Simplified Use of Policy Abstractions (SUPA) Policy Data Model Overview

Michiaki Hayashi

KDDI R&D Labs. Inc

Nov. 3rd, 2015
The GPIM defines a generic structure for imperative and declarative policies.

This is converted to generic YANG data models.

In the preferred approach, SUPA generic policy data models are then used to create vendor- and technology-specific data models.
SUPA Policy Data Model

- A data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol (typically, but not necessarily, all three).

- SUPA generic policy data model is derived from GPIM with semantics defined by GPIM.

- SUPA generic policy YANG data models contain enough information for the Policy Interface to create appropriate input mechanisms for the operator to define policies.

- SUPA Data Model-Specific Translation Function transfers SUPA generic policy data model to vendor- and technology- specific data models.

- For example, an application developer could build an application that uses the SUPA information and data models to directly output configuration snippets.
Structure of Policy Abstractions

The combination of the GPIM and the EPRIM can be used to construct an Event-Condition-Action (ECA) policy data model.

Figure 1: Overview of SUPA Policy Rule Abstractions
Current Issue on Policy Data Model

• The format and content of the Data model is not decided.
  • Scripts or structure or something in between.
  • From Juergen:
    
    “What I find valuable is a framework that allows to write policies that can operate on arbitrary YANG configuration data models. I want to be able to apply policies how my network interfaces are configured without having to write an interface policy data model first.”

• How and who to use the data model?
  • Operators use data model to define policy or they just define the policy, SUPA does the rest.

• Need more example to show how it works
Current work

- ECA Policy YANG Data Model

  - draft-chen-supra-eca-data-model-05

  - **Refined** from GPIM and EPRIM to denote the ECA policy hierarchy.
  - Not perfect but a good start for discussion.
  - Still hand write policy rules with leaf, nodes…
  - Lack of **reusability**
Possible Solution on Events

• **Predefine** a set of events, such as:
  • 1. Interface Counter
  • 2. SNMP
  • 3. Syslog
  • 4. Timers
  • 5. Watchdog system monitor
  • 6. Application Specific

• Or use **reference** to a predefined service model

• Or **leave this to user** to fill in
Possible Solution on Conditions

- Most tricky part and the key of reusability and generality

- Break logic statement into YANG objects:

```
<rw condition-list>
  <rw condition-name>
    <rw (clauseType)?
    | <rw supa-clause-content? string
    | <rw supa-clause-format? string
    | <boolean>
      <rw supa-policy-variable? string
      <rw supa-policy-operator? enumeration
      <rw supa-policy-value? uint32
```

- Or use other scripts embedded in YANG and keep unparsed

```
<condition-linkThreshold>
  <conditionType>script</conditionType> // entity or script or boolean
  <supa-script>
    <supa-script-content>
      <Script:Python>if Subnetwork.link.bandwidth >= 8M: return TRUE</Script:Python>
    </supa-script-content>
    <supa-script-type>Python</supa-script-type> //Python or Perl or any other script
  </supa-script>
</condition-linkThreshold>
```

- Or use xpath statement
Possible Solution on Actions

• Predefined a set of actions, user use it with choice statement, such as:

```
    rw action-list
    rw (actionName)?
    : (remark)
        rw remarkVlanPri? uint32
        rw remarkDscpValue? uint32
        rw applySLALevel? string
    : (car)
        rw cir? uint32
        rw pir? uint32
        rw Cbs? uint32
        rw Pbs? uint32
    : (redirect)
        rw egressInterface? string
        rw egressVpnName? uint32
        rw encapType? enumeration
        rw encapValue? uint32
        rw serviceId? uint32
```

• Or leave this to user to fill in
module: ietf-eca-policy
  +---rw supa-policy
    +---rw supa-policy-name? string
    +---rw supa-policy-priority? uint8
    +---rw supa-policy-validity-period
      +---rw start? yang:date-and-time
      +---rw end? yang:date-and-time
      +---rw duration? uint32
      +---rw periodicity? enumeration
    +---rw supa-policy-target
      +---rw profileType? string
      +---rw asDomainName? string
      +---rw adminSubnetwork? string
      +---rw businessTypeName? string
      +---rw instance
    +---rw supa-policy-atomic
      +---rw supa-eca-policy-rule
        +---rw policy-rule-deploy-status? enumeration
        +---rw policy-rule-exec-status? enumeration
      +---rw supa-eca-component
        +---rw supa-policy-events
          +---rw has-policy-events? boolean
        +---rw supa-policy-conditions
          +---rw has-policy-conditions? boolean
          +---rw conjunctive-type? enumeration
        +---rw supa-policy-actions
          +---rw action-execution? enumeration
      +---rw supa-policy-statement
        +---rw event-list
          +---rw event-name
            +---rw (eventType)?
              +---:(entity)
                +---rw entity? empty
              +---:(script)
                +---rw supa-script-type? scriptType
                +---rw supa-script-content
        +---rw condition-list
        +---rw action-list
Thanks!