NETCONF/YANG: A Southbound Interface for SDN/NFV Configuration and Orchestration

Mi-Jung Choi
Dept. of Computer Science, Kangwon National Univ.
mjchoi@kangwon.ac.kr
2015. 06. 23
Contents

**NETCONF/YANG: SDN/NFV Configuration & Orchestration**
- ONF: OF-CONFIG
- OpenDaylight: YANG + Controller Configuration
- NFV MANO & IETF SFC YANG Models
- CISCO NCS & ConfD

**NETCONF/YANG/RESTCONF**
- NETCONF/YANG/RESTCONF Basics
- Why NETCONF/YANG?
- Comparisons NETCONF/YANG, SNMP & TOSCA
- Implementations of NETCONF/YANG

**Conclusions**

**References**
SDN/NFV Configuration and Orchestration
Need to SDN/NFV Management Protocol

- Formal model and well-known protocol reduces time to market and risk for network equipment providers
- Open, well defined API reduces implementation cost and risk for network management teams at service providers
- SDN & NFV also need a management protocol and information model language in order to reduce cost and risk
- Industrial vendors such as CISCO & Junipers and SDOs such as ONF & OpenDaylight adapted NETCONF/YANG for SDN/NFV configuration & orchestration → Why?
ONF Overview

• Open Networking Foundation
• Standardizing OpenFlow-based SDN
• http://opennetworkingfoundation.org

• Working groups
  • Extensibility
    OpenFlow: wire protocol, extensibility, features,... (OF 1.x)
  • Config-mgmt
    Protocol & schema for configuration of a switch (OF-Config 1.x)
  • Testing-interop
    Interoperability tests, plug-fests; performance benchmarking
  • Hybrid
    OpenFlow in legacy networks; hybrid switches, hybrid networks
  • Architecture
    SDN architecture based on OpenFlow
  • Forwarding
    Abstractions Table Typing, Forwarding plane models
OF-CONFIG 1.2

- An OpenFlow Configuration Point communicates with an operational context which is capable of supporting an OpenFlow Switch using the OpenFlow Configuration and Management Protocol (OF-CONFIG)
Use of NETCONF and YANG

- **NETCONF was chosen as a management protocol**
  - Not necessarily accepted as ideal solution
  - Still discussing alternatives

- **XML schema was chosen as a modeling language**
  - Yang is also used, but XML is normative
  - Normative XML schema generated from Yang code

- **So far, the focus has been on configuration**
  - Bootstrap of an OpenFlow network is the obvious first thing to do

- **New work items will be more on OAM**
  - Including event notifications
What is OpenDaylight building?

- OpenDaylight is an open community that is building:
  - An evolvable SDN platform capable of handling diverse use cases and implementation approaches
  - Common abstractions of capabilities NorthBound for people to program
  - Intermediation of those capabilities to multiple Southbound implementations
  - Programmable network services
  - Network applications
  - Whatever else we need to make it work
Project Framework
OpenDaylight: SDN Controller Architecture
ODL YANG Resource

- **YangTools main page:**
  - https://wiki.opendaylight.org/view/YANG_Tools:Main

- **Code Generation demo**

- **Java “Binding Specification”**:

- **DLUX**
  - Main page: https://wiki.opendaylight.org/view/OpenDaylight_dlux:Main

- **Controller**:
  - Swagger UI Explorer:
  - DLUX (YangUI):
ODL NETCONF Resource

• **Config Subsystem:**
  • How to configure the Netconf connector (client):
    • https://wiki.opendaylight.org/view/OpenDaylight_Controller:Config:Examples:Netconf
  • Netopeer installation

• **Netconf test tool:**
  • https://wiki.opendaylight.org/view/OpenDaylight_Controller:Netconf:Testtool
NFV Objectives and Concepts

- NFV decouples software implementations of Network Functions from the computation, storage, and networking resources
- The decoupling exposes a new set of entities
  - NFV Infrastructure (NFVI)
  - Virtualized Network Functions (VNFs)
  - Physical Network Functions (PNFs)
- Requires a new and different set of Management and Orchestration function
- NFV-MANO architectural framework has the role to manage the NFVI and orchestrate the allocation of resources needed by the NSs and VNFs
  - Virtualized infrastructure
  - Virtualized network function
  - Network service
## NFV Organization

