LET’S BUILD TOMORROW TODAY
A Model-driven Approach to Software Defined Networks with Yang, NETCONF/RESTCONF

Wojciech Dec
BRKSDN-1903
Agenda

• Context
• Introduction
• Data Modeling Basics
• Overview of NETCONF
• RESTCONF – providing http access
• Exercise - Simple IPSec Service
• Summary
Model driven networking

The (ancient) Context

From a long long time ago:
A single language (and model)

Tower of babel – c.a. 570BC
Model driven networking

Modern day Context

To today’s language and model cornucopia

Networking Landscape – c.a. 2014AD
Model Driven Networking Context

Enable faster deployment of services
Simplification and automation of both network and OSS systems

Starting with data models representing services and configuration

Using standard protocols to transmit modeled data. Protocol is selected based on the task suitability & consumer community.
Introduction
What are Data Models, NETCONF, Yang, RESTCONF?
Information/Data Models, and Protocols

Ancient concepts in communication…

**Information Model**

*Eg A Verb, a noun, etc. in natural languages:*

**Protocol & Representation**

*Eg Visual characters, alphabet, sentence structure*

*The model & protocol need to be shared*
Information/Data Models, and Protocols
Ancient concepts in communication…

Data Model

E.g. Yang Types, Leafs, Lists:

Protocol & Representation

E.g. NETCONF, RESTCONF

The model & protocol need to be shared
NETCONF, RESTCONF and YANG in Context

Data Models + Right protocols for the task

- NETCONF
- RESTCONF
- YANG

Higher Level Applications

HTTP:RESTCONF

Network Control Applications

Yang Models

Data

Protocol & Representation

Yang Models

Data

Yang Models

Data

Yang Models

Data
Products Supporting Yang/NETCONF/RESTCONF

- Cisco
  - Tail-f Network Control System (NETCONF/Yang)
  - Tail-f Confd Agent (Nectonf/Yang)
  - Open SDN Controller/Opendaylight (NETCONF/Yang/RESTCONF)
  - IOS-XR (NETCONF/Yang)
  - NXOS (NETCONF)
  - IOS-XE (NETCONF)

- Industry
  - Juniper, Huawei, Ericsson, …
### What makes NETCONF/YANG & RESTCONF different?

<table>
<thead>
<tr>
<th></th>
<th>SNMP</th>
<th>NETCONF</th>
<th>SOAP</th>
<th>REST</th>
<th>RESTConf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>IETF</td>
<td>IETF</td>
<td>W3C</td>
<td>-</td>
<td>IETF</td>
</tr>
<tr>
<td>Resources</td>
<td>OIDs</td>
<td>Paths</td>
<td>URLs</td>
<td>URLs</td>
<td></td>
</tr>
<tr>
<td>Data models</td>
<td>Defined in MIBs</td>
<td>YANG Core Models</td>
<td>(WSDL, not data)</td>
<td>Undefined, (WSDL), WADL, text...</td>
<td>YANG Models</td>
</tr>
<tr>
<td>Data Modeling Language</td>
<td>SMI</td>
<td>YANG</td>
<td></td>
<td></td>
<td>YANG</td>
</tr>
<tr>
<td>Management Operations</td>
<td>SNMP</td>
<td>NETCONF</td>
<td>In the XML Schema, not standardized</td>
<td>HTTP operations</td>
<td>HTTP operations</td>
</tr>
<tr>
<td>Encoding</td>
<td>BER</td>
<td>XML</td>
<td>XML</td>
<td>XML, JSON,...</td>
<td>XML, JSON,...</td>
</tr>
<tr>
<td>Transport Stack</td>
<td>UDP</td>
<td>SSH</td>
<td>SSL</td>
<td>SSL</td>
<td>SSL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TCP</td>
<td>HTTP</td>
<td>HTTP</td>
<td>TCP</td>
</tr>
</tbody>
</table>

SMI: Simple Management Interface
YANG: Yang
W3C: World Wide Web Consortium
IETF: Internet Engineering Task Force
WSDL: Web Service Description Language
WADL: Web Application Description Language
UDP: User Datagram Protocol
TCP: Transmission Control Protocol
SSL: Secure Sockets Layer
HTTP: HyperText Transfer Protocol
Yang Data Modeling Basics
**YANG?**

- IETF Standard (RFC6020) Data modeling language used to model network:
  - Configuration data
  - State data
  - RPCs
  - Notifications
- Tree structure
- Data and Types
YANG ....

