tail-f

Tutorial: NETCONF and YANG

Presented by Stefan Wallin, Tail-f

Table of Contents

NETCONF and YANG Tutorial Part 1

What is it?

- NETCONF and YANG in 2 minutes

Standards background, motivation and history

- Operators' management requirements captured in RFC 3535
- Implications of transactional management

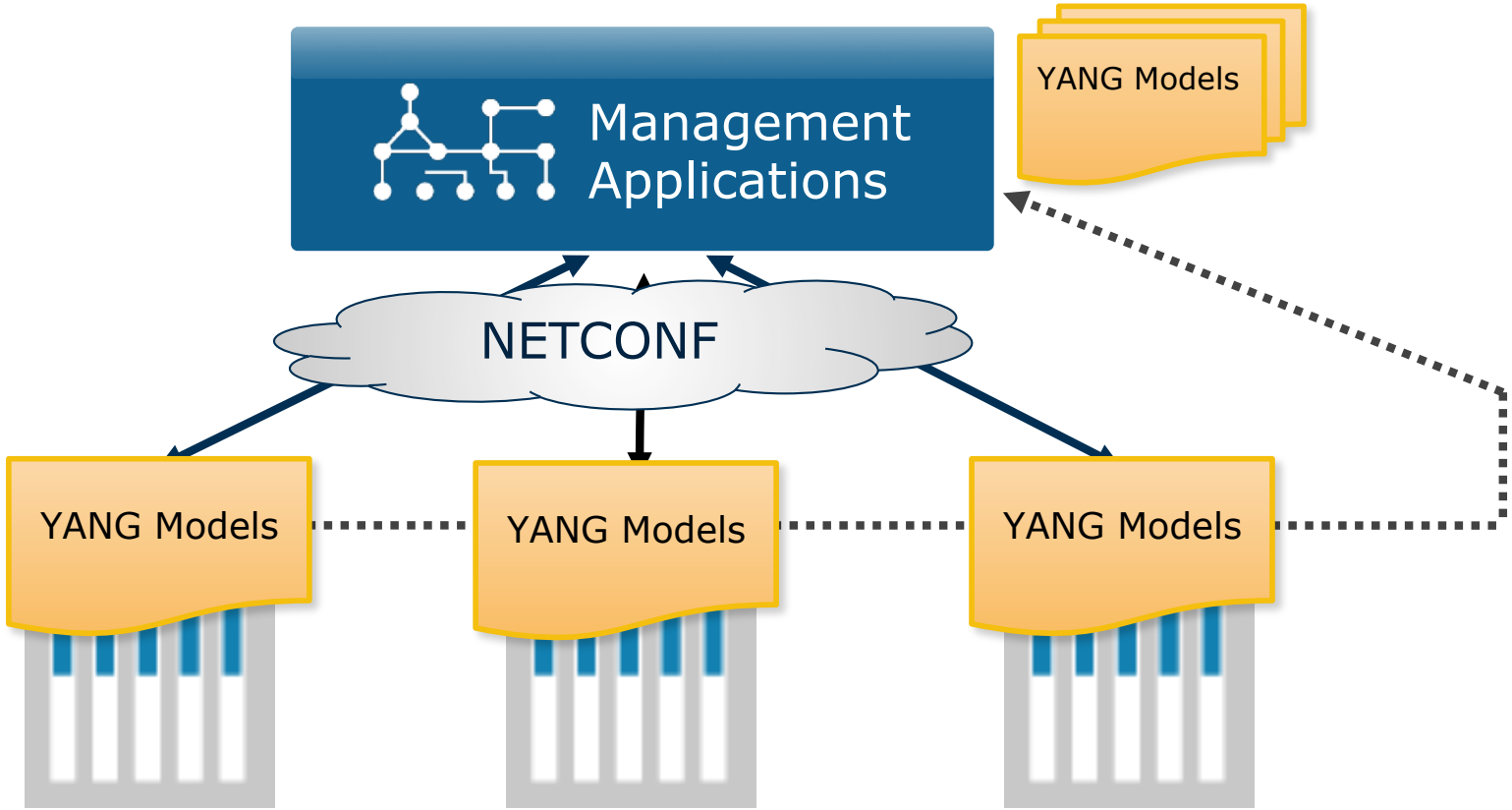
Demonstration of NETCONF and YANG

tail-f

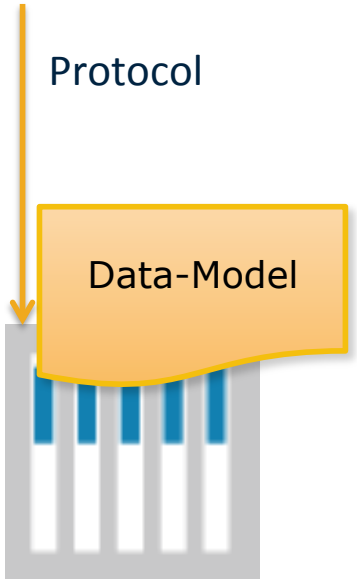
NETCONF and YANG

What is it?

NETCONF and YANG in Context

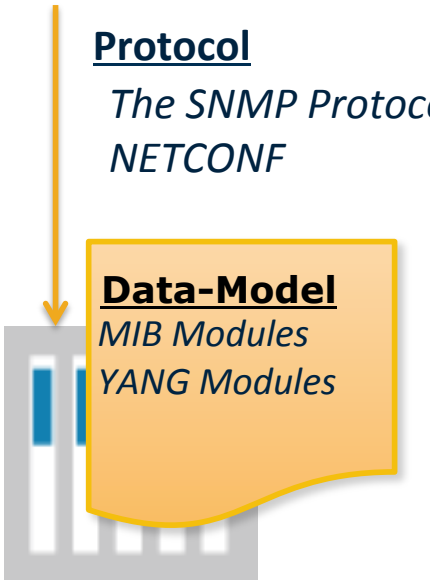


What is a Data-Model? What is a Network Management Protocol?



- Data-Model
  - A data-model explicitly and precisely determines the structure, syntax and semantics of the data…
  - …that is *externally* visible
  - Consistent and complete
- Protocol
  - Remote primitives to view and manipulate the data
  - Encoding of the data as defined by the data-model

Confusion and Comparison



- Data-Models and information Models
  - Information models are for humans
    - Not everything
    - Not always detailed and precise
    - Starting-point for data-model
- Protocol
  - Confusion between domain-specific network management protocols and general RPC mechanisms
    - NETCONF vs. CORBA, SOAP, REST, …

Standards background, motivation and history

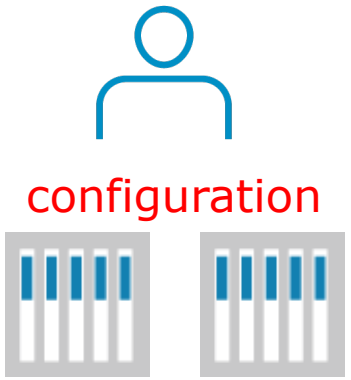
RFC 3535: Operators' problems and requirements on network management

Informational RFC 3535

Abstract

This document provides an overview of a workshop held by the Internet Architecture Board (IAB) on Network Management. The workshop was hosted by CNRI in Reston, VA, USA on June 4 thru June 6, 2002. The goal of the workshop was to continue the important **dialog** started between **network operators** and protocol developers, and to guide the IETFs focus on future work regarding network management.

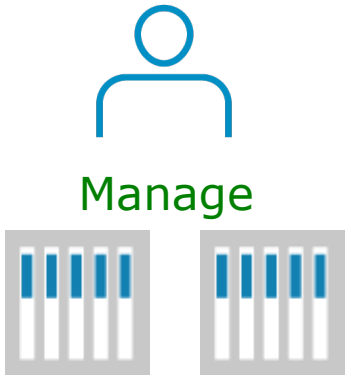
- SNMP had failed
  - For configuration, that is
  - Extensive use in fault handling and monitoring
- CLI scripting
  - "Market share" 70%+



Operator Requirement #1/14

1. **Ease of use** is a key requirement for any network management technology from the operators point of view.

Maybe not assume integrators and software developers for any addition or change



Operator Requirement #2-3/14

2. It is necessary to make a **clear distinction** between **configuration data**, data that describes **operational state and statistics**.

3. It is required to be able to **fetch separately configuration data**, operational state data, and statistics from devices, and to be able to compare these between devices.