<table>
<thead>
<tr>
<th>WG/EG</th>
<th>Group Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NFV INF Working Group</strong></td>
<td>INF-001 “Infrastructure Overview”, v0.3.6</td>
</tr>
<tr>
<td>(Architecture of Virtualization Infrastructure)</td>
<td>INF-002 “Hypervisor Domain”, v0.3.1</td>
</tr>
<tr>
<td></td>
<td>INF-003 “Compute Domain”, v0.3.1</td>
</tr>
<tr>
<td></td>
<td>INF-004 “Hypervisor Domain”, v0.3.1</td>
</tr>
<tr>
<td></td>
<td>INF-005 “Infrastructure Networking Domain”, v0.3.1</td>
</tr>
<tr>
<td><strong>NFV SWA Working Group</strong></td>
<td>SWA-001 “Virtual Network Function Architecture”, v0.2.1</td>
</tr>
<tr>
<td>(Software Architecture)</td>
<td></td>
</tr>
<tr>
<td><strong>NFV MANO Working Group</strong></td>
<td>MAN-001 “Management and Orchestration”, v0.6.3</td>
</tr>
<tr>
<td>(Management &amp; Orchestration)</td>
<td></td>
</tr>
<tr>
<td><strong>NFV REA Working Group</strong></td>
<td>REL001 “Resiliency Requirements”, v0.1.3</td>
</tr>
<tr>
<td>(Reliability &amp; Availability)</td>
<td></td>
</tr>
<tr>
<td><strong>NFV PER Expert Group</strong></td>
<td>PER-001 “NFV Performance &amp; Portability Best Practices”</td>
</tr>
<tr>
<td>(Performance and Portability)</td>
<td>PER-002 “NFV Proof of Concepts Framework”</td>
</tr>
<tr>
<td><strong>NFV SEC Expert Group</strong></td>
<td>SEC001 “Security Problem Statement”, v0.2.1</td>
</tr>
<tr>
<td>(Security)</td>
<td>SEC002 “Cataloging security features in management software relevant to NFV”, v0.0.4</td>
</tr>
<tr>
<td></td>
<td>SEC003 “Security ad Trust Guidance”, v0.1.0</td>
</tr>
</tbody>
</table>
NFV YANG Schema Examples

• YANG schema for NSD (Network Service Descriptor)
  • NSD is a deployment template for a Network Service referencing all other descriptors which describe components that are part of that Network Service

• YANG schema for VNFD (VNF Descriptor)
  • VNFD is a deployment template which describes a VNF in terms of its deployment and operational behavior requirements

• YANG schema for VLD (Virtual Link Descriptor)
  • VLD is a deployment template which describes the resource requirements that are needed for a link between VNFs, PNFs and endpoints of the Network Service, which could be met by various link options that are available in the NFVI
IETF SFC

- **SFC: Service Function Chaining**
  - An abstract set of service functions and their ordering

- **Constraints**
  - Standardized in IETF SFC WG
IETF SFC WG

- First BoF meeting in July 2013
- First WG meeting in March 2014

- **IETF SFC WG: Charter**
  - Problem statement
  - Architecture
  - Generic SFC encapsulation
  - Control plane mechanisms
  - Manageability
IETF SFC WG I-Ds

- IETF SFC WG charter: 5 WG Internet Draft (I-D), 32 Individual I-Ds
- Currently, SFC WGs standardizes problem definition, use cases, SFC functional architectures
- WG Internet Drafts
  - draft-ietf-sfc-architecture-02, Service Function Chaining (SFC) Architecture
  - draft-ietf-sfc-dc-use-cases-01, Service Function Chaining Use Cases In Data Centers
  - draft-ietf-sfc-long-lived-flow-use-cases-01, SFC Long-lived Flow Use Cases
  - draft-ietf-sfc-problem-statement-10, Service Function Chaining Problem Statement
  - draft-ietf-sfc-use-case-mobility-01, Service Function Chaining Use Cases in Mobile Networks
SFC Yang Data Model

- YANG data model that can be used to configure and manage Service Function Chains
- Module structure of Draft-penno-sfc-yang-13 → 2015. 3. 2 updated

```
import ietf-inet-types
import ietf-yang-type
```

Diagram:

- Service Function Chaining
  - Service Function
  - Service Function Type
  - Service Function Chain
  - Service Node
  - Service Function Path
  - Service Function Forwarder
  - Service Function Forwarder Open vSwitch
  - Service Locator
  - Service Function-Type
  - Service-Locator

All import
Introducing Cisco NCS

- Network abstraction engine
- Data-model driven
- Multi-vendor support
- Device and service models
NCS as an OpenFlow Switch Manager

- Import OF-CONFIG modules!
- No code needed
- All northbound interfaces
- NETCONF feature set
  - Validation
  - Rollback
  - Transactions
What is ConfD?

ConfD provides a data-model driven management plane framework which provides a variety of standards-based northbound management interfaces for use in building network elements:

- Physical Network Devices
- Virtual Network Devices
  (VNF in NFV)
- Network Appliances
- etc.
ConfD Overview

Network Element

- NETCONF
- SNMP
- REST
- CLI
- Web

Management Agent API

Auto-rendered management interfaces and data schema

ConfD Core Engine

CDB Database

CDB API

Data Provider API

Operational Data

External Database

App

YANG Data Models
NETCONF Interface

- **IETF RFC 4741/6241**
  - Full implementation
  - All mandatory and all optional capabilities
- **RFC 4742/6242 NETCONF over SSH**
- **Streams XML over SSH**
- **Separates operational data from configuration**
- **Distributed transactions**
- **RFC 5277 NETCONF notifications**
- **Support included for various IETF standardized YANG data models**

```xml
<get-config>
<edit-config>
<delete-config>
<lock>
<unlock>
<get>
<close-session>
<kill-session>
<commit>
<discard-changes>
```
## Why ConfD?

### Make your customer happy
- Manageable
- Programmable
- Standards compliant

### Save time
- Render management interfaces
- Model-driven
- Iterative development

### Make your device
- NETCONF, CLI, SNMP, REST, Web
- Transactions and rollbacks
- Validations
- Configuration and monitoring
- No feature lag

### Core components
- Embedded database
- Domain-specific functions
- Rich APIs
- Well-tested, 70+ customers
NETCON/YANG/RESTCONF Basics
Why NETCONF?

- Typical Network configuration/monitoring still seen in majority of networks
  - Manual typing/scripting proprietary CLIs + backup repository to track changes, labor intensive, expensive, error prone
  - SNMP extensively used for fault handling and monitoring, but failed for configuration tasks
- Some operator’s requirements that paved the way for NETCONF and YANG (detailed in RFC 3535 – “Overview of the 2002 IAB Network Management Workshop”)
  - Must be easy to use
  - Clear distinction between configuration and operational data
  - Must scale to network-wide configurations rather than being focused on single devices
  - Must provide a way to backup and restore configurations
  - Must provide error-checking to ensure consistent configurations
  - Desirable to be able to process and store results using text-management tools like diff and VCS
  - Distinguish between modifying configuration and activating those modifications
  - Desirable to have multiple configuration stores on devices
- Need for move from “The Network is the Record” approach to “Network-wide” configuration database
NETCONF – High Level Concept

**NETCONF Protocol (RFC 6241)**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>Configuration Data, Notification Data</td>
</tr>
<tr>
<td>Operation (methods)</td>
<td>&lt;get-config&gt;, &lt;edit-config&gt;</td>
</tr>
<tr>
<td>Messages</td>
<td>&lt;rpc&gt;, &lt;rpc-reply&gt;</td>
</tr>
<tr>
<td>Secure Transport</td>
<td>SSH, TLS, ..</td>
</tr>
</tbody>
</table>

**YANG defined**

**Client** -> **Server**
NETCONF Data Stores and Transaction Models

- Data stores are named contains that may hold an entire copy of the configuration.
- Not all data stores are supported by all devices.
- Running is the only mandatory data store.
- Not all data stores are writable.
- Check the device’s capabilities.
- To make changes to a non-writeable data store, copy from a writable one.
- URL is supported by IOS (for config-copy).

**Direct model**

- `<edit-config>`
- `Running` to `Candidate` to `Running` with `<commit>`

**Candidate model (optional)**

- `<edit-config>`
- `Candidate` to `Running` with `<commit>`

**Distinct Startup model (optional)**

- `<edit-config>`
- `Running` to `Startup` with `<copy-config>`
NETCONF Capabilities (1)

- Capabilities are exchanged in hello messages
- RFC 6241 defines some base capabilities
  - :writable-running – the running data store can be modified directly
  - :candidate – the candidate data store is supported
  - :confirmed-commit – the NETCONF server will support the <cancel-commit> and the <confirmed>, <confirm-timeout>, <persist>, and <persist-id> parameters for the <commit> operation
  - :rollback-on-error – server will rollback the configuration to the previous state if an error is encountered
  - :validate – the server will validate the requested data store or config
  - :startup – the startup data store is supported
  - :url – the URL data store is supported
  - :xpath – filtering can be done using XPATH notation
  - :notification – NETCONF asynchronous event messages (RFC 5277)
NETCONF Capabilities (2)

S: <?xml version="1.0" encoding="UTF-8"?>
S:  <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
S:   <capabilities>
S:     <capability>
S:       urn:ietf:params:netconf:base:1.1
S:     </capability>
S:     <capability>
S:     </capability>
S:   </capabilities>
S:   <session-id>4</session-id>
S: </hello>
S:]]>]]>

C: <?xml version="1.0" encoding="UTF-8"?>
C:  <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
C:   <capabilities>
C:     <capability>
C:       urn:ietf:params:netconf:base:1.1
C:     </capability>
C:     </capabilities>
C:   </hello>
C: ]]>]]>
# NETCONF Protocol Operations (1)

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>REQ. CAPABILITY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;get-config&gt;</td>
<td>:base</td>
<td>Retrieve data from the running configuration database</td>
</tr>
<tr>
<td>&lt;get&gt;</td>
<td>:base</td>
<td>Retrieve data from the running configuration database and/or device statistics</td>
</tr>
<tr>
<td>&lt;edit-config&gt;</td>
<td>:base</td>
<td>Modify a configuration database</td>
</tr>
<tr>
<td>&lt;copy-config&gt;</td>
<td>:base</td>
<td>Copy a configuration database</td>
</tr>
<tr>
<td>&lt;delete-config&gt;</td>
<td>:base</td>
<td>Delete a configuration database</td>
</tr>
<tr>
<td>&lt;discard-changes&gt;</td>
<td>:base and :candidate</td>
<td>Clear all changes from the &lt;candidate/&gt; configuration database and make it match the &lt;running/&gt; configuration database</td>
</tr>
<tr>
<td>&lt;create-subscription&gt;</td>
<td>:notification</td>
<td>Create a NETCONF notification subscription</td>
</tr>
<tr>
<td>&lt;lock&gt;</td>
<td>:base</td>
<td>Lock a configuration database so only my session can write</td>
</tr>
<tr>
<td>&lt;unlock&gt;</td>
<td>:base</td>
<td>Unlock a configuration database so any session can write</td>
</tr>
<tr>
<td>&lt;commit&gt;</td>
<td>:base and :candidate</td>
<td>Commit the contents of the &lt;candidate/&gt; configuration database to the &lt;running/&gt; configuration database</td>
</tr>
<tr>
<td>&lt;cancel-commit&gt;</td>
<td></td>
<td>Cancels an ongoing confirmed commit.</td>
</tr>
<tr>
<td>&lt;close-session&gt;</td>
<td>:base</td>
<td>Terminate this session</td>
</tr>
<tr>
<td>&lt;kill-session&gt;</td>
<td>:base</td>
<td>Terminate another session</td>
</tr>
</tbody>
</table>

**DATA MANIPULATION**

**NOTIFICATION MGMT.**

**LOCKING**

**TRANSACTION MGMT.**

**SESSION MGMT.**
NETCONF Protocol Operations (2)

- Client initiates session (typically over SSH) to Server
- Both sides exchange capabilities using <hello> message
- Operations are wrapped in XML-encoded RPC
- Client performs tasks using set of RPC transactions
- Example: Edit-config for device with <running> and <startup> datastore
  - Lock<running>, lock<startup>, edit-config<running>, copy<running> to <startup>, unlock<startup>, unlock<running>
- Example: Edit-config for device with <candidate> datastore
  - Lock<running>, lock<candidate>, edit-config<candidate>, commit <candidate>, unlock<candidate>, unlock<running>
NETCONF: Flow Breakdown – Request (IOS – XR)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <Configuration>
      </Configuration>
    </filter>
  </get-config>
</rpc>
]]>]]>
```
NETCONF: Flow Breakdown - Response (IOS –XR)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="11" xmlns="urn:ietf:params:netconf:base:1.0">
  <data>
    <xml-config-data>
      <Device-Configuration xmlns="urn:cisco:xml-pi">
        <version>
          <Param>15.2</Param>
        </version>
        <service>
          <timestamps>
            <debug>
              <datetime>
                <msec/>
              </datetime>
            </debug>
            </timestamps>
          </service>
        ...
      </Device-Configuration>
    </xml-config-data>
  </data>
</rpc-reply>
```
Why YANG?

- In order for NETCONF to be useful as a network-wide protocol, it must have a common data model
- Simply wrapping CLI in XML is not enough as each vendor has its own CLI
- YANG provides the common data model necessary for to consume NETCONF data from any network device
- Each vendor must implement common YANG modules
- Work on defining these modules is happening in the NETMOD group in the IETF
What is YANG?

- YANG is a modeling language defined in RFC 6020
- Used by NETCONF to define the objects and data in requests and replies
- Analogous to XML schema and SMI for SNMP (but more powerful)
- Models configuration, operational, and RPC data
- Provides semantics to better define NETCONF data
  - Constraints (i.e., “MUSTs”)
  - Reusable structures
  - Built-in and derived types
- YANG is extensible and modular
- YANG modules are for NETCONF what MIBs are for SNMP
NETCONF Concept versus SNMP

**“Framework”**
- Definition language: YANG
- Information model: YANG modules
- Instantiated info/transfer syntax: XML
- Management services: Netconf

**“Content”**
- Ability to express hierarchy
- Richer conditions, constraints
- Facilities for easier reuse
- RPC/Action support

**“Payload”**
- Import conversion rules exist
  (MIBs → YANG)
  “instant content”
- Human readability
- Dynamic extensibility
  B2B, Web toolkits

**Management services:**
- Bulk vs only incremental ops
  (manipulation of config files, e.g. edit-config)
  Transaction support
  Configuration vs monitoring

**“Framework”**
- Definition language: SMIv2
- Information model: MIB modules
- Instantiated info/transfer syntax: ASN.1 BER
- Management service: SNMP

- or possibly other
  (no inherent dependency but will require different bindings)
YANG RPC Methods: delete-config

```yang
rpc delete-config {
    nacm:default-deny-all;
    description "Delete a configuration datastore.";
    reference "RFC 6241, Section 7.4";

    input {
        container target {
            description "Particular configuration to delete.";
            choice config-target {
                mandatory true;
                description "The configuration target to delete.";
                leaf startup {
                    if-feature startup;
                    type empty;
                    description "The startup configuration is the config target.";
                }
                leaf url {
                    if-feature url;
                    type inet:uri;
                    description "The URL-based configuration is the config target.";
                }
            } // choice config-target
        } // container target
    } // input
}
```
Example of YANG Module (1)

```yang
module SystemTime {
  namespace "urn:cisco:params:xml:ns:yang:SystemTime";
  prefix "Cisco-SystemTime";
  organization "CISCO";
  contact "MKRAMOLI@CISCO.COM";
  revision "2014-06-16" {
    description "Example of YANG Schema";
  }
  grouping system_uptime {
    leaf Hostname {
      type string;
      description "Host name";
    }
    leaf Uptime {
      type uint32;
      description "Seconds Up";
    }
    description "System uptime";
  }
  container SystemTime {
    description "System time";
    container Clock {
      config false;
      uses "time_date";
      description "System clock";
    }
    container Uptime {
      config false;
      uses "system_uptime";
      description "Sys. uptime";
    }
  }
}
```
Example of YANG Module (2)

typedef time_source {
    type enumeration {
        enum TIME_SOURCE_ERROR {
            value 0;
            description "Error";
        }
        enum TIME_SOURCE_NONE {
            value 1;
            description "Unsynchronized";
        }
        enum TIME_SOURCE_NTP {
            value 2;
            description "NTP protocol";
        }
        enum TIME_SOURCE_MANUAL {
            value 3;
            description "User configured";
        }
        enum TIME_SOURCECALENDAR {
            value 4;
            description "HW calendar";
        }
    }
    description "Time source";
}
Example of YANG Module (3)

```yang

grouping time_date {
    leaf Year {
        type uint16;
        description "Year [0..65535]";
    }
    leaf Month {
        type uint8;
        description "Month [1..12]";
    }
    leaf Day {
        type uint8;
        description "Day [1..31]";
    }
    leaf Second {
        type uint8;
        description "Second [0..60]";
    }
    leaf Millisecond {
        type uint16;
        description "Millisecond [0..999]";
    }
    leaf TimeZone {
        type string;
        description "Time zone";
    }
    leaf TimeSource {
        type time_source;
        description "Time source";
    }
    description "Date and time";
}
```
YANG Models and Structure

- **YANG modules**
  - Can be Automatically Validated
  - Can be Visualized to UML diagrams, compact Trees, etc.
  - Can be Translated to schemas like DSDL, XSD, etc.
  - Can be Converted to YIN
  - Can be Derived from YIN
  - Can drive Code Generation
YANG Model Execution in NETCONF

- Query/Response for System Time aligned with YANG module definition
- Note: screenshots taken from IOS XRv 5.1.1
### YANG Models – Industry and Cisco

**IETF**
- Interface management [RFC 7223]
- IP management [draft-ietf-netmod-ip-cfg]
- System management [draft-ietf-netmod-system-mgmt]
- SNMP configuration [draft-ietf-netmod-snmp-cfg]
- Generic OAM [Cisco Involvement, draft-tissa-netmod-oam]
- OSPF [Cisco Involvement, draft-yeung-netmod-ospf-01]
- BGP [Cisco Involvement, draft-zhdankin-netmod-bgp-cfg-00]
- IPFIX configuration [Cisco involvement, RFC6728]
- ACL configuration [Cisco involvement, draft-huang-netmod-acl-03]
- Network topology [Cisco involvement, draft-clemm-i2rs-yangnetwork-topo-00.txt]
- Routing management [draft-ietf-netmod-routing-cfg]
- RIB [I2RS] [Cisco involvement, draft-clemm-i2rs-yangnetwork-topo-00]
- Netconf monitoring [RFC6022], Netconf access control [RFC6536]

**Cisco:** PIM, IPSLA, L2VPN, VLAN, DNA, Synthetic models XR
- Cablelabs: CCAP (Converged Cable Access Point)
- ONF: Openflow Switch Configuration (OF-Config)
- MIBs (for monitoring data) via SMIPv2 ->YANG conversion

- YANG@CISCO to be supported over NETCONF, REST, or XMPP
- YANG modules of interest
  - draft-ietf-netmod-system-mgmt
  - draft-ietf-netmod-interfaces-cfg
  - draft-ietf-netmod-ip-cfg
  - draft-ietf-netmod-routing-cfg
  - draft-ietf-ipfix-configuration-model
- Customer-driven modules for VLAN, QoS, environment, and ACL configuration
RESTCONF

- Still an emerging story (draft-bierman-netconf-restconf-4)
- RESTful protocol to access YANG defined data
- Representational State Transfer, i.e. server maintains no session state
- URIs reflect data hierarchy in a NETCONF datastore
- HTTP as transport
- Data encoded with either XML or JSON
- Operations

<table>
<thead>
<tr>
<th>RESTCONF</th>
<th>Netconf</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>&lt;get-config&gt;, &lt;get&gt;</td>
</tr>
<tr>
<td>POST</td>
<td>&lt;edit-config&gt;(&quot;create&quot;)</td>
</tr>
<tr>
<td>PUT</td>
<td>&lt;edit-config&gt;(&quot;replace&quot;)</td>
</tr>
<tr>
<td>PATCH</td>
<td>&lt;edit-config&gt;(&quot;merge&quot;)</td>
</tr>
<tr>
<td>DELETE</td>
<td>&lt;edit-config&gt;(&quot;delete&quot;)</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>(discover supported operations)</td>
</tr>
<tr>
<td>HEAD</td>
<td>(get without body)</td>
</tr>
</tbody>
</table>
YANG Mapping to JSON

- **JSON** is a popular compact and easy to parse data format used by many REST APIs
- Subset of YANG compatible XML documents can be translated to JSON text
- Translation driven by YANG data model (must be known in advance)
- YANG datatype information is used to translate leaf values to the most appropriate JSON representation
- Slightly more compact (irrelevant with compression)
- Increased human readability (less noise)
YANG Mapping to JSON vs. XML

JSON – 214 octets*

```
{  
  "ietf-interfaces:interfaces": {  
    "interface": [  
      {  
        "name": "eth0",  
        "type": "ethernetCsmacd",  
        "location": "0",  
        "enabled": true,  
        "if-index": 2  
      },  
      {  
        "name": "eth1",  
        "type": "ethernetCsmacd",  
        "location": "1",  
        "enabled": false,  
        "if-index": 2  
      }  
    ]  
  }  
}
```

XML – 347 octets*

```
<interfaces xmlns:="urn:ietfparams:xml:ns:yang:ietf-interfaces">!  
  <interface>  
    <name>eth0</name>  
    <type>ethernetCsmacd</type>  
    <location>0</location>  
    <enabled>true</enabled>  
    <if-index>2</if-index>  
  </interface>  
  <interface>  
    <name>eth1</name>  
    <type>ethernetCsmacd</type>  
    <location>1</location>  
    <enabled>false</enabled>  
    <if-index>7</if-index>  
  </interface>  
</interfaces>
```

*all white space removed
RESTCONF Example

C: GET /restconf/operators/operational/opendaylight-inventory:nodes HTTP/1.1
C: Host: example.com
S: HTTP/1.1 200 OK
S: Date: Fri, 6 June 2014 17:01:00 GMT
S: Server: example-server
S: Content-Type: application/json

S:{
S:   "nodes": {
S:     "node": [
S:       {
S:         "flow-node-inventory:hardware": "Test vSwitch",
S:         "flow-node-inventory:software": "1.1.0",
S:         "id": "openflow:1",
S:         "flow-node-inventory:switch-features": {
S:           "flow-node-inventory:capabilities": [
S:             "flow-node-inventory:flow-feature-capability-flow-stats",
S:             "flow-node-inventory:flow-feature-capability-port-stats",
S:           ],
S:           "flow-node-inventory:max_buffers": 256,
S:           "flow-node-inventory:max_tables": 255
S:         }
S:       }
S:   }
S:}
TOSCA & NETCONF/YANG Applicability

NETCONF/YANG

EMS1

NFVI

VNF1

Virtual Computing
Virtual Storage
Virtual Network
Virtualization Layer
Computing Hardware
Storage Hardware
Network Hardware

BSS

OSS

VNF2

EMS2

EMS3

NFV Orchestrator
VNF Manager(s)

Virtualized Infrastructure Manager(s)

NFV Management and Orchestration (MANO)

TOSCA

NFVI
# A Comparison of TOSCA & YANG

<table>
<thead>
<tr>
<th>Organization</th>
<th>OASIS</th>
<th>IETF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roots</td>
<td>IT applications</td>
<td>Network service / devices</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>Application-centric, lifecycle management of applications and their dependent artefacts in a data center. Design-time focus on what needs to be preserved and implemented across deployments in different environments, e.g., scaling, healing, upgrade.</td>
<td>Configuration of running network devices and network services, Runtime focus on what can be configured in a running deployment.</td>
</tr>
<tr>
<td><strong>“Slogan”</strong></td>
<td>Manage them operationally ready and able to run well</td>
<td>Configure them</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Start CRM-system, SQL DB, IPTV server, network service</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 and before Stop</td>
</tr>
<tr>
<td><strong>Lifecycle Phase</strong></td>
<td>Deployment and at runtime for specific functions (update, scaling, healing)</td>
<td>Runtime(service fulfilment)</td>
</tr>
<tr>
<td><strong>Technologies</strong></td>
<td>Topology template: application structure; Plans: process models to create and terminate (and manage) applications; Bundled in a physical CSAR file (zipped directory), Data model : YAML</td>
<td>Protocol : NETCONF Data model : YANG</td>
</tr>
</tbody>
</table>
NETCONF/YANG Implementations
NETCONF Implementations (Open source)

- **Ensuite**
  - Ensuite is a network management platform prototype based on Netconf and Yang

- **libnetconf**
  - Netconf library in C intended for building Netconf clients and servers

- **ncclient**
  - Python library for Netconf clients

- **NetconfX**
  - Provides an implementation of the client-side of a Netconf interface in Java

- **netopeer**
  - Remote configuration system using Netconf protocol
YANG Implementations (Open source)

- **libsmi**
  - Contains a YANG output driver for the smidump tool
  - Generates YANG modules from SMI/SMIv2 MIBs

- **webinterface to smidump**
  - Now supports YANG output

- **pyang (manual page)**
  - An extensible YANG validator written in Python
  - Can be used standalone as a validator of YANG modules, or to generate YIN, YANG, DSDL and XSD from YANG and YIN
  - Can also be used to generate UML diagrams from YANG data models

- **yangdump**

- **jYang**
  - A YANG validator and translator written in Java
Commercial Implementations of NETCONF/YANG

- **Yumaworks’s “YumaPro SDK”**
  - YANG-based Unified Modular Automation Toolkit NETCONF, HTTP/REST and CLI
  - It includes a NETCONF-over-SSH client and server, YANG Complier, and other development tools
  - Key Feature of Yumapro SDK
    - Data-driven network management tools slash development costs
    - Yang-based API code generation increases feature consistency and stability
    - Reliable Netconf, Rest, and CLI transactions with rollback simplifies configuration management
  - Yumaworks provides server/client tools as netconfd-pro, yangcli-pro, respectively
Implementation of SDN/NFV with NETCONF/YANG (1)

- Tail-f’s ‘NCS(Network Control System)’
  - Tail-f performs a leading role in developing and implementing of NETCONF protocol and YANG data modeling language (recently acquired by Cisco)
  - NCS provides a single network-wide interface to all network devices and all network applications and services, as well as a common modeling language and datastore for both services and devices.
  - Services, device configurations and Openflow applications are specified in declarative YANG data models

- Cyan’s ‘Blue Planet SDN’
  - Network management via a variety of protocols and interfaces such as CLI, TL1, NETCONF/YANG and others
  - Modular platform that includes a multi-layer WAN SDN controller, NFV orchestrator, FCAPS management of both physical and virtual elements, advanced visualization, and more
Implementation of SDN/NFV with NETCONF/YANG (3)

• **Brocade’s ‘Vyatta Controller’ (1/2)**
  - Vyatta Controller is the first commercial controller built directly from OpenDaylight code, allows users to freely optimize their network infrastructure to address the needs of their specific workloads
  - Vyatta Controller is part of Brocade’s “new IP” vision and will incorporate OpenStack orchestration, OpenDaylight-based control, as well as virtual and physical network gear
  - Vyatta Controller is open source, but Brocade will offer a commercial version
Implementation of SDN/NFV with NETCONF/YANG (3)

- Brocade’s ‘Vyatta Controller’ (2/2)
Who Is Using It?

Equipment Vendors
- Brocade (MLX, NetIron)
- Ericsson (SEA 20)
- Cisco (IOS, IOS-XE)
- H3C (S9500E)
- Huawei (AR x200)
- Juniper (JUNOS >7.5)
- Sonus (NBS 5200)
- Verivue (MDX 9020)

SDOs
- IETF
  - Drives standard
- CableLabs
  - CCAP (Next-gen CMTS)
- MEF
  - FM, PM, and services
- ONF
  - OF-CONFIG
Conclusion
Conclusion

• Many vendors and SDOs adapt open and standardized NETCONF and YANG for SDN/NFV configuration and orchestration

• NETCONF can a single protocol for managing configuration for both the traditional, and software defined network

• Open Source vs. Open Standards debate
  • When interoperability is concerned open standards are essential, and having a standard set of Yang models would be advantageous
  • It also has one other benefit, enabled by RESTCONF(Device-level Northbound API standardization)
References (1)

- ETSI NFV MANO, GS NFV-MAN 001 V0.6.2
- Stefan Wallin, “Tutorial: NETCONF and YANG”, May 2013
- RFC 6244: NETCONF+Yang Architectural Overview
- RFC 6241: Base NETCONF Protocol
- RFC 6020: YANG Base Specification
- draft-penno-sfc-yang-07: Yang Data Model for Service Function Chaining
- MG-Soft, Visual YANG Designer
- YANG-Central, http://www.yang-central.org
References (2)

• Tail-f, ConfD Technical Introduction
• Tail-f, Systems ConfD Overview
• Tail-f, OpenFlow Switch Management using NETCONF and YANG
• Jan Medved, Yang Modeling in OpenDaylight
• Heavy Reading, Deploying Virtual Network Functions : The Complementary Roles of TOSCA & NETCONF/YANG
• Martin Kramolis, NETCONF, YANG, RESTCONF
• ONF, OF-CONFIG 1.2 OpenFlow Management and Configuration Protocol
• ONF, Jürgen Quittek, ONF Configuration and Management WG