A YANG Data model for ECA Policy Management

draft-wwx-netmod-event-yang-10

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(bclaise@cisco.com)
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.
- ECA can 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 105

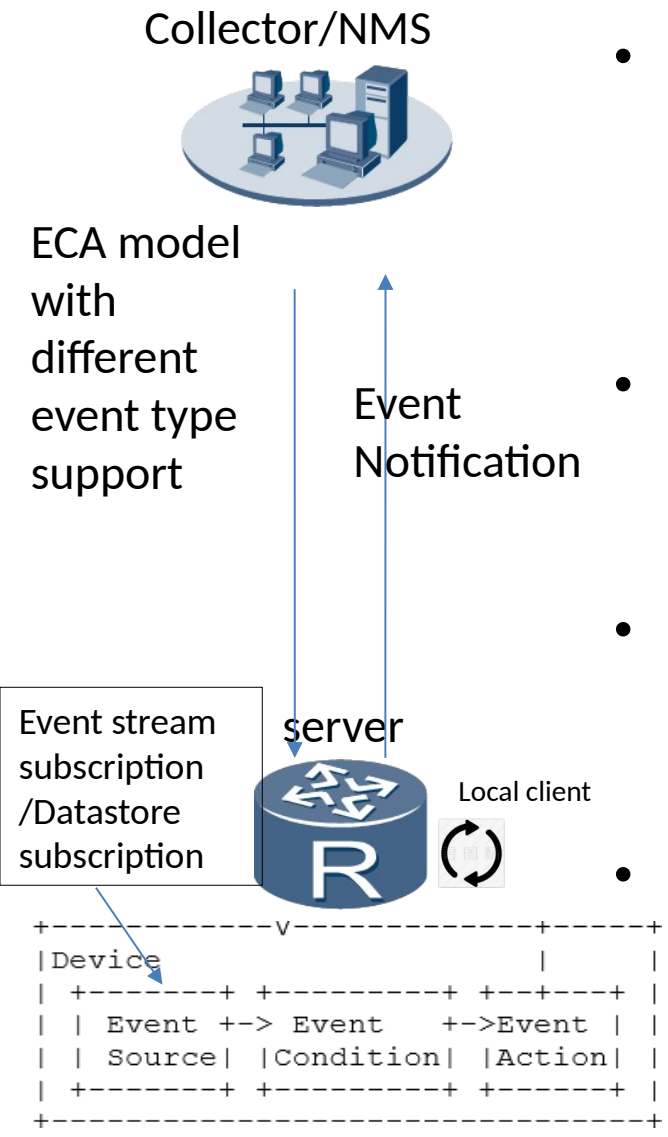
- draft-wwx-netmod-event-yang
  - v-05 was presented in the IETF 106 meeting, and agreed to adopt this work as WG draft.
    - it was suggested by chairs the model should be independent from NETCONF protocol
  - The WG adoption call was started on v-06 at Feb 18, 2020, it was suggested:
    - Use XPath for logical expressions
    - The Server is ECA capable to support XPATH expression and general purpose scripting environment
    - XPath + PVs + new XPath functions and bind ECA components to real scripting languages
  - V-06 was presented in the IETF 107 virtual meeting, and it was agreed to initiate the second round adoption call when it's ready.
  - Work together with Andy, Igor, Henk, Benoit, and other proponents to Address remaining open issues
  - V-09 got reviewed by Jonathan Hansford in section 1 and 2.

- The latest update is v-(10), changes compared to previous versions:
  - Rewrite ECA Model Self Monitoring Usage Example;
  - Add usage Example of High CPU Utilization Troubleshooting;
  - Add usage Example of Router Log Dump using Timer Event Trigger;
  - Reintroduce iterate action, function call and rpc call action type. These action types are exchanged between local client and the server.
  - Move notification operation as separate notification since the notification is exchange between the management system and the server.
  - Add ECA function libraries list in the ECA model.

