TOSCA and YANG
What is it?

Stefan Vallin, Cisco, svallin@cisco.com
Viktor Leijon, Data Ductus, vleijon@cisco.com
Previous Work


Deploying Virtual Network Functions: the Complementary Roles of TOSCA and YANG

Date / Time: Thursday, February 26, 2015, 10:00 AM New York / 3:00 PM London

Overview:

Both TOSCA and YANG are needed to deploy VNFs and the network services they support in an agile model. TOSCA is required for its application-centric deployment model which takes care of the desired, long-lived state of an application throughout its lifecycle while YANG contributes its end-user service centric view and ability to carry out short-lived reconfigurations of virtual network functions, dynamically, at service runtime. This webinar will provide an overview of both modelling languages as well as a discussion on how they may fit into the OSS and the ETSI NFV architecture.
Purpose of this Webinar

• The YANG RFC
• The TOSCA Specification

What?
How do they relate?
Disclaimer

• Only educational based on the specifications
• We will try to avoid partisanship
• We have a YANG background, yes….
I want to do my Service Models in YANG!

I want to do my Service Models in TOSCA!

It is not like a debate between C++ or Java
TOSCA positioning

This is a “service” for an IT OPS guy
TOSCA Overview

- Service Template
- Topology Template
- Graph of Node Templates
- Relationship Templates
Simple TOSCA Example

tosca_definitions_version: tosca_simple_yaml_1_0

description: >
  TOSCA simple profile that just defines a single compute instance and selects a (guest) host Operating S

topology_template:
  inputs:
    cpus:
      type: integer
      description: Number of CPUs for the server.
      constraints:
        - valid_values: [ 1, 2, 4, 8 ]

node_templates:
  my_server:
    type: Compute
capabilities:
  host:
    properties:
      disk_size: 10 GB
      num_cpus: { get_input: cpus }
      mem_size: 4 MB
  os:
    properties:
      architecture: x86_64
      type: Linux
distribution: ubuntu
  version: 12.04
  outputs:
    private_ip:
      description: The private IP address of the deployed server instance.
      value: { get_attribute: [my_server, private_address] }
YANG positioning

This is a “service” for a network engineer
YANG Overview

- Hierarchical configuration data models
- Reusable types and groupings (structured types)
- Extensibility through augmentation mechanisms
- Supports definition of operations (RPCs)
- Formal constraints for configuration validation
- Data modularity through modules and sub-modules
- Well defined versioning rules
Simple YANG Example

```yang
list l3vpn {
  key name;
  leaf name {
    type string;
  }
  list endpoint {
    key "id";
    leaf id {
      type string;
    }
    leaf as-number {
      description "AS used within all VRF of the VPN";
      mandatory true;
      type uint32;
    }
    container ce {
      leaf device {
        mandatory true;
        type leafref {
          path "/ncs:devices/ncs:device/ncs:name";
        }
      }
    }
    container local {
    }
  }
}
```
YANG Instances

**XML**

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <vpn xmlns="http://com/example/l3vpn">
    <l3vpn>
      <name>acme</name>
      <as-number>15602</as-number>
      <endpoint>
        <id>end-point</id>
        <ce-device>ce5</ce-device>
        <ce-interface>GigabitEthernet0/1</ce-interface>
        <ip-network>10.10.0.24</ip-network>
        <bandwidth>1000000</bandwidth>
      </endpoint>
    </l3vpn>
  </vpn>
</config>
```

**JSON**

```
"l3vpn:vpn": {
  "l3vpn": [
    {
      "name": "acme",
      "as-number": 15602,
      "endpoint": [
        {
          "id": "end-point",
          "ce-device": "ce5",
          "ce-interface": "GigabitEthernet0/1",
          "ip-network": "10.10.0.24",
          "bandwidth": 1000000
        }
      ]
    },
    "l3vpn-assure:assurance": {}
  ]
}
```
## Original Scope Comparison

<table>
<thead>
<tr>
<th></th>
<th><strong>TOSCA</strong></th>
<th><strong>YANG</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Org</strong></td>
<td>OASIS</td>
<td>IETF</td>
</tr>
<tr>
<td><strong>Roots</strong></td>
<td>IT, Applications</td>
<td>Networking</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Application-centric, deployment of applications and their dependent artifacts in a data-centers. What needs to be preserved across deployments in different environments? Scaling, healing, up-grade. Design-time focus, Describes Implementation</td>
<td>Configuration of running network devices and applications. What can be configured in a running deployment? Run-time focus Black-Box</td>
</tr>
<tr>
<td><strong>“Slogan”</strong></td>
<td>Bring them up and running</td>
<td>Configure them</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Start CRM-system, SQL DB, IPTV server, network template virtual network at deployment.</td>
<td>Provision a new IPTV service for customer ACME</td>
</tr>
<tr>
<td><strong>Operations</strong></td>
<td>Install, Start, Configure, Scale, Heal, Upgrade, Stop, Uninstall</td>
<td>Operations to configure things after Start, Configure and before Stop</td>
</tr>
<tr>
<td><strong>Personas</strong></td>
<td>Enterprises write template types. Users write template instances</td>
<td>Standard organizations and enterprises writes models. Users writes instances.</td>
</tr>
</tbody>
</table>
# Language Comparison

<table>
<thead>
<tr>
<th>Schema Language</th>
<th>Schema(s) / Models</th>
<th>Instance Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOSCA</td>
<td>TOSCA XSD (1 model)</td>
<td>XML document Workload Template (pre-service instance)</td>
</tr>
<tr>
<td>TOSCA Simple Profile</td>
<td>TOSCA (1 model)</td>
<td>YAML document Workload Template (pre-service instance)</td>
</tr>
<tr>
<td>YANG</td>
<td>Numerous:</td>
<td>XML or JSON Service Instance</td>
</tr>
<tr>
<td></td>
<td>IETF models</td>
<td>Shared Config</td>
</tr>
<tr>
<td></td>
<td>MEF models</td>
<td>Template (if modelled)</td>
</tr>
<tr>
<td></td>
<td>Enterprise specific models</td>
<td>....</td>
</tr>
</tbody>
</table>
## Feature Comparison

<table>
<thead>
<tr>
<th>Area</th>
<th>YANG</th>
<th>TOSCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instances</td>
<td>XML defined by NETCONF</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>JSON defined by RESTCONF</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Complete, extensible</td>
<td>Rudimentary</td>
</tr>
<tr>
<td>Interfaces</td>
<td>Defined by NETCONF &amp; RESTCONF</td>
<td>None</td>
</tr>
<tr>
<td>Concepts</td>
<td>Custom in orchestrator</td>
<td>Defined in standard</td>
</tr>
<tr>
<td>Orchestration</td>
<td>Custom in orchestrator</td>
<td>Defined in standard</td>
</tr>
<tr>
<td>Extensibility</td>
<td><strong>Good</strong></td>
<td>Weak</td>
</tr>
</tbody>
</table>
So…. 

- YANG is a data modeling language used to model configuration and state data  
  - [RFC 6020]

- The TOSCA metamodel uses the concept of service templates to describe cloud workloads as a topology template  
  - [OASIS TOSCA Spec]
“I want to do all my service models in []:

• **TOSCA**
  - Template documents for specific combinations of applications in a data-center
  - Give me a new copy of the workload, “what needs to be deployed”

• **YANG:**
  - Focus on modeling a schema for a specific purpose
  - L3 VPN, MEF Services, EPC, Service Chain
Check-Point, do you get this?

- YANG could/can be used to specify the TOSCA model
- A YANG instance document corresponds to the service template
TOSCA Strengths

- Codifies ideas that have appeal:
  - Composability/Modularity.
  - End-user service models.

- Gives a simple framework for building bigger things.

- Looks easy ("no coding")
TOSCA Weaknesses

- Implementation will not be as easy as it seems:
  - The standard is in-precise
  - Orchestration is not that neatly structured in reality
  - Writing the artifacts is hard
  - Relationships will be locked to types

- Life cycle events are cumbersome

- No good type language

- No instance description

- No protocols

- Hard to extend
Our View
Our view

• TOSCA brings good ideas on application orchestration
  • But in-precise as a standard

• YANG is a well-defined Schema Language
  • But general-purpose, nothing specific on Application Orchestration
What is Orchestration?