- Clearly separating configuration
- Ability to compare across devices

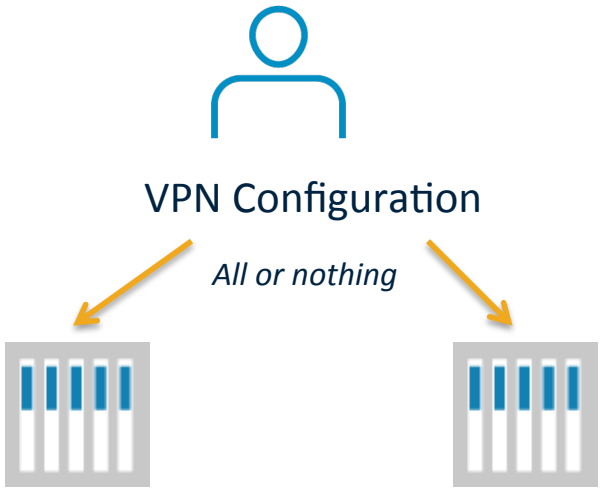
```
$show running-config
```

Operator Requirement #4-5/14

4. It is necessary to enable operators to concentrate on the **configuration of the network** as a whole rather than individual devices.

5. Support for **configuration transactions** across a number of devices would significantly simplify network configuration management.

- Service and Network management, not only device management
- Network wide transactions



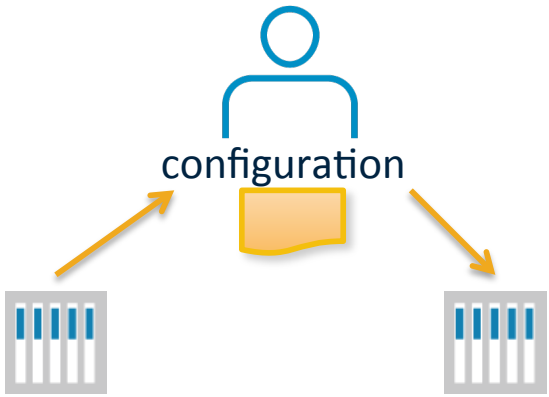
Operator Requirement #6-7/14

6. Given configuration A and configuration B, it should be possible to generate the **operations necessary to get from A to B** with minimal state changes and effects on network and systems. It is important to minimize the impact caused by configuration changes.

7. A mechanism to dump and restore configurations is a primitive operation needed by operators. Standards for **pulling and pushing configurations** from/to devices are desirable.

- Devices figure out ordering
- No unnecessary changes
- Finally: backup/restore of configuration

The litmus-test of a management interface

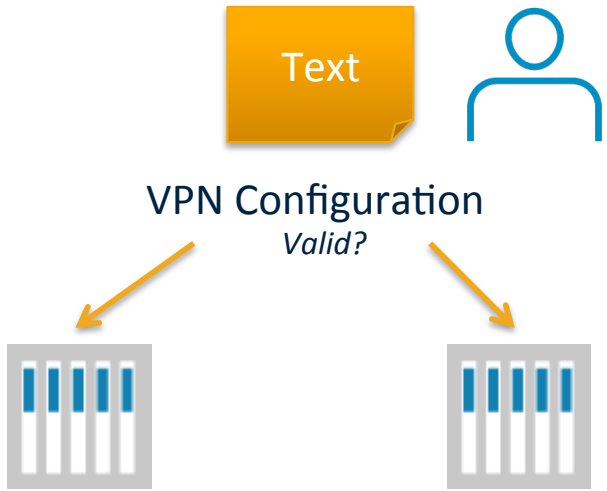


Operator Requirement #8, 10/14

8. It must be easy to do **consistency** checks of configurations over time and between the ends of a link in order to determine the changes between two configurations and whether those configurations are consistent.

10. It is highly desirable that **text** processing tools such as diff, and version management tools such as RCS or CVS, can be used to process configurations, which implies that devices should not arbitrarily reorder data such as access control lists.

- Validation of configuration
- Validation at network level
- Text based configuration



Operator Requirement #9/14

9. Network wide configurations are typically stored in central master databases and transformed into formats that can be pushed to devices, either by generating sequences of CLI commands or complete configuration files that are pushed to devices. There is no **common database schema** …, although the models used by various operators are probably very similar.

It is desirable to extract, document, and standardize the common parts of these network wide configuration database schemas.

- Standardized data models

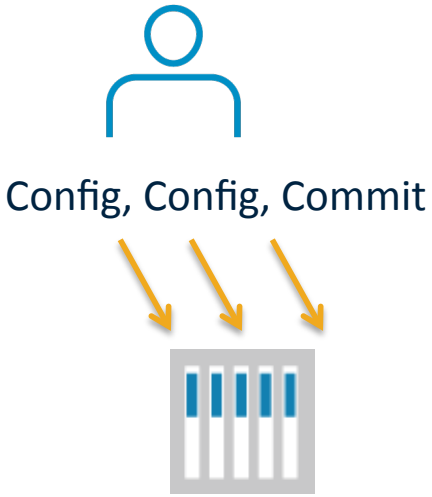
Interfaces Data-Model

Operator Requirement #13/14

13. It is important to distinguish between the distribution of configurations and the activation of a certain configuration.

Devices should be able to hold multiple configurations.

- Support for multiple configuration sets
- Delayed, orchestrated activation



Operator Requirement #11,12,14/14

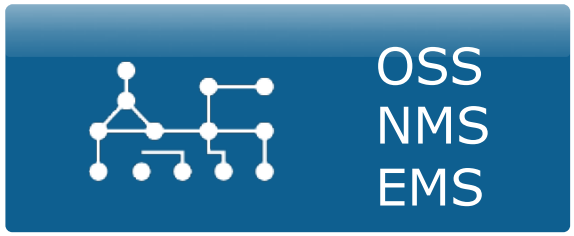
11. … Typical requirements are a role-based access control model and the principle of least privilege, where a user can be given only the minimum access necessary to perform a required task.

12. It must be possible to do consistency checks of access control lists across devices.

14. SNMP access control is data-oriented, while CLI access control is usually command (task) oriented. … As such, it is a requirement to support both data-oriented and task-oriented access control

- Role-Based Access Control (RBAC)
  - Data oriented
  - Task oriented

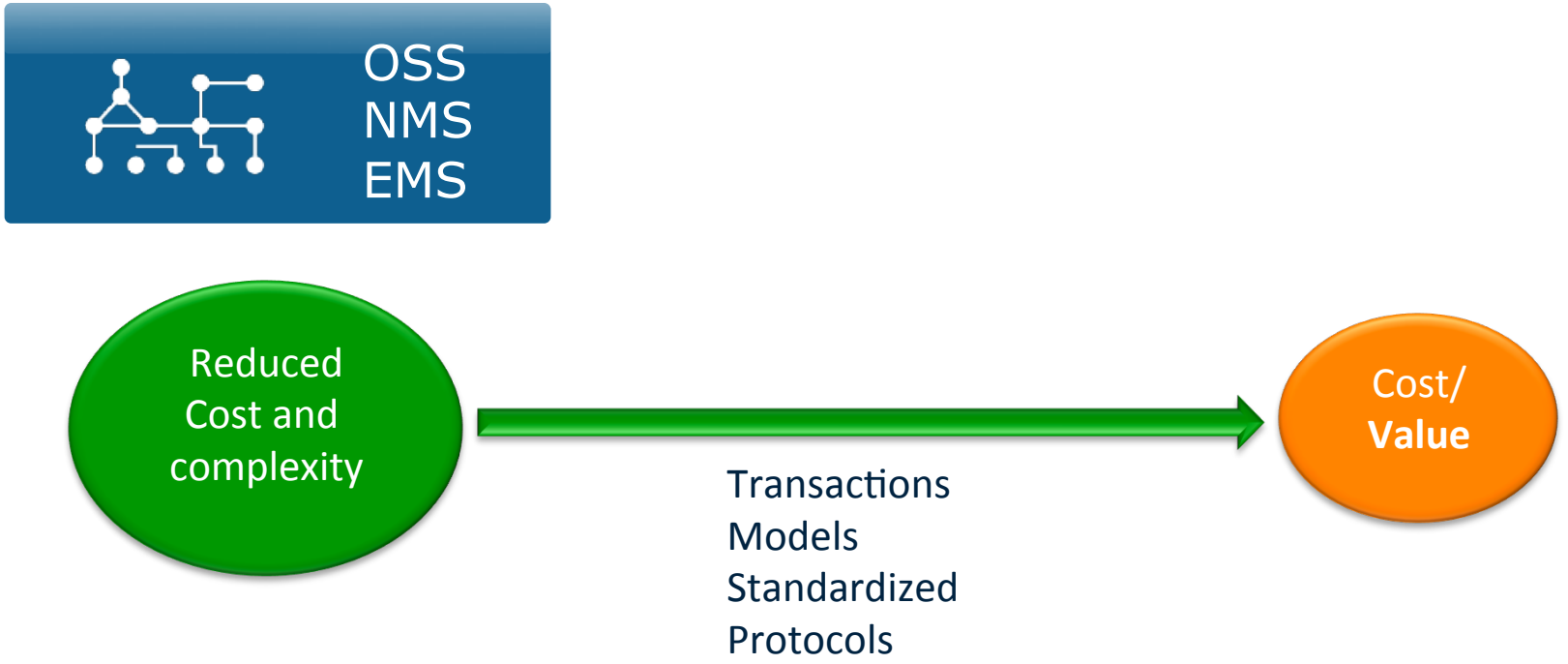
Implications of RFC 3535, legacy situation



Cost and complexity

- No well-defined protocols and data-models
- Lack of atomicity
- Ordering problem

Implications of RFC 3535, with transactions



NETCONF was designed to conform to RFC 3535.