- Readability is highest priority
  - Compare with other schema languages e.g. RelaxNg, XSD.

- Extensible
  - Add new content to existing data models
  - Add new statements to the YANG language
    - Vendor extensions and future proofing

- Limited Scope
  - Doesn't boil the ocean – intended to describe networking concepts in a tree structure

- Directly maps to XML representation
  - JSON Mapping also

- Allows preservation of investment in SNMP MIBs
  - libsmi tool translates MIBs to YANG
  - See tools at www.yang-central.org
YANG Concepts

- Standard Models
- Proprietary Models
- Config/operational data
- RPCs (Operations)
- notifications
- Yang data nodes & types

Models ↔ Verbe, Nouns, ...

BRKSDN-1903 © 2015 Cisco and/or its affiliates. All rights reserved. Cisco Public
Modules can use submodules

- Header statements
  - yang-version, namespace, prefix
- Linkage statement
  - import and include
- Meta information
  - organization, contact
- Revision history
  - revision
Submodules

module acme-module {
  namespace "...";
  prefix acme;

  import "ietf-yang-types" {
    prefix yang;
  }
  include "acme-system";

  organization "ACME Inc.";
  contact joe@acme.example.com;
  description "Module describing the ACME products";
  revision 2007-06-09 {
    description "Initial revision.";
  }
}

submodule acme-system {
  belongs-to acme-module {
    prefix acme;
  }
  import "ietf-yang-types" {
    prefix yang;
  }
  container system {
    ...
  }
}

Attention: The submodule cannot reference definitions in main module.

Each imported submodule is standalone.

Each included submodule belongs to one specific main module.
YANG Base Types

- Most YANG elements have a data type
- Type may be a base type or derived type
  - Derived types may be simple typedefs or groupings (structures)
  - There are 20+ base types to start with

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>int8/16/32/64</td>
<td>Integer</td>
</tr>
<tr>
<td>uint8/16/32/64</td>
<td>Unsigned integer</td>
</tr>
<tr>
<td>decimal64</td>
<td>Non-integer</td>
</tr>
<tr>
<td>string</td>
<td>Unicode string</td>
</tr>
<tr>
<td>enumeration</td>
<td>Set of alternatives</td>
</tr>
<tr>
<td>boolean</td>
<td>True or false</td>
</tr>
<tr>
<td>bits</td>
<td>Boolean array</td>
</tr>
<tr>
<td>binary</td>
<td>Binary BLOB</td>
</tr>
<tr>
<td>leafref</td>
<td>Reference “pointer”</td>
</tr>
<tr>
<td>identityref</td>
<td>Unique identity</td>
</tr>
<tr>
<td>empty</td>
<td>No value, void</td>
</tr>
</tbody>
</table>

...and more
Typedef Statement

Defines a new simple data type

typedef percent {
    type uint16 {
        range "0 .. 100";
    }
    description "Percentage";
}

leaf completed {
    type percent;
}
Type Restrictions

### Integers

typedef my-base-int32-type {
  type int32 {
    range "1..4 | 10..20";
  }
}

typedef derived-int32 {
  type my-base-int32-type {
    range "11..max"; // 11..20
  }
}

### Strings

typedef my-base-str-type {
  type string {
    length "1..255";
  }
}

typedef derived-str {
  type my-base-str-type {
    length "11 | 42..max";
    pattern "[0-9a-fA-F]*";
  }
}
Common YANG Types

• Commonly used YANG types defined in RFC 6021

• Use

import "ietf-yang-types" {
  prefix yang;
}

to reference these types as e.g.

type yang:counter64;

counter32/64  ipv4-address

gauge32/64    ipv6-address

object-identifier  ip-prefix

date-and-time     ipv4-prefix

timeticks         ipv6-prefix

timestamp         domain-name

phys-address      uri

ip-version        mac-address

flow-label        bridgeid

port-number       vlanid

ip-address        ... and more

www.rfc-editor.org/rfc/rfc6021.txt
Grouping Statement

Defines a new structured type

grouping target {
    leaf address {
        type inet:ip-address;
        description "Target IP";
    }
    leaf port {
        type inet:port-number;
        description "Target port number";
    }
}

container peer {
    container destination {
        uses target;
    }
}
YANG Data Node Definitions
Leaf Statement

