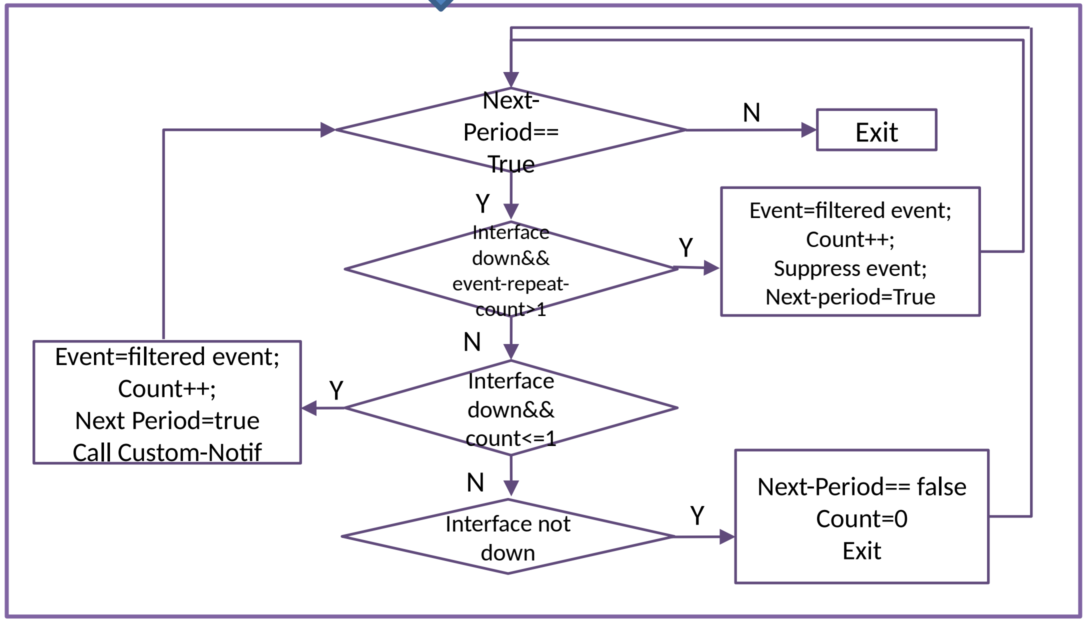
Event:
Event Name:interface-self-monitoring
Event Type: **Server event**
Event Module:IETF-Interface
Event:if:interfaces/if:interface
[if:type=if:gigabitEthernet]

Condition:
Event-repeat-count >1
Event-repeat-count <=1
Interface down=
if:interfaces/if:interface
[if:type=if:gigabitEthernet
,
if:oper-status=down]
Interface not down ..

Extract policy variable

- **Policy Variable:**
Event-repeat-count =0
interface-statistics-event
=if:interfaces/if:interface
[if:type=if:gigabitEthernet]

Map into XPATH variable



Next-Period== True
N
Exit
Y
Interface down&& event-repeat-count>1
Y
Event=filtered event; Count++; Suppress event; Next-period=True
N
Interface down&& count<=1
Y
Event=filtered event; Count++; Next Period=true Call Custom-Notif
N
Interface not down
Y
Next-Period== false Count=0 Exit