Today many operators require NETCONF and YANG in devices.

NETCONF makes a difference on the bottom line.

tail-f

Introduction to NETCONF

What makes NETCONF/YANG different?

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

"RESTConf"

What makes NETCONF/YANG different?

SNMP

- GET
- GET-NEXT
- SET
- TRAP
- …

… so what?

NETCONF

- <get-config>
- <edit-config>
- <copy-config>
- <delete-config>
- <get>
- <lock>
- …

… same same?

tail-f

What makes NETCONF/YANG different?

This is where the difference is:
In the supported use cases!

Use Case	SNMP	NETCONF
Get collection of status fields	Yes	Yes. Bulk xfer up to 10x faster. Really.
Set collection of configuration fields	Yes, up to 64kB	Yes
Set configuration fields in transaction	No	Yes
Transactions across multiple network elements	No	Yes
Invoke administrative actions	Well…	Yes
Send event notifications	Yes	Yes, connected
Backup and restore configuration	Usually not	Yes
Secure protocol	v3 is fair	Yes
Test configuration before final commit	No	Yes

The Meaning of Transactions

The four properties that define a transaction: ACID

Atomicity

- Transactions are indivisible, all-or-nothing

Consistency

- Transactions are all-at-once
- There is no internal order inside a transaction, it is a **set** of changes, **not a sequence**
- Implies that { create A, create B } and { create B, create A } are identical
- Implies that a system behaving differently with respect to the sequence is not transactional

Independence

- Parallel transactions do not interfere with each other
- Transactions appear to happen always-in-sequence

Durability

- Committed data always-sticks, i.e. remains in the system even in the case of a fail-over, power failure, restart, etc

The Meaning of Transactions

- Consider transaction A
  - Add interface eth5
  - Add route 55.66.0.0/24 over interface eth5
- Transactions are all-at-once, there is no internal ordering
- Transaction A is therefore equivalent to
  - Add route 55.66.0.0/24 over interface eth5
  - Add interface eth5

Obviously, the order matters in the execution.

But it is not the manager's concern in a transactional system.

The Meaning of Transactions

Backup

- Read the configuration
  - Manager does not need to know which elements are configuration
- Save result to a file
  - Human readable XML file
  - Use diff and other XML processing tools
  - Edit file, if desired

Restore

- Send the saved configuration
  - No need to sort data
  - All-or-nothing semantics
  - No PDU size limit

The Meaning of Transactions

Service Activation

- Operator creates new service in OSS GUI
  - IPTV service, HD quality
- OSS computes configuration changes to send to network
  - Some IPTV server edits
  - Three routers
  - Two firewalls
  - One billing machine
- OSS sends configuration change to all concerned devices in a network wide transaction
  - No need to sort data
  - All-or-nothing semantics across all devices
  - Each device validates
- Optionally test service
  - Confirm or roll back

Network-wide Transactions is the most important leap in network management technology since SNMP.

The error recovery and sequencing tasks are removed from the manager side.

This is usually more than half the cost in a mature system; more than the entire cost of the managed devices.

Who ?

NETCONF

- Phil Shafer, Rob Enns
  - Juniper
- Jürgen Schönwälder
  - Jacobs University
- Martin Björklund
  - Tail-f
- Andy Bierman
  - Yumaworks
- Ken Crozier Eliot Lear
  - Cisco Systems
- Ted Goddard
  - IceSoft
- Steve Waldbusser
- Margaret Wasserman
  - Painless Security, LLC

YANG

- Phil Shafer
  - Juniper, XML
- Jürgen Schönwälder
  - Jacobs University, SNMP SMIng
- Martin Björklund
  - Tail-f, Vendor specific
- David Partain
  - Ericsson, made it happen

tail-f

NETCONF Overview and Examples

NETCONF Transport

NETCONF messages are encoded in XML

- Each message is framed by
  - NETCONF 1.0: a character sequence]]>]]>
  - NETCONF 1.1: a line with the number of characters to read in ASCII

NETCONF messages are encrypted by SSH

- NETCONF over SOAP, BEEP (both now deprecated) and TLS are also defined, but not used
- SSH provides authentication, integrity and confidentiality

NETCONF is connected oriented using TCP

- No need for manager to request resends
- Efficient use of the medium

NETCONF Extensibility

When a NETCONF Manager connects to a NETCONF Server (Device), they say `<hello>`

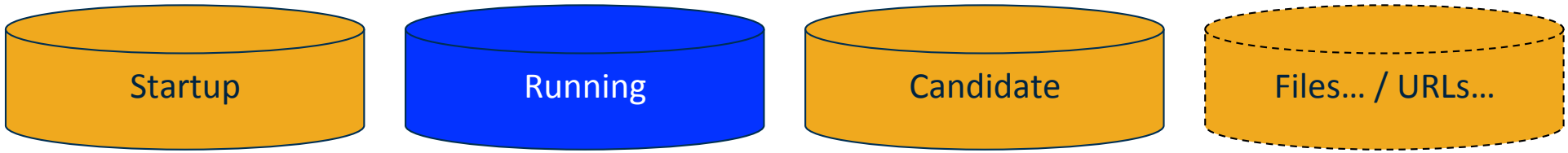
The contents of the <hello> message declares which NETCONF Capabilities each party is capable of.

- 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 <hello>, the manager will know which operations it can submit to the Device.

Extensions go in separate XML namespaces, which makes it easier to build backwards and forwards compatible management applications.

NETCONF Configuration Data Stores



- 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 *(:writable-running)*
  - Need to copy from other stores if not directly writable

NETCONF Transactions, Network-wide Transactions

NETCONF allows a Manager to send down a set of configuration changes, or an entirely new configuration, in a single `<edit-config>` transaction.

When doing so, the Manager does not need to

- Figure out which order to send down the configuration changes in. All different sequences are treated equal.
- Recover if the transaction fails. If the transaction was unsuccessful because of
  - inconsistency in the configuration
  - an out of memory condition
  - any other reason

  … none of the transaction content has been activated.

The transaction did not roll back. It was simply never activated.

NETCONF Transactions, Network-wide 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 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 Example Configuration Sequence

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

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

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

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.
  <commit>
    <confirmed/>
  </commit>
</rpc>
```

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

NETCONF RFC6241 Optional Capabilities

:writable-running

:candidate

:confirmed-commit

:rollback-on-error

:validate

:startup

:url (scheme=http, ftp, file, …)

:xpath (filters)

Non-base NETCONF Capabilities

`:notification`, `:interleave` *(RFC 5277)*

`:partial-lock` *(RFC 5717)*

`:with-defaults` *(RFC 6243)*

`:ietf-netconf-monitoring` *(RFC 6022)*

And you can define your own, like

`:actions` *(tail-f)*

`:inactive` *(tail-f)*

NETCONF Operations

NETCONF <hello> Operation

- Capabilities exchange
- Data model ID exchange
- Encoding
- Framing
  - NETCONF 1.0 EOM,]]>]]>
  - NETCONF 1.1 Chunk Framing

```
<?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>
  </capabilities>
</hello>
```

NETCONF <hello> Operation

```
<?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</capabilit
<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>
<capability>urn:ietf:params:xml:ns:yang:ietf-inet-types?
revision=2010-09-24&amp;module=ietf-inet-types</capability>
</capabilities>
<session-id>5</session-id>
</hello>
```

NETCONF <get-config> Operation

- Sub-tree filtering
- XPATH filtering

```
<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>
```

NETCONF <get-config> Operation

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="1">
  <data>
    <aaa xmlns="http://tail-f.com/ns/aaa/1.1">
      <authentication>
        <users>
          <user>
            <name>admin</name>
            <uid>9000</uid>
            <gid>0</gid>
            <password>$1$3ZHhR6Ow$acznsyClFc0keo3B3BVjx/</password>
            <ssh_keydir>/var/confd/homes/admin/.ssh</ssh_keydir>
            <homedir>/var/confd/homes/admin</homedir>
          </user>
          <user>
            <name>oper</name>
          …
        </users>
      </authentication>
    </aaa>
  </data>
</rpc-reply>
```

NETCONF <edit-config> Operation

```
<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"
            xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1">
        <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>
```

NETCONF <edit-config> Operation

```
nc:test-option (:validate)
    test-then-set (default)
    set
    test-only
nc:error-option
    stop-on-error (default)
    continue-on-error
    rollback-on-error
    (:rollback-on-error)
```

```
nc:operation
    merge
    replace
    create
    delete
    remove (:base:1.1)
```