Holds a single value of a particular type

Has no children

```yaml
leaf host-name {
  type string;
  mandatory true;
  config true;
  description "Hostname for this system";
}
leaf cpu-temp {
  type int32;
  units degrees-celsius;
  config false;
  description "Current temperature in CPU";
}
```
## Attributes for leaf

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>config</code></td>
<td>Whether this leaf is a configurable value (&quot;true&quot;) or operational value (&quot;false&quot;). Inherited from parent container if not specified.</td>
</tr>
<tr>
<td><code>default</code></td>
<td>Specifies default value for this leaf. Implies that leaf is optional.</td>
</tr>
<tr>
<td><code>mandatory</code></td>
<td>Whether the leaf is mandatory (&quot;true&quot;) or optional (&quot;false&quot;).</td>
</tr>
<tr>
<td><code>must</code></td>
<td>XPath constraint that will be enforced for this leaf.</td>
</tr>
<tr>
<td><code>type</code></td>
<td>The data type (and range etc) of this leaf.</td>
</tr>
<tr>
<td><code>when</code></td>
<td>Conditional leaf, only present if XPath expression is true.</td>
</tr>
<tr>
<td><code>description</code></td>
<td>Human readable definition and help text for this leaf.</td>
</tr>
<tr>
<td><code>reference</code></td>
<td>Human readable reference to some other element or spec.</td>
</tr>
<tr>
<td><code>units</code></td>
<td>Human readable unit specification (e.g. Hz, MB/s, °F).</td>
</tr>
<tr>
<td><code>status</code></td>
<td>Whether this leaf is &quot;current&quot;, &quot;deprecated&quot; or &quot;obsolete&quot;.</td>
</tr>
</tbody>
</table>
Container Statement

Groups related leafs and containers

container system {
    container services {
        container ssh {
            presence "Enables SSH";
            description "SSH service specific configuration";
            // more leafs, containers and other things here...
        }
    }
}

Leaf-list Statement

Holds multiple values of a particular type

Has no children

```plaintext
leaf-list domain-search {
    type string;
    ordered-by user;
    description "List of domain names to search";
}
```
List Statement

```
list user {
    key name;
    leaf name {
        type string;
    }
    leaf uid {
        type uint32;
    }
    leaf full-name {
        type string;
    }
    leaf class {
        type string;
        default viewer;
    }
}
```
### Attributes for list and leaf-list

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-elements</td>
<td>Max number of elements in list. If max-elements is not specified, there is no upper limit, i.e. “unbounded”</td>
</tr>
<tr>
<td>min-elements</td>
<td>Min number of elements in list. If min-elements is not specified, there is no lower limit, i.e. 0</td>
</tr>
<tr>
<td>ordered-by</td>
<td>List entries are sorted by “system” or “user”. System means elements are sorted in a natural order (numerically, alphabetically, etc). User means the order the operator entered them in is preserved. “ordered-by user” is meaningful when the order among the elements have significance, e.g. DNS server search order or firewall rules.</td>
</tr>
</tbody>
</table>
The key field is used to specify which row we’re talking about.

No two rows can have same key value.

/user[name='yang']/name = yang
/user[name='yang']/uid = 1010
/user[name='yang']/class = admin

/user[name='ling']/class = viewer
Augment Statement

```yml
augment /sys:system/sys:user {
  leaf expire {
    type yang:date-and-time;
  }
}
```
Identity Statement

Identities for modeling families of related enumeration constants

module phys-if { ...

  identity ethernet {
    description "Ethernet family of PHY interfaces";
  }

  identity eth-1G {
    base ethernet;
    description "1 GigEth";
  }

  identity eth-10G {
    base ethernet;
    description "10 GigEth";
  }

  ...