Terminology Definitions

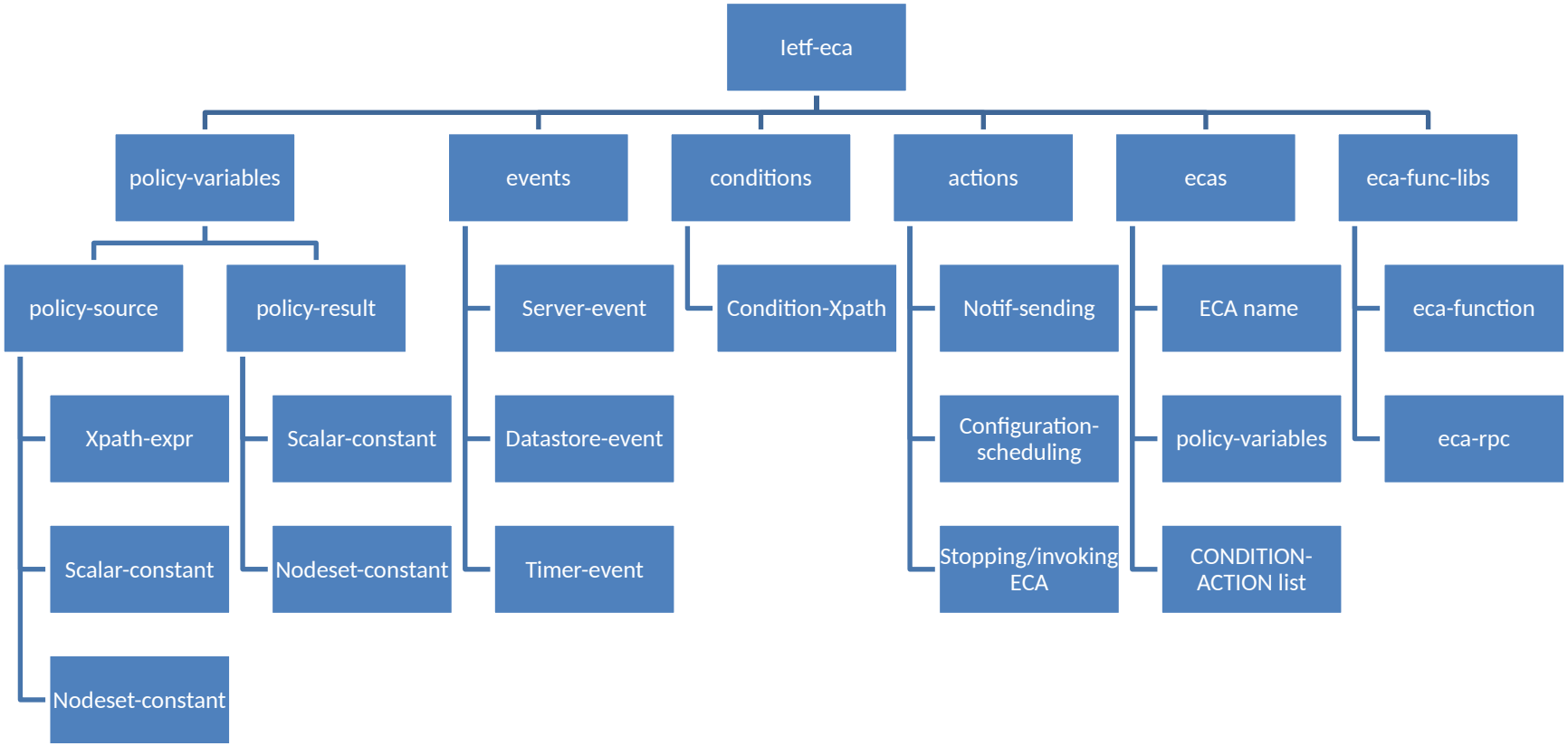
- ECA Event: The input to the ECA logic that initiates the processing derived from an extensible list of platform event types.
- Server Event: An event that happens in the server for which a Notification could be generated in an Event Stream subscription.
- Datastore Event: An event that happens within a datastore within the server for which 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 management Use Cases Classification



- Use Case Category 1: Server Event w/ state
  - e.g., Smart filter
    - scan all interfaces for a certain type every 5 seconds and check the counters or status, return an array of interface entries (XPath node-set) that match the search
- Use Case Category 2: Server Event w/o state
  - when the CPU utilization goes above 60%, output stack, cpu, fan statistics information to a flash
- Use Case Category 3: Timer Event
  - Use a watchdog to dump the router log every 180 seconds to a flash.
- Use Case Category 4: Diagnostic Event with debug mode
  - Debug smart filter ECA logic described above and walk through ECA logic, i.e., run start, next-action, stop action manually in the debug mode on behalf of local client within the server

ECA Model Overview



Four standard events: Datastore event, server event, timer event, Diag Event

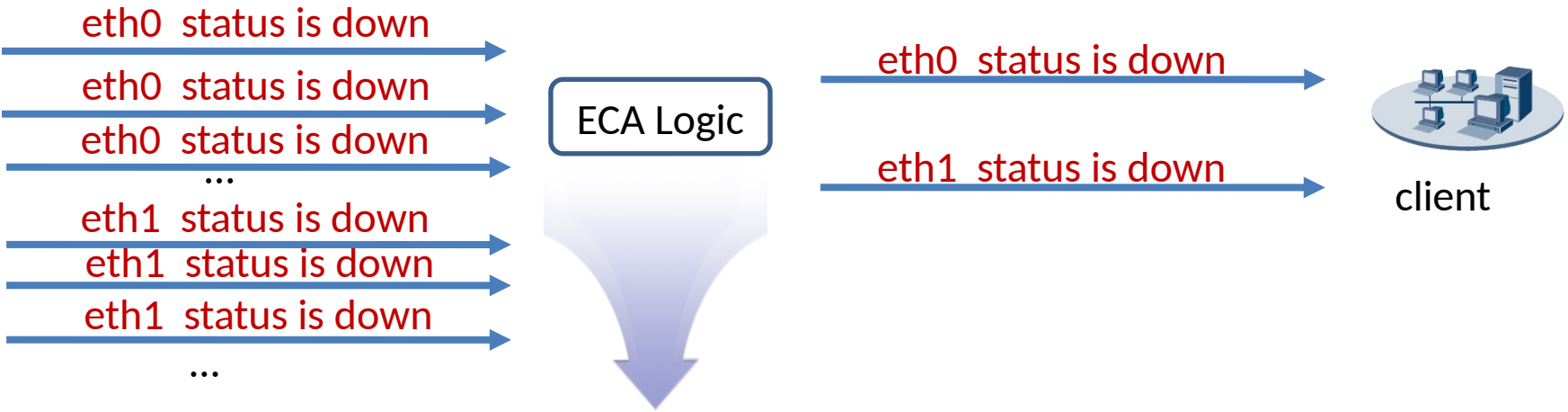
Four key elements: policy-variables, events, conditions, actions

Two type of Policy variables: PV source, PV result

- Event, Condition and Action are defined separately
- and ECAs comprise of a set of ECA entries,
  - each ECA entry have one Event, a set of condition-action entries.
  - Each condition-action entry have a set of conditions, one action entry,
  - each action entry have multiple action items.

ECA Usage example

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

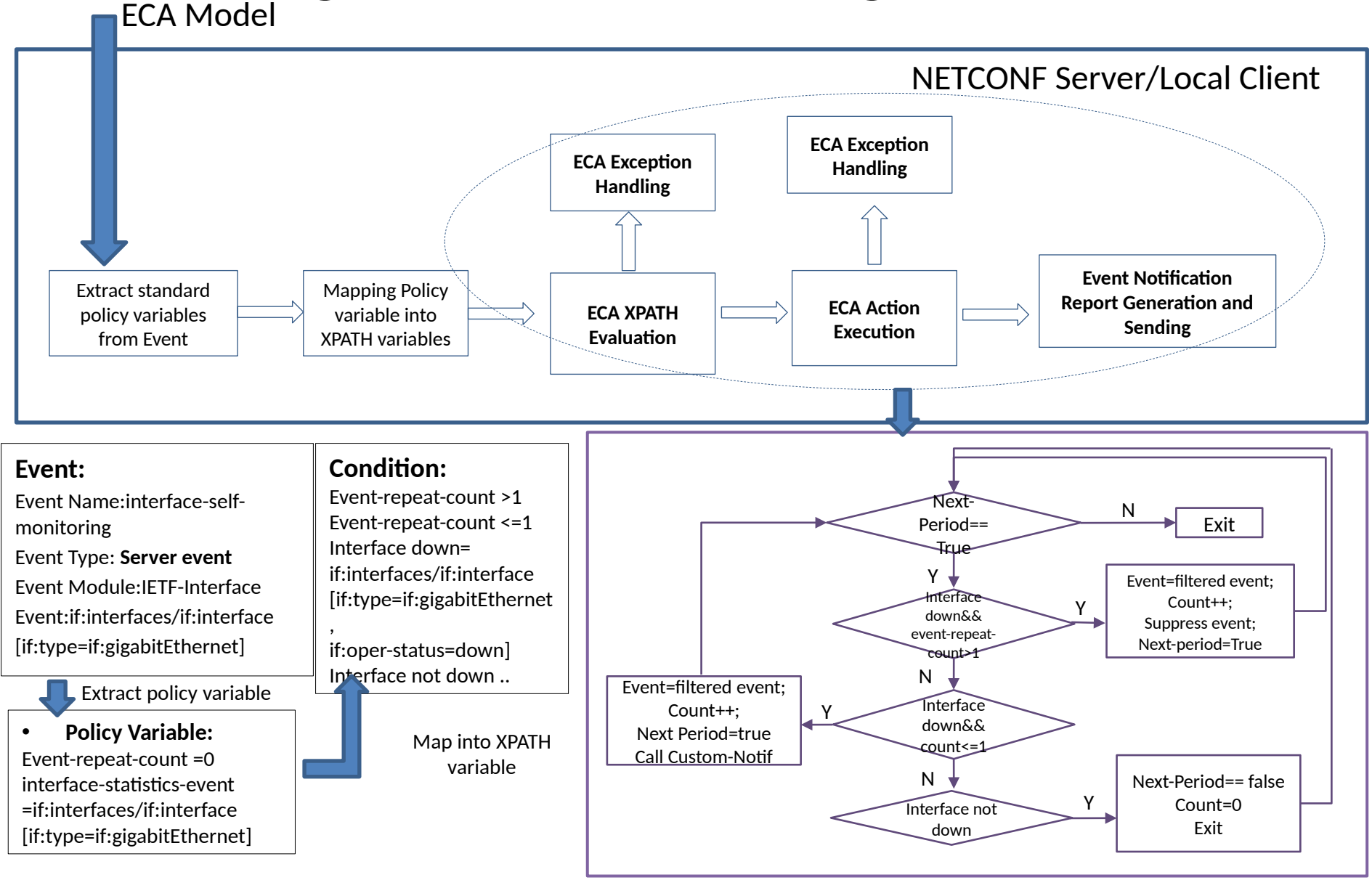


ECA Logic Design

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

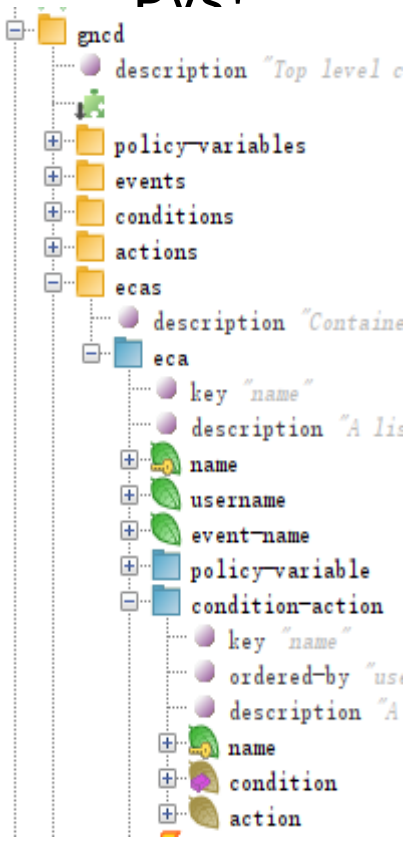


Issue 1：Exception Handling

- Not all errors can be detected at configuration time. Error that occur while ECA logic is being evaluated will cause the server to generate an eca-exception notification.
  - Q1:Does an exception cause the ECA entry to be disabled automatically?
    - YES and No, depends on whether ECA logic is executed one time or periodically
      - In case of one time ECA, it will generate exception notification, stop executing other actions associated with the same event, exit ECA execution process
      - In case of periodical ECA execution, it will generate exception notification, stop executing other action associated with the same event, exit ECA execution process in the current period, move to the next period, if the same exception is generated, the ECA entry will be disabled, send another exception notification;
  - Q2: What else exceptions and details parameters need to be defined?
    - varbind-unknown
    - func-invoke-error
    - Rpc-invoke-error
    - cca-entry-disable

Issue 2: Interaction Between Event, Condition, Action

- How different ECAs do not impact each other if they share PVs?



In the current ECA model, Event, Condition and Action are defined separately and ECAs comprise of a set of ECA entries, each ECA entry have one Event, a set of condition-action entries. Each condition-action entry have a set of conditions, one action entry, each action entry have multiple action items.

Take Smart filter as an example, it is one ECA with one Event, multiple condition and action pairs, two policy variables are defined:
Event-repeat-count is a shared PV and persists across multiple condition-action entries execution operation in the consecutive period;

interface-statistics-event is not shared PV and only exist in each condition action entry execution operation.

ECA interacts with another ECA (i.e., two ECAs have different data source) hasn't been covered yet.

Issue 3: ECA access specific datastores

- 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.
- In the current ECA model draft, only config-false values are targeted.

Issue 4: Diagnostic Test

- Diagnostic Event is a user interface implemented in the server which allow user to view and/or manipulate the ECA logic 's internal state for the purpose of debugging.
- Diagnostic Test provides health check on ECA logic and the ECA XPath function library, e.g., allow user to trigger start action, stop action or next action execution step by step.

Follow Up

- Address remaining comments raised in the meeting.
- Request another adoption call?