Error if item to delete does not exist

Ok if item to delete does not exist

NETCONF <copy-config> Operation

- Copy configuration data between stores or URLs

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <copy-config>
    <target><running/></target>
    <source>
      <url>https://user@example.com:passphrase/cfg/new.txt
      </url>
    </source>
  </copy-config>
</rpc>
```

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <ok/>
</rpc-reply>
```

NETCONF <delete-config> Operation

- Delete a complete data store (not running)

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <delete-config>
    <target>
      <startup/>
    </target>
  </delete-config>
</rpc>
```

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <ok/>
</rpc-reply>
```

NETCONF <lock>, <unlock> Operation

- Lock/unlock a complete data store

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <lock>
    <target>
      <candidate/>
    </target>
  </lock>
</rpc>
```

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <ok/>
</rpc-reply>
```

NETCONF <get> Operation

- Read configuration **and** status

```
<rpc message-id="101"
xmlns="urn:ietf:param
<get>
  <filter type="subtree">
    <top xmlns="http://example.com/ns/dhc
      <interfaces>
        <interface>
          <ifName>eth0</ifName>
        </interface>
      </interfaces>
    </top>
  </filter>
</get>
```

```
<rpc-reply message-id="101" xmlns="urn:ie
  <data>
    <top xmlns="http://example.com/ns/dhc
      <interfaces>
        <interface>
          <ifName>eth0</ifName>
          <ifInOctets>45621</ifInOctets>
          <ifOutOctets>774344</ifOutOctet
        </interface>
      </interfaces>
    </top>
  </data>
</rpc-reply>
```

NETCONF <close-session> Operation

- Polite way of disconnecting

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <close-session/>
</rpc>
```

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <ok/>
</rpc-reply>
```

NETCONF <kill-session> Operation

- Not so polite way of disconnecting another session

  Releases any locks, aborts any confirmed commit related to session

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <kill-session>
    <session-id>17</session-id>
  </kill-session>
</rpc>
```

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <ok/>
</rpc-reply>
```

Additional NETCONF operations by capabilities

- `<commit>, <discard-changes>` *`(:candidate)`*
- `<validate>` *`(:validate)`*
  - Copy candidate to running
  - Discard changes in candidate (copy running to candidate)
- `<create-subscription>` *`(:notification)`*
- `<partial-lock>, <partial-unlock>` *`(:partial-lock)`*
- `<commit>, <cancel-commit>` *`(:commit)`*
- `<get-schema>` *`(:ietf-netconf-monitoring)`*

NETCONF <partial-lock> Operation

- Lock parts of the configuration in the running store

```
<nc:rpc xmlns="…partial-lock"
        xmlns:nc="…netconf"
        message-id="135">
  <partial-lock>
    <select xmlns:rte="…">
      /rte:routing/rte:virtualRouter
[rte:routerName='router1']
    </select>
    <select xmlns:if="…">
if:interfaces/if:interface[if:id='eth1']
    </select>
  </partial-lock>
</nc:rpc>
```

```
<nc:rpc-reply xmlns:nc="…net"
     xmlns="…partial-lock"
     message-id="135">
  <lock-id>127</lock-id>
  <locked-node xmlns:rte="…"
    /rte:routing/rte:virtualRouter
[rte:routerName='router1']
  </locked-node>
  <locked-node xmlns:if="…"
/if:interfaces/if:interface[if:id='eth1']
  </locked-node>
</nc:rpc-reply>
```

Event Notifications

- What is right for you?
  - SNMP Traps over UDP
  - SYSLOG Messages over UDP
  - NETCONF Notifications over SSH
  - Define your own

ⓘ www.rfc-editor.org/rfc/rfc5277.txt

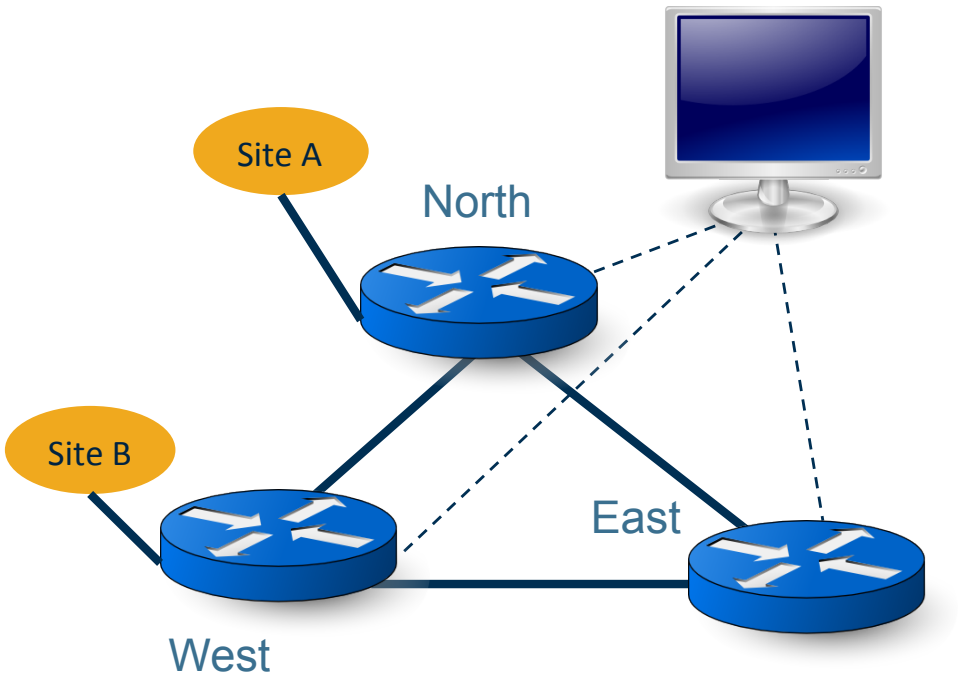
- NETCONF
  - Requires open connection
  - Supports replay
- SNMP
  - Much used, works ok
  - Limited payload
  - Connection-less
- SYSLOG
  - Lacks standard format

tail-f

Example:

VPN provisioning using NETCONF Network-wide Transactions

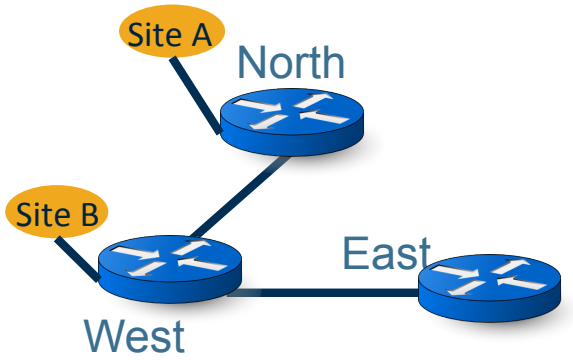
VPN Scenario



- An operator owns a network of routers and other equipment from different vendors
- They have a management station connected to all the devices in the network to provision and monitor services
- Now, we need to set up a VPN between two customer sites
- There is no point what so ever to make any changes on any device unless all changes succeed
- We need a Network-wide Transaction

Hello

- Exchange capabilities




```
>>>>> Router-West (Sun Nov 15 14:41:25 CET 2009)
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
  </capabilities>
</hello>
```

```
<<<< Router-West (Sun Nov 15 14:41:25 CET 2009)
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
    <capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
    <capability>http://tail-f.com/ns/aaa/1.1</capability>
    <capability>http://tail-f.com/ns/example/quagga/1.0</capability>
  </capabilities>
  <session-id>4</session-id>
</hello>
```

Clear the candidate data stores

```
>>>>> Router-West (Sun Nov 15 15:24:32 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="1">
   <nc:discard-changes></nc:discard-changes>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="1">
   <ok></ok>
</rpc-reply>
```

Lock the candidate data stores

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="2">
  <nc:lock>
    <nc:target>
       <nc:candidate></nc:candidate>
    </nc:target>
  </nc:lock>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="2">
   <ok></ok>
</rpc-reply>
```

Lock the running data stores

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="3">
  <nc:lock>
    <nc:target>
      <nc:running></nc:running>
    </nc:target>
  </nc:lock>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="3">
   <ok></ok>
</rpc-reply>
```

Copy running to candidates (logically)

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="4">
  <nc:copy-config>
    <nc:target>
      <nc:candidate></nc:candidate>
    </nc:target>
    <nc:source>
      <nc:running></nc:running>
    </nc:source>
  </nc:copy-config>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="4">
   <ok></ok>
</rpc-reply>
```

Edit the candidates

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="5">
  <nc:edit-config>
    <nc:target><nc:candidate></nc:candidate></nc:target>
    <nc:config>
      <quagga:system xmlns:quagga="http://tail-f.com/ns/example/quagga
      <quagga:vpn>
        <quagga:ipsec>
          <quagga:tunnel>
            <quagga:name>volvo-0</quagga:name>
            <quagga:local-endpoint>10.7.7.4</quagga:local-endpoint>
            <quagga:local-net>33.44.55.0</quagga:local-net>
            <quagga:local-net-mask>255.255.255.0</quagga:local-net-mas
            <quagga:remote-endpoint>10.3.4.1</quagga:remote-endpoint>
            <quagga:remote-net>62.34.65.0</quagga:remote-net>
            <quagga:pre-shared-key>ford</quagga:pre-
            <quagga:encryption-algo>default
            <quagga:hash-algo>defau
```