} module newer { ...

  identity eth-40G {
    base phys-if:ethernet;
    description "40 GigEth";
  }

  identity eth-100G {
    base phys-if:ethernet;
    description "100 GigEth";
  }

  ...

  leaf eth-type {
    type identityref {
      base "phys-if:ethernet";
    }
  }

  }
Versioning Yang Modules

You must / must not:
- Add a revision statement on top
- Update organization, contact, etc
- Do not rename module or namespace
- Do not remove obsolete definitions
- Do not reorder data definitions

You may:
- Add enum, bits, typedef, grouping, rpc, notification, extension, feature, identity, case
- Add non-mandatory data definitions
- Add mandatory data definitions within new if-feature statements

RFC 6020 sec 10
Yang Resources

RFC6020 IETF Spec:

Yang Central Useful tools and document collection:
http://www.yang-central.org/twiki/bin/view/Main/WebHome
NETCONF overview
So What is NETCONF?

• NETCONF is an IETF standard (RFC 6241) network management protocol. Provides:
  • Distinction between configuration and state data
  • Multiple configuration data stores (candidate, running, startup)
  • Configuration change transactions
  • Selective data retrieval with filtering
  • Event notifications
  • Extensible remote procedure call mechanism
**What makes NETCONF different?**

<table>
<thead>
<tr>
<th>Use Case</th>
<th>SNMP</th>
<th>NETCONF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get collection of status fields</td>
<td>Yes</td>
<td>Yes. Bulk xfer up to 10x faster. Really...</td>
</tr>
<tr>
<td>Set collection of configuration fields</td>
<td>Yes, up to 64kB</td>
<td>Yes</td>
</tr>
<tr>
<td>Set configuration fields in transaction</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Transactions across multiple network elements</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Send event notifications</td>
<td>Yes</td>
<td>Yes, connected</td>
</tr>
<tr>
<td>Secure protocol</td>
<td>v3 is fair</td>
<td>Yes</td>
</tr>
<tr>
<td>Test configuration before final commit</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
• Named configuration stores
  • Each data store may hold a full copy of the configuration

• Running is mandatory, Startup and Candidate optional (capabilities :startup, :candidate)

• Running may or may not be directly writable (capability :writable-running)
  • Need to copy from other stores if not directly writable

NETCONF Configuration Data Stores

Startup

Running

Candidate

Files… / URLs…
NETCONF Transactions, Network-wide Transactions

Transaction support is a key NETCONF feature.

Using the Candidate data store, a NETCONF Manager can implement a network-wide transaction.

- Send a configuration change to the candidate of each participating device
- Validate candidate
- If all participants are fine, tell all participating devices to commit changes

If satisfactory, commit. If not, drop the connection to the devices.

- Connection closed/lost is the NETCONF command for abort transaction
- All devices will roll back
NETCONF Layering Model

Layer
- Content
- Operations
- RPC
- Transport Protocol

NETCONF
- Configuration Data
  - <get> <get-config>
  - <notification>
  - <rpc> <rpc-reply>
  - SSH, SSL, BEEP, etc
NETCONF Transport

NETCONF is server-client and connected oriented using TCP

- Typically managed device (e.g. Router) is NETCONF Server

NETCONF messages are encoded in XML

- RFC defines message framing

NETCONF messages are encrypted by SSH

- NETCONF over SOAP, BEEP (both now deprecated) and TLS are also defined, but not used
NETCONF Capabilities and Extensibility

A NETCONF Manager (Client) connecting to a NETCONF Server (Device), says

\texttt{<hello>}

The contents of the \texttt{<hello>} message declares which NETCONF Capabilities each party is capable of.

- Some capabilities are defined by the base NETCONF specification
- Each YANG Data model the device knows is also a capability
- Other specifications (standards body or proprietary) also define capabilities

By declaring support for a capability in \texttt{<hello>}, the manager will know which operations it can submit to the Device.
NETCONF <hello> Operation

```xml
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:confirmed-commit:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
<capability>http://tail-f.com/ns/netconf/with-defaults/1.0</capability>
<capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
<capability>http://tail-f.com/ns/netconf/commit/1.0</capability>
<capability>http://tail-f.com/ns/example/dhcpd?module=dhcpd</capability>
</capabilities>
<session-id>5</session-id>
</hello>
```
NETCONF Base Operations

