NETCONF/YANG

- Let’s get started
  - What is NETCONF?
  - What is YANG?
- NETCONF
  - What is the difference?
  - Network Management Datastore Architecture (RFC 8342)
  - configuration datastore:
    - NETCONF Operations
    - How does it work?
- YANG
  - Built-in data types
  - Common data types
- References/Links

Let’s get started

What is NETCONF?

A network managing protocol to configure network elements.

It transports xml (json) via SSH.

```xml
<rpc-reply message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface name="eth0">
        <enabled>true</enabled>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address ip="2001:db8:c18:1::3" prefix-length="128" />
        </ipv6>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

What is YANG?

A data modeling language for the definition of data sent over the NETCONF network configuration protocol.

It is a schema to validate xml.

```xml
container interfaces {
  description "Interface configuration parameters."
  list interface {
    key "name";
    leaf name {
      type string;
    }
    leaf type {
      type identityref {
        base interface-type;
      } mandatory true;
    }
    leaf enabled {
      type boolean;
      default "true";
    }
  } // list interface
} // container interfaces
```

NETCONF

NETCONF was designed based on Operators Requirements, which are documented in RFC3535. It addresses security topics and operational and maintenance topics.

What is the difference?

<table>
<thead>
<tr>
<th>Protocol:</th>
<th>SNMP</th>
<th>NETCONF</th>
<th>SOAP</th>
<th>REST</th>
<th>RESTCONF</th>
<th>gRPC</th>
<th>What comes next ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport Stack</td>
<td>UDP (connectionless)</td>
<td>SSH TCP</td>
<td>SSL HTTP TCP</td>
<td>SSL HTTP TCP</td>
<td>SSL HTTP TCP</td>
<td>HTTP/2 TCP</td>
<td></td>
</tr>
<tr>
<td>Encoding</td>
<td>BER</td>
<td>XML (new JSON)</td>
<td>XML</td>
<td>XML JSON</td>
<td>XML JSON</td>
<td>binary</td>
<td></td>
</tr>
<tr>
<td>Resources</td>
<td>OIDs</td>
<td>Path (XPath on server)</td>
<td>URLs</td>
<td>URLs</td>
<td>URLs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data models</td>
<td>MIBs</td>
<td>YANG modules</td>
<td>WSDL, XSD</td>
<td>YANG modules</td>
<td>Protocol Buffers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Modeling Language</td>
<td>SMI</td>
<td>YANG</td>
<td>WSDL, XSD</td>
<td>Undefined, (WSDL), WADL, text...</td>
<td>YANG</td>
<td>Protocol Buffers</td>
<td></td>
</tr>
<tr>
<td>Management Operations</td>
<td>SNMP operations</td>
<td>NETCONF operations</td>
<td>In the XML Schema, not standardized</td>
<td>HTTP operations</td>
<td>HTTP operations</td>
<td>HTTP/2 operations</td>
<td></td>
</tr>
<tr>
<td>SDO (like)</td>
<td>IETF</td>
<td>IETF</td>
<td>W3C</td>
<td>W3C</td>
<td>IETF</td>
<td>Google</td>
<td></td>
</tr>
</tbody>
</table>
Network Management Datastore Architecture (RFC 8342)

configuration datastore:
The datastore holding the complete set of configuration data that is required to get a device from its initial default state into a desired operational state.

running configuration datastore:
A configuration datastore (<running>) holding the complete configuration currently active on the device. The running configuration datastore always exists.

candidate configuration datastore:
A candidate datastore (<candidate>) that can be manipulated without impacting the device's current configuration and that can be committed to the running configuration datastore. Not all devices support a candidate configuration datastore.

startup configuration datastore:
The startup datastore (<startup>) holding the configuration loaded by the device when it boots. Only present on devices that separate the startup configuration datastore from the running configuration datastore.

operational datastore:
The operational state datastore (<operational>) is a read-only datastore that consists of all "config true" and "config false" nodes defined in the datastore's schema.

NETCONF Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get</td>
<td>Retrieve running configuration and device state information.</td>
</tr>
<tr>
<td>get-config</td>
<td>Retrieve all or part of a specified configuration datastore.</td>
</tr>
<tr>
<td>edit-config</td>
<td>Loads all or part of a specified configuration to the specified target (&lt;running&gt;, &lt;candidate&gt;) configuration datastore. operation-types: merge, replace, create, delete, remove</td>
</tr>
<tr>
<td>copy-config</td>
<td>Create or replace an entire configuration datastore with the contents of another complete configuration datastore.</td>
</tr>
<tr>
<td>delete-config</td>
<td>Delete a configuration datastore. The &lt;running&gt; configuration datastore cannot be deleted.</td>
</tr>
<tr>
<td>(partly-) lock</td>
<td>It allows the client to lock the entire configuration datastore system of a device.</td>
</tr>
<tr>
<td>(partly-) unlock</td>
<td>Releases a configuration lock, previously obtained with the &lt;lock&gt; operation.</td>
</tr>
<tr>
<td>commit</td>
<td>Sets the running configuration to the current contents of the candidate configuration.</td>
</tr>
<tr>
<td>validate</td>
<td>This protocol operation validates the contents of the specified configuration.</td>
</tr>
<tr>
<td>close-session</td>
<td>Request graceful termination of a NETCONF session.</td>
</tr>
<tr>
<td>kill-session</td>
<td>Force the termination of a NETCONF session.</td>
</tr>
<tr>
<td>(hello-message)</td>
<td>(exchange of yang capabilities (yang modules) between server and client)</td>
</tr>
</tbody>
</table>

How does it work?
(one of x possibilities)
YANG

- YANG is a data modeling language for NETCONF (and RESTCONF) configuration and state data.
- It defines the syntax of the exchanged xml (and json) data between a NETCONF client (e.g. OpenDaylight) and a NETCONF server (a device).

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
<td>Any binary data</td>
</tr>
<tr>
<td>bits</td>
<td>A set of bits or flags</td>
</tr>
<tr>
<td>boolean</td>
<td>&quot;true&quot; or &quot;false&quot;</td>
</tr>
<tr>
<td>decimal64</td>
<td>64-bit signed decimal number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ietf-yang-types</td>
<td></td>
</tr>
<tr>
<td>counter32</td>
<td>Counter32 (SNMPv2-SMI)</td>
</tr>
<tr>
<td>zero-based-counter32</td>
<td>ZeroBasedCounter32 (RMON2-MIB)</td>
</tr>
<tr>
<td>counter64</td>
<td>Counter64 (SNMPv2-SMI)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ietf- inet-types</td>
<td></td>
</tr>
<tr>
<td>ip-version</td>
<td>InetVersion (INET-ADDRESS-MIB)</td>
</tr>
<tr>
<td>dscp</td>
<td>Dscp (DIFFSERV-DSCP-TC)</td>
</tr>
</tbody>
</table>
empty | A leaf that does not have any value
---|---
enumeration | Enumerated strings
identityref | A reference to an abstract identity
instance-identifier | References a data tree node
[u]int | [8|16|32|64]-bit [un]signed integer
leafref | A reference to a leaf instance
string | Human-readable string
union | Choice of member types
zero-based-counter64 | ZeroBasedCounter64 (HCNUM-TC)
gauge32 | Gauge32 (SNMPv2-SMI)
gauge64 | CounterBasedGauge64 (HCNUM-TC)
object-identifier | OBJECT IDENTIFIER
object-identifier-128
date-and-time
timeticks | TimeTicks (SNMPv2-SMI)
timestamp | TimeStamp (SNMPv2-TC)
phys-address | PhysAddress (SNMPv2-TC)
mac-address | MacAddress (SNMPv2-TC)
xpath1.0
hex-string
uuid
dotted-quad
ipv6-flow-label | IPv6FlowLabel (IPV6-FLOW-LABEL-MIB)
port-number | InetPortNumber (INET-ADDRESS-MIB)
as-number | InetAutonomousSystemNumber
ip-address
ipv4-address
ipv6-address
ip-address-no-zone
ipv4-address-no-zone
ipv6-address-no-zone
ip-prefix
ipv4-prefix
ipv6-prefix
domain-name
host
uri | Uri (URI-TC-MIB)

References/Links

RFC 3535  Overview of the 2002 IAB Network Management Workshop (see chapter 3 – Requirements by Network Operators)
RFC 6241  Network Configuration Protocol (NETCONF)
RFC 6020  YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)
RFC 6991  Common YANG Data Types
RFC 7950  The YANG 1.1 Data Modeling Language
Overview  NETCONF and YANG Overview
Tutorial  YANG Tutorial
Tutorial  YANG Boot Camp
pyang  An extensible YANG validator and converter in python
pyang  pyang documentation