Validate candidates

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="6">
   <nc:validate>
          <nc:source>
                 <nc:candidate></nc:candidate>
          </nc:source>
   </nc:validate>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="6">
   <ok></ok>
</rpc-reply>
```

Commit candidates to running

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="7">
   <nc:commit></nc:commit>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="7">
   <ok></ok>
</rpc-reply>
```

Unlock candidates

```
>>>>> Router-West (Sun Nov 15 15:24:33 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="8">
   <nc:unlock>
          <nc:target>
                <nc:candidate></nc:candidate>
          </nc:target>
   </nc:unlock>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:24:33 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="8">
   <ok></ok>
</rpc-reply>
```

Using confirmed-commit

- Now do the same thing again, but instead of commit…

```
>>>>> Router-West (Sun Nov 15 15:29:19 CET 2009)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" nc:message-id="16">
   <nc:commit>
          <nc:confirmed></nc:confirmed>
          <nc:confirm-timeout>120</nc:confirm-timeout>
   </nc:commit>
</nc:rpc>
```

```
<<<< Router-West (Sun Nov 15 15:29:19 CET 2009)
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.1" message-id="16">
   <ok></ok>
</rpc-reply>
```

Disaster happens

- One of the devices disconnected
- The management station disconnects all the rest
- They all roll back to the previous configuration
- The management station reconnects

```
>>>>> Router-West (Sun Nov 15 15:30:22 CET 2009)
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
   <capabilities>
         <capability>urn:ietf:params:netconf:base:1.1</capability>
   </capabilities>
</hello>
```

```
<<<< Router-West (Sun Nov 15 15:30:22 CET 2009)
<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.
```

Common NETCONF Use Cases

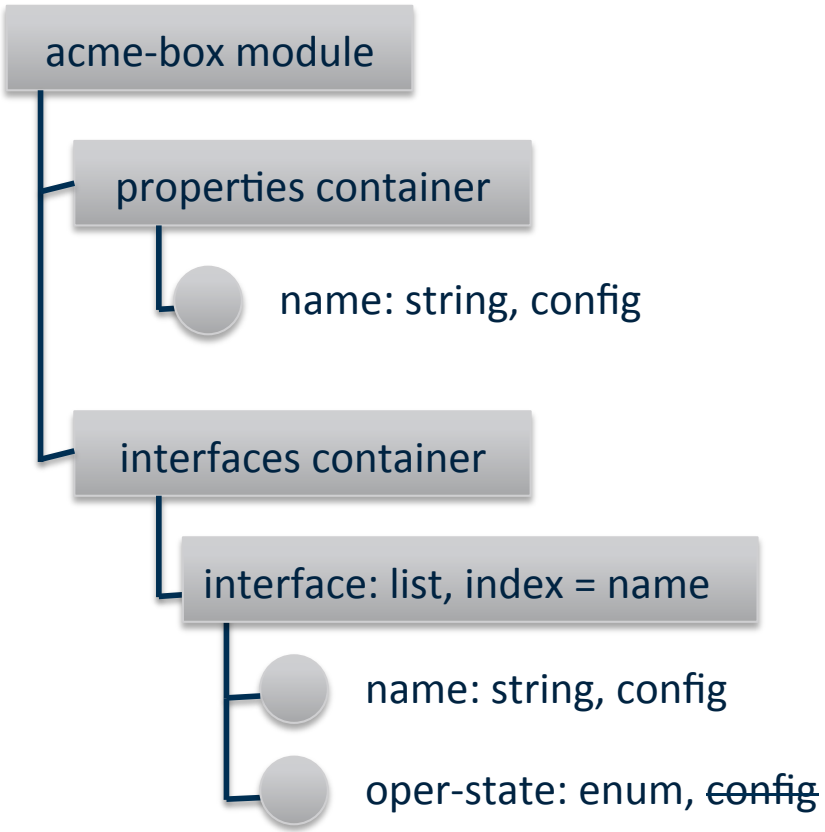
- Network-wide transactions
- Applying and testing a configuration
- Testing and rejecting a configuration
- Rollback when device goes down
- Transactions requiring all devices to be up
- Backlogging transactions
- Synchronizing

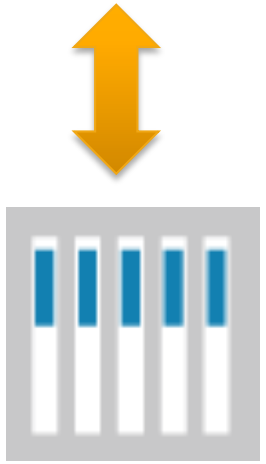
tail-f

YANG Overview and Examples

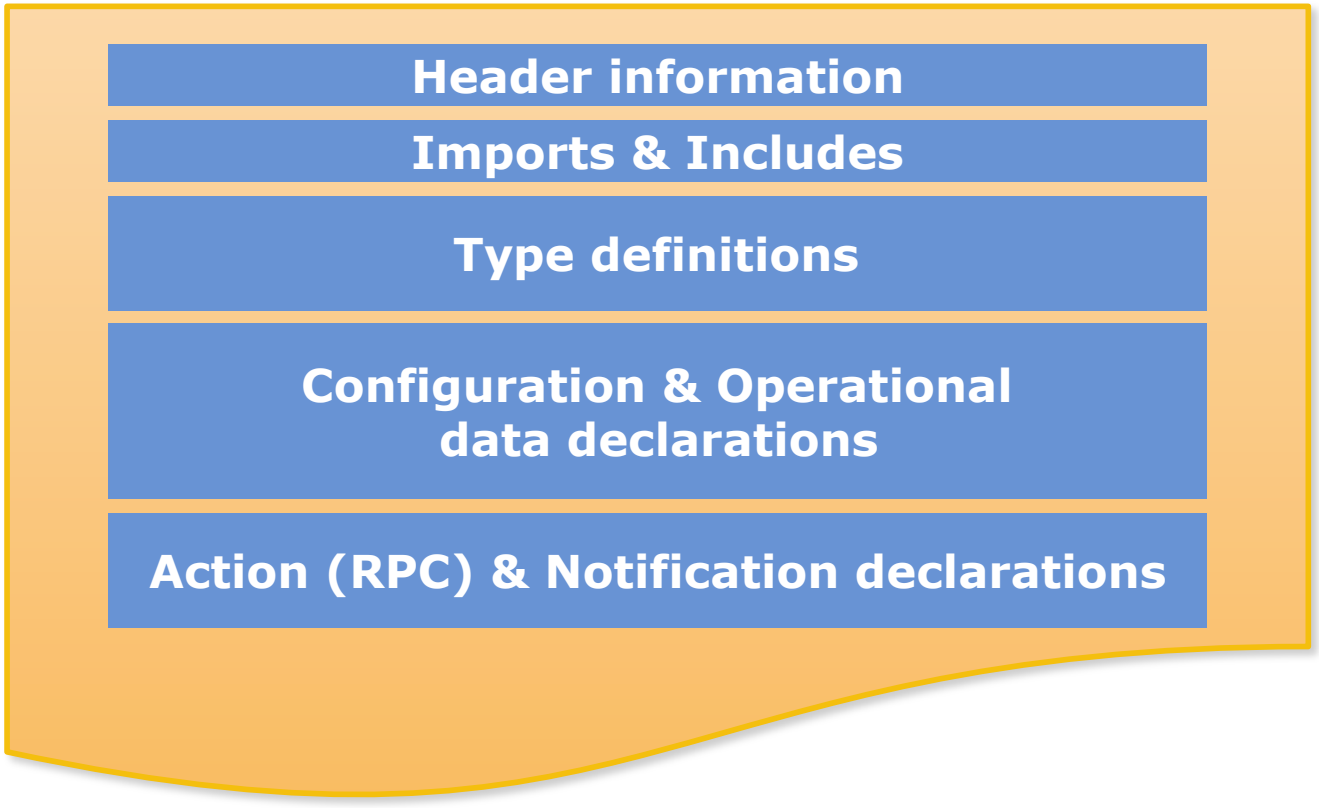
YANG ?