- `<get>`
- `<get-config>`
- `<edit-config>`
  - test-option (:validate)
  - error-option
  - operation
- `<copy-config>`
- `<commit>` (:candidate, :confirmed)
- `<discard-changes>` (:candidate)
- `<cancel-commit>` (:candidate)

- `<delete-config>`
- `<lock>`
- `<unlock>`
- `<close-session>`
- `<kill-session>`
NETCONF <get-config> Operation

- Sub-tree filtering
- XPATH filtering

```xml
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="1">
  <get-config>
    <source>
      <running/>
    </source>
    <filter xmlns="http://tail-f.com/ns/aaa/1.1">
      <aaa/>
    </filter>
  </get-config>
</rpc>
```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="1">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <dhcp xmlns="http://tail-f.com/ns/example/dhcpd"
        <defaultLeaseTime nc:operation="merge">PT1H</defaultLeaseTime>
      </dhcp>
    </config>
  </edit-config>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="1">
  <ok/>
</rpc-reply>
More on NETCONF Transactions

Using the Candidate data store a NETCONF Manager can implement a network wide transaction.

• Send a configuration change to the candidate of each participating device
• Validate candidate
• If all participants are fine, tell all participating devices to commit changes

Confirmed-commit allows a manager to activate a change, and test it for a while

• Measure KPIs, test connectivity, …
If satisfactory, commit. If not, drop the connection to the devices.

• Connection closed/lost is the NETCONF command for abort transaction
• All devices will roll back
NETCONF Example Configuration Sequence

```xml
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="5">
  <edit-config xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <target>
      <candidate/>
    </target>
    <error-option>rollback-on-error</error-option>
    <config>
      <interface xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <name>eth1</name>
        <ipv4-address>192.168.5.10</ipv4-address>
        <macaddr>aa:bb:cc:dd:ee:ff</macaddr>
      </interface>
    </config>
  </edit-config>
</rpc>

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="6">
  <validate/>
  <source>
    <candidate/>
  </source>
</rpc>

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="7">
  <commit>
    <confirmed/>
  </commit>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="5">
  <ok/>
</rpc-reply>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="6">
  <ok/>
</rpc-reply>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="7">
  <ok/>
</rpc-reply>
```

Config is applied by can still be rolled back!
RESTConf

REST-like protocol for access to Yang modeled data
RESTCONF - why?

• How many people you know that can read the following?
  • draft-ietf-netconf-restconf*
    • “RESTCONF uses HTTP operations to provide CRUD operations on a NETCONF datastore containing YANG-defined data.”

• Allows use of familiar HTTP Tools and programming libraries
• Tap into pool of developers familiar with HTTP & REST
• Supports XML or JSON data representation
• Why REST-like?
  • Is Yang model based
  • This is actually an advantage: Schema allows API predictability and automation vs model discovery

*Presentation assumes version 00 of the IETF draft
RESTCONF API (1/4)

Constructing the URI for data or operations

A RESTCONF URI is:

```
/restconf/<resource-type>/<yang-module:resource>
```

Module `my.interfaces`{
  namespace "com.my.interfaces";

  container interfaces {
    list interface {
      key name;
      leaf name { type string; }
      leaf admin-status { type enum; }
    }
  }

  rpc flap-interface {
    input {
      leaf name { type string; }
    }
    output {
      leaf result { type boolean; }
    }
  }
}

URI: `/restconf/data/my.interfaces:interfaces`

URI: `/restconf/operations/my.interfaces:flap-interface`
RESTCONF API (2/4)

Containers & lists

**A RESTCONF URI is:**

```
/restconf/<resource-type>/<yang-module:resource>
```

Module my-interfaces {
  namespace "com.my-interfaces";

  container interfaces {
    list interface {
      key name;
      leaf name { type string; }  
      leaf admin-status { type 
      enum; }
    }
  }

URI:
/restconf/data/my-interfaces:interfaces

URI:
/restconf/data/my-interfaces:interfaces/interface/<some name>
RESTCONF API (3/4)

Acting on resources

Module my-interfaces {
{
    namespace "com.my-interfaces";
    container interfaces {
        list interface {
            key name;
            leaf name { type string; }
            leaf admin-status { type enum; }
        }
        rpc flap-interface {
            input {
                leaf name { type string; }
            }
            output {
                leaf result { type boolean; }
            }
        }
    }
}

GET: Gets a resource

GET /restconf/data/my-interfaces:interfaces
GET /restconf/data/my-interfaces:interfaces/interface/<some name>

POST: Creates a resource or invoke operation

POST /restconf/operations/my-interfaces:flap-interface + JSON/XML Form Data (including name)
Response will have JSON/XML result

PUT: Replaces a resource

PUT /restconf/data/my-interfaces:interfaces/interface/<some name> + JSON/XML Form Data (admin-status)

DELETE: Removes a resource

DELETE /restconf/data/my-interfaces:interfaces/interface/<some name>
RESTCONF API (4/4)

List Data.

- RESTCONF requires that data passed includes module name-space.
- Creation of List items is done using **POST** to resource: /restconf/data/my-interfaces:interfaces with JSON or XML data expressing name.

```
Module my-interfaces {
    namespace "com.my-interfaces";

    container interfaces {
        list interface {
            key name;
            leaf name { type string; }
            leaf admin-status { type enum; }
        }
    }
}
```

```json
{ "my-interfaces:interface" : [ 
    { "name" : "GigabitEthernet0/0", 
      "status" : up 
    } ]
}
```

```xml
<interface xmlns="com.my-interfaces">
    <name>GigabitEthernet0/0</name>
    <admin-status>up</admin-status>
</interface>
```
Example:
Using Yang to model a simple IPSec Service and derive its RESTCONF API
Example: IPSec Hub Service

Spoke1:
IP: 10.194.126.2/24
Peer: Hub1
Subnet IP: 192.168.1.0/24

Spoke2:
IP: 10.194.127.1/24
Peer: Hub1
Subnet IP: 192.168.2.0/24

Controller

Device Config:
crypto isakmp key MY_K3Y
  address 10.194.126.2
  crypto ipsec transform-set TS
    esp-des esp-md5-hmac
  crypto map CRYPTO 10 ipsec-isakmp
    set peer 10.194.126.2
    set transform-set TS
    match address Spoke1
  !
  interface Gig0/0
    ip address 10.194.128.1 255.255.255.0
    crypto map CRYPTO
    !

Example:
IPSec Hub Service

Service IPSec (Model)
Nodes:
  Hub1, Spoke1, Spoke2, …
Parameters:
  IP addresses, Encryption Alg, Keys, etc

Devices Config (Models)
Vendors, models/versions, features, etc
Spoke1:
IP: 10.194.126.2/24
LAN Network
IP: 192.168.1.0/24

Spoke2:
IP: 10.194.127.1/24
LAN Network
IP: 192.168.2.0/24

Controller
App
Service IPSec
(Model)
Nodes:
Hub1, Spoke1, Spoke2, …
Parameters:
IP addresses, Encryption Alg,
Keys, etc

Connectivity Service App
Customer A Secure VPN
Service:
Spoke1, Spoke2, …
crypto isakmp key MY_K3Y
address 10.194.126.2
!
crypto ipsec transform-set TS
esp-des esp-md5-hmac
!
crypto isakmp
set peer 10.194.126.2
set transform-set TS
match address Spoke1
!
interface Gig0/0
ip address 10.194.128.1 255.255.255.0
crypto map CRYPTO
!

crypto ipsec transform-set TS
esp-des esp-md5-hmac
!
crypto isakmp
set peer 10.194.126.2
set transform-set TS
match address Spoke1
!
interface Gig0/0
ip address 10.194.128.1 255.255.255.0
crypto map CRYPTO
!

NETCONF
RestCONF

Separation of concerns

Application: I care about the service to Customer A, not about network configuration
Service Data Model

Controller: I care about rendering the required device configuration for the service, not about running it
Device Data Models

Devices: We care about running the configuration
Device Data Models
Let’s model the IPSec service!

The Basics

module ipsec-service {

    namespace "com.example.sipsec";
    prefix sipsec;

    import ietf-inet-types {
        prefix inet;
    }
}

*Note: This is a rather naïve IPSec Service Model*
Let's model the service

The data (1/2)

```
container ipsec {
    presence "Setting IPSec service";
    description "A simple IPSec service";

    list node-list {
        description "Name of customer hub router";
        key node-name;

        leaf node-name {type string;}

        leaf node-ip {
            description "Hub IP address";
            type inet:ipv4-address;
        }
    }
```
Let's model the service

The data (2/2)

list peer-node {
    key peer-name;
    leaf peer-name {type string;}
}

list node-subnets {
    description "IP and mask behind route";
    key "ip inv-mask";
    leaf ip {type inet:ipv4-address;}
    leaf inv-mask {type inet:ipv4-address;}
}

list shared-key {
    key shared-key;
    leaf shared-key {type string;}
    leaf peer-address {type inet:ip-address;}
}

list encryption-protocols{
    key set-name;
    leaf set-name {type string;}
}
Complete Model

module ipsec-service {

namespace "com.example.sipsec";
  prefix sipsec;

  import ietf-inet-types {
    prefix inet;
  }

container ipsec {
  presence "Setting IPSec service";
  description "A simple IPSec service";

  list node-list {
    description "Name of customer hub router";
    key node-name;

    leaf node-name {type string;}

    leaf node-ip {
      description "Hub IP address";
      type inet:ipv4-address;
    }
  }

  list peer-node {
    key peer-name;
    leaf peer-name {type string;}
  }

  list node-subnets {
    description "IP and mask behind route";
    key "ip inv-mask";
    leaf ip {type inet:ipv4-address;}
    leaf inv-mask {type inet:ipv4-address;}
  }

  list shared-key {
    key shared-key;
    leaf shared-key {type string;}
    leaf peer-address {type inet:ip-address;}
  }

  list encryption-protocols{
    key set-name;
    leaf set-name {type string;}
  }
}
Let’s now derive the RESTConf API: The URI’s

/restconf/data/ipsec-service:ipsec/ <list name> / <key value(s)>

module **ipsec-service** {

container **ipsec** {

    list **node-list** {
        key **node-name**;
        leaf **node-name**
    }

    list **peer-node** {
        key **peer-name**;
        leaf **peer-name**
    }

    list **node-subnets** {
        key **ip inv-mask**;
        leaf **ip**
        leaf **inv-mask**
    }

    list **shared-key** {
        key **shared-key**;
        leaf **shared-key** {type string;}
    }

    list **encryption-protocols** {
        key **set-name**;
        leaf **set-name**
    }

}
Let’s add a router node
Pass some JSON data

POST /restconf/data/ipsec-service:ipsec/node-list

module ipsec-service {

container ipsec {

list node-list {

key node-name;

leaf node-name

leaf node-ip {

description "Hub IP address";

type inet:ipv4-address;

}
}

{ "ipsec-service:node-list": [

{ "node-name": "Hub1",
	node-ip": "10.194.128.1"

]

}
}
Questions?
Summary

- Today’s business needs drive programmatic API needs, but serve two mutually dependent consumer groups:
  - Higher level applications
  - Network orchestration and configuration applications
- Network orchestration API needs to be application friendly, allow automated service to configuration mapping:
  - RESTCONF + Yang Models
- Network Device API needs to be robust, standardized, extensible, easy to maintain:
  - NETCONF + Yang Models
Participate in the “My Favorite Speaker” Contest
Promote Your Favorite Speaker and You Could Be a Winner

• Promote your favorite speaker through Twitter and you could win $200 of Cisco Press products (@CiscoPress)

• Send a tweet and include
  • Your favorite speaker’s Twitter handle @wojd
  • Two hashtags: #CLUS #MyFavoriteSpeaker

• You can submit an entry for more than one of your “favorite” speakers

• Don’t forget to follow @CiscoLive and @CiscoPress

• View the official rules at http://bit.ly/CLUSwin
Complete Your Online Session Evaluation

• Give us your feedback to be entered into a Daily Survey Drawing. A daily winner will receive a $750 Amazon gift card.

• Complete your session surveys though the Cisco Live mobile app or your computer on Cisco Live Connect.

Don’t forget: Cisco Live sessions will be available for viewing on-demand after the event at CiscoLive.com/Online
Continue Your Education

• Demos in the Cisco campus
• Walk-in Self-Paced Labs
• Table Topics
• Meet the Engineer 1:1 meetings
• Related sessions
Thank you
TOMORROW starts here.