- Data modeling language
  - Configuration data
  - State data
- Tree structure
- Data and Types

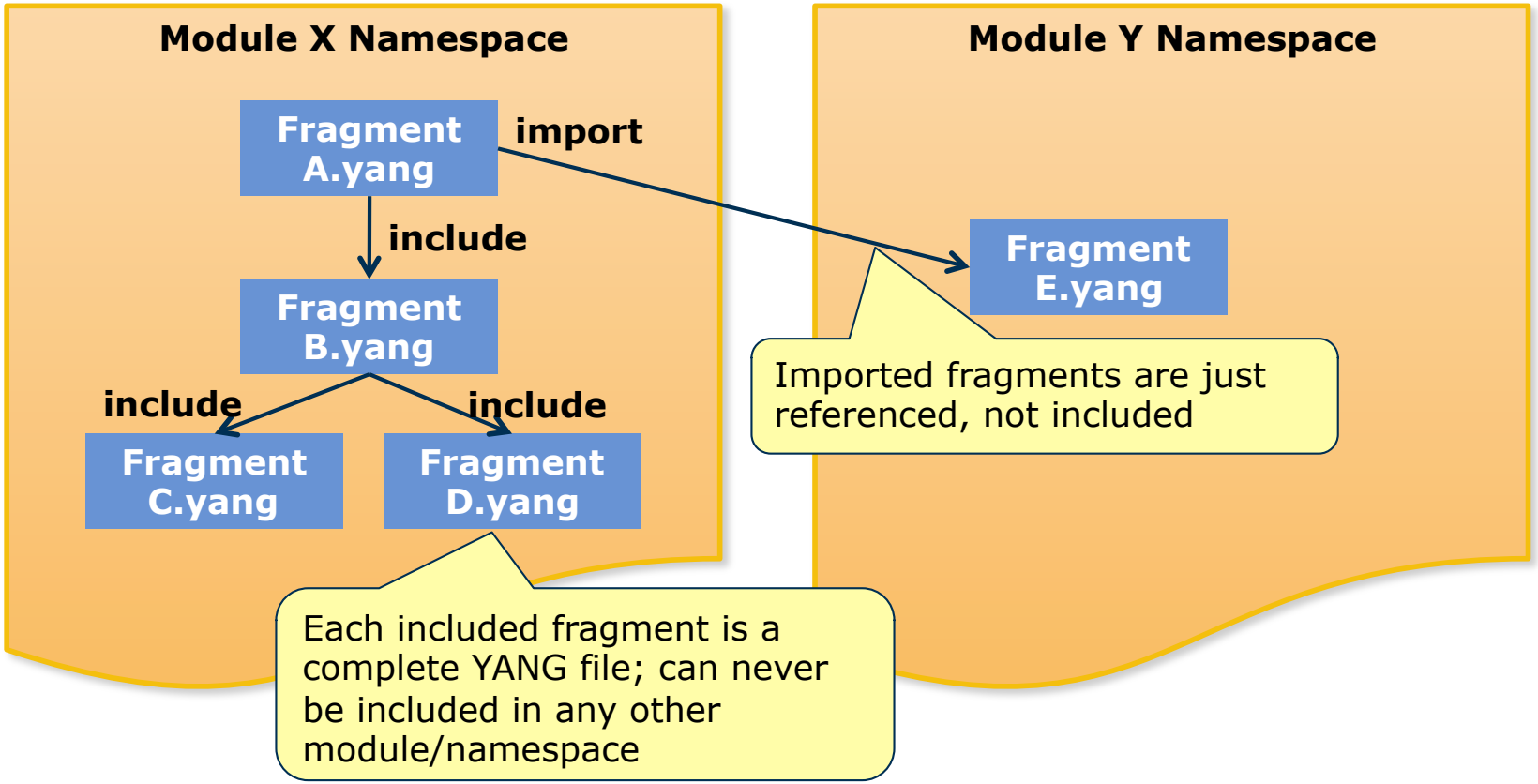




YANG Module Contents



Imports & Includes



YANG Header

```
module acme-module {
  namespace "http://acme.example.com/module";
  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.";
  }
```

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 {
    …
  }
}
```

Each submodule belongs to one specific main module

Attention: The submodule cannot reference definitions in main module

YANG Types

YANG Base Types

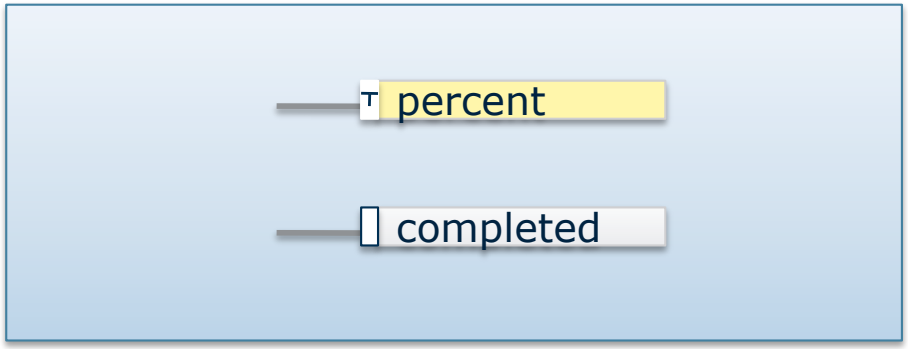
- Most YANG elements have a data type
- Type may be a base type or derived type
  - Derived types may be simple typedefs or groupings (structures)
  - There are 20+ base types to start with

Type Name	Meaning
`int8/16/32/64`	Integer
`uint8/16/32/64`	Unsigned integer
`decimal64`	Non-integer
`string`	Unicode string
`enumeration`	Set of alternatives
`boolean`	True or false
`bits`	Boolean array
`binary`	Binary BLOB
`leafref`	Reference “pointer”
`identityref`	Unique identity
`empty`	No value, void
	…and more

Typedef Statement

Defines a new simple 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]*";
  }
}
```

Union Statement

A value that represents one of its member types

```
typedef threshold {
  description "Threshold value in percent";
  type union {
    type uint16 {
      range "0 .. 100";
    }
    type enumeration {
      enum disabled {
        description "No threshold";
      }
    }
  }
}
```

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;
```

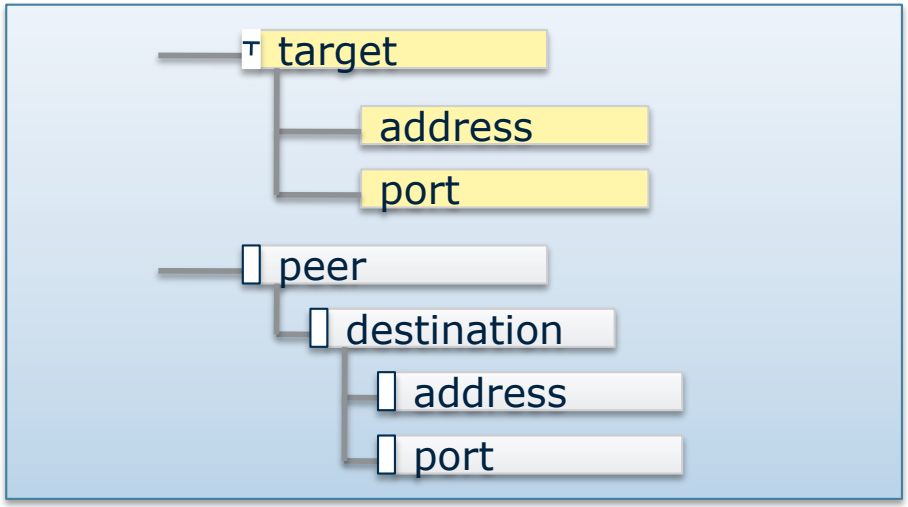
counter32/64	ipv4-address
gauge32/64	ipv6-address
object-identifier	ip-prefix
date-and-time	ipv4-prefix
timeticks	ipv6-prefix
timestamp	domain-name
phys-address	uri
ip-version	mac-address
flow-label	bridgeid
port-number	vlanid
ip-address	… and more

ⓘ 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;
  }
}
```



Grouping Statement with Refine

Groupings may be refined when used

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

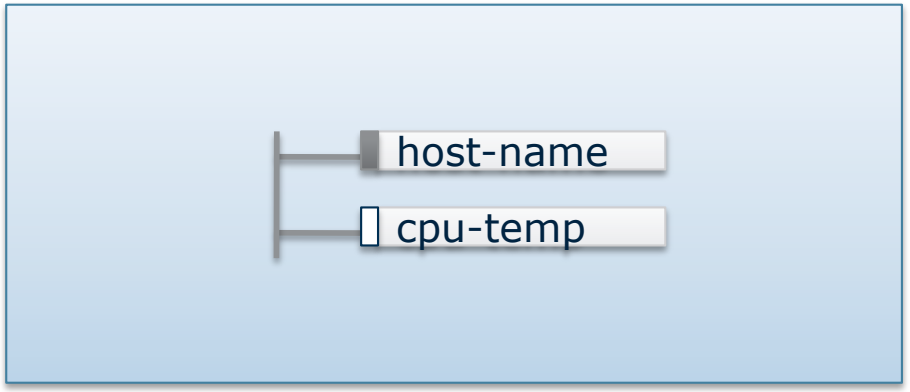
```
container servers {
  container http {
    uses target {
      refine port {
        default 80;
      }
    }
  }
}
```

YANG Data Definitions

Leaf Statement

Holds a single value of a particular type

Has no children



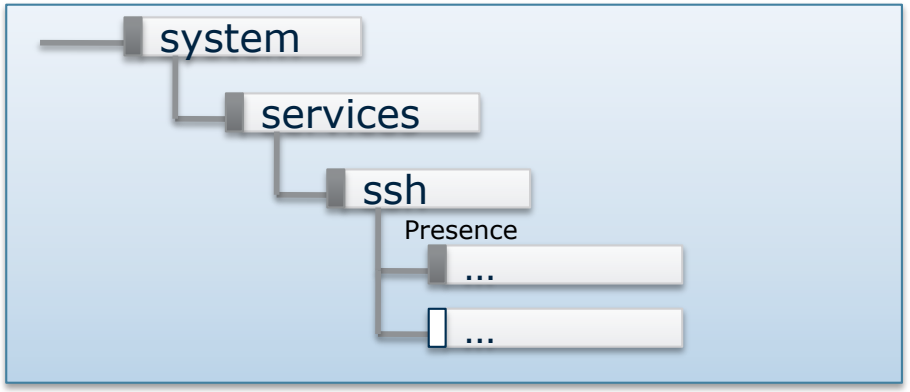

```
leaf host-name {
  type string;
  mandatory true;
  config true;
  description "Hostname for this system";
}
leaf cpu-temp {
  type int32;
  units degrees-celsius;
  config false;
  description ”Current temperature in CPU";
}
```

Attributes for leaf

`config`	Whether this leaf is a configurable value (“true”) or operational value (“false”). Inherited from parent container if not specified
`default`	Specifies default value for this leaf. Implies that leaf is optional
`mandatory`	Whether the leaf is mandatory (“true”) or optional (“false”)
`must`	XPath constraint that will be enforced for this leaf
`type`	The data type (and range etc) of this leaf
`when`	Conditional leaf, only present if XPath expression is true
`description`	Human readable definition and help text for this leaf
`reference`	Human readable reference to some other element or spec
`units`	Human readable unit specification (e.g. Hz, MB/s, ℉)
`status`	Whether this leaf is “current”, “deprecated” or “obsolete”

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...
    }
  }
}
```

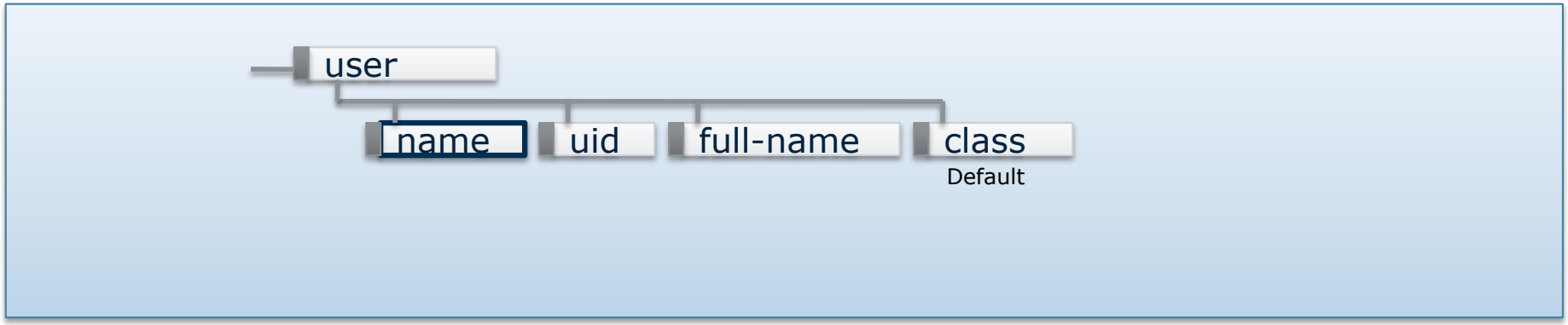
Leaf-list Statement

Holds multiple values of a particular type

Has no children


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

List Statement

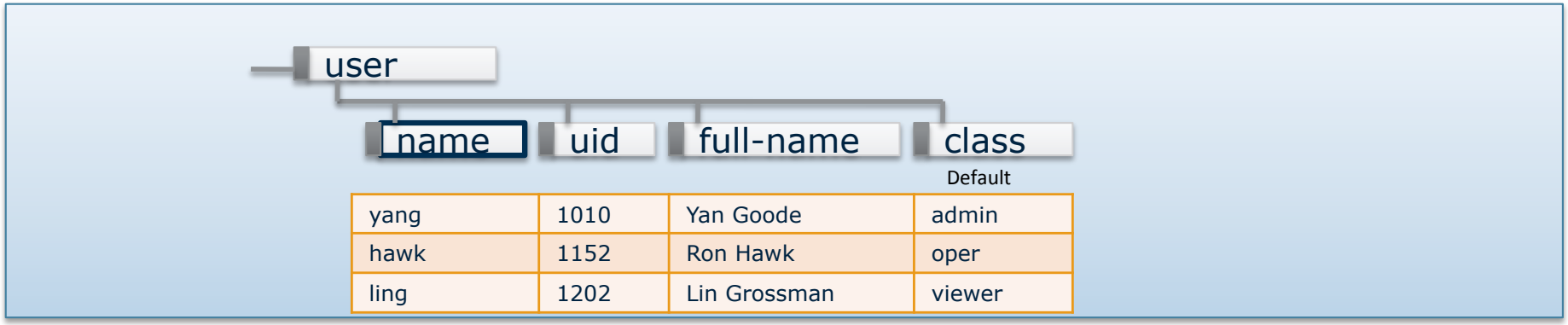



```
list user {
  key name;
  leaf name {
    type string;
  }
  leaf uid {
    type uint32;
  }
  leaf full-name {
    type string;
  }
  leaf class {
    type string;
    default viewer;
  }
}
```

Attributes for list and leaf-list

`max-elements`	Max number of elements in list. If max-elements is not specified, there is no upper limit, i.e. "unbounded"
`min-elements`	Min number of elements in list. If min-elements is not specified, there is no lower limit, i.e. 0
`ordered-by`	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.

Keys

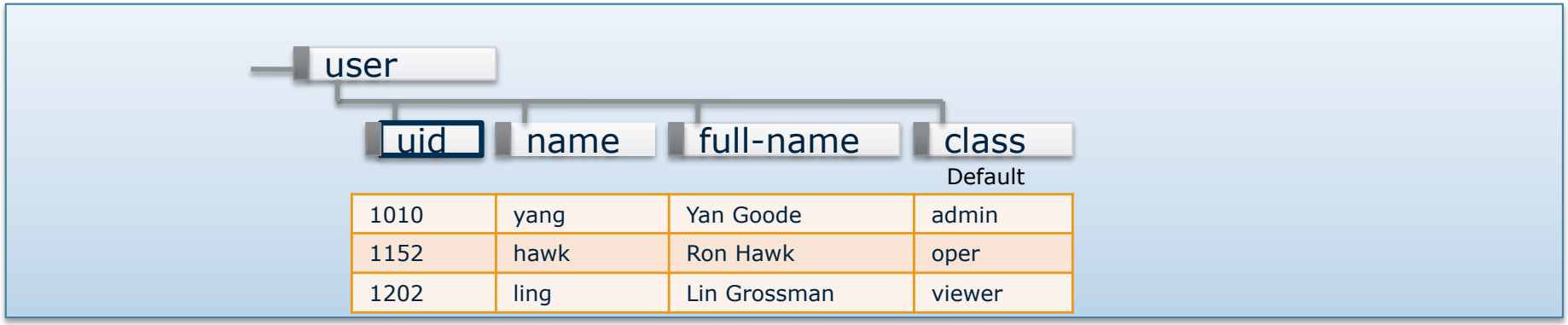


The key field is used to specify which row we're talking about.

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
```

Keys

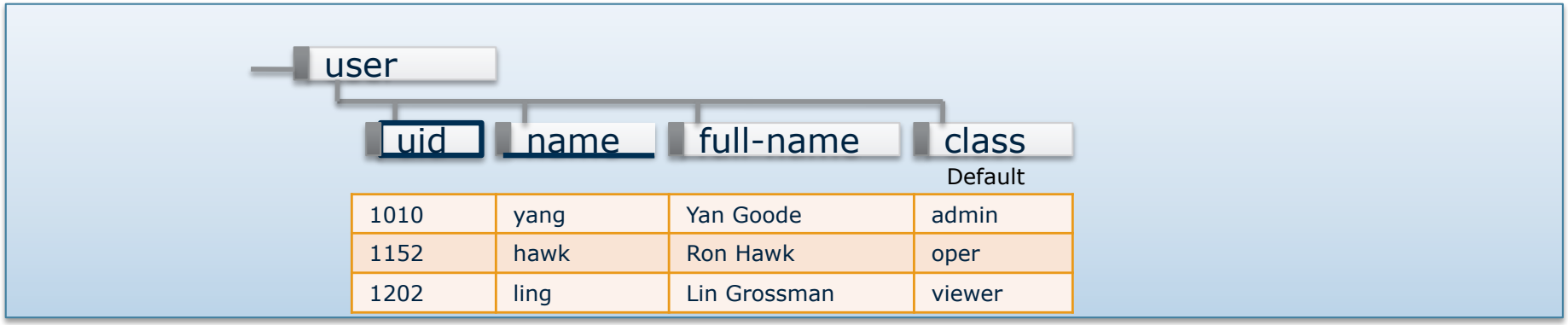


1010	yang	Yan Goode	admin
1152	hawk	Ron Hawk	oper
1202	ling	Lin Grossman	viewer

If we want, we could select the uid to be key instead.

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

/user[uid='1202']/class = viewer
```

Unique Statement



1010	yang	Yan Goode	admin
1152	hawk	Ron Hawk	oper
1202	ling	Lin Grossman	viewer

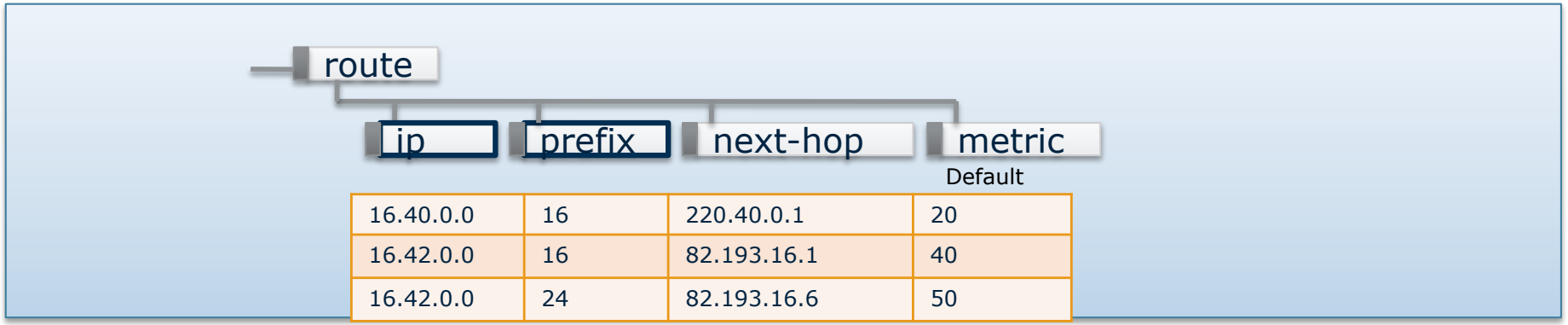
Non- key fields can also be declared unique.

Multiple fields can be declared unique separately or in combination

```
list user {
  key uid;
  unique name;
…
```

No two rows above can have same uid, nor name

Multiple keys



Multiple key fields are needed when a single key field isn't unique.

Key fields must be a unique combination

```
list route {
  key "ip prefix";
  …

/route{16.40.0.0 16}/next-hop
 = 220.40.0.1
```

Key order significant

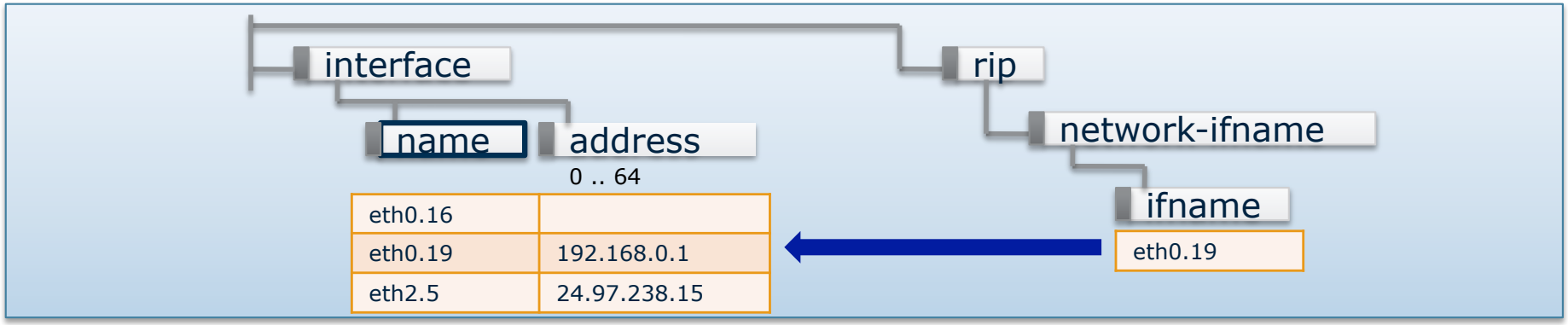
Leafref

To make an element reference one of the rows in a list, set the element type to leafref

For lists with multiple keys, the #leafrefs must match #keys in list

- A valid leafref can never be null/empty
  - But the parent leaf can be optional
- A valid leafref can never point to a row that has been deleted or renamed
- System checks validity of leafrefs automatically

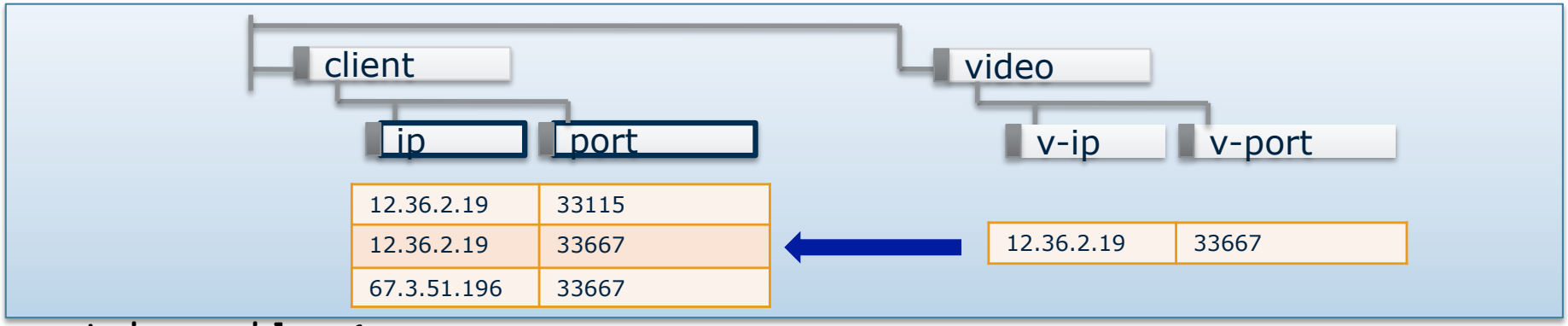
Leafref



Here, the RIP routing subsystem has a list of leafrefs pointing out existing interfaces

```
container rip {
  list network-ifname {
    key ifname;

    leaf ifname {
      type leafref {
        path "/interface/name";
      }
    }
```

Multiple Key Leafref




```
container video {
  leaf v-ip {
    type leafref {
      path "/client/ip";
    }
  }
  leaf v-port {
    type leafref {
      path "/client[ip=current()/../v-ip]/port";
    }
  }
}
```

Deref() XPATH Operator

```
container video {
  leaf v-ip {
    type leafref {
      path "/client/ip";
    }
  }
  leaf v-port {
    type leafref {
      path "/client
[ip=current()/../v-ip]/port";
    }
  }
  leaf v-stream {
    type leafref {
      path "/client
[ip=current()/../v-ip]
[port=current()/../v-port]
/stream";
    }
  }
}
```

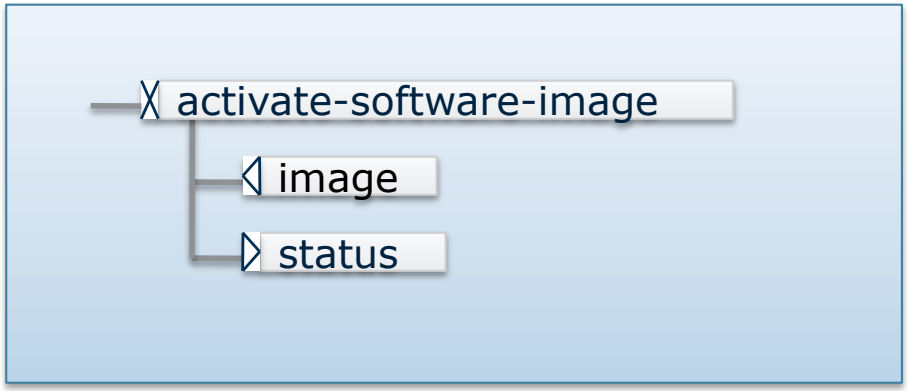
```
container video-deref {
  leaf v-ip {
    type leafref {
      path "/client/ip";
    }
  }
  leaf v-port {
    type leafref {
      path "deref(../v-ip)
/../port”;
    }
  }
  leaf v-stream {
    type leafref {
      path "deref(../v-port)
/../stream";


    }
  }
}
```

YANG RPCs & Notifications

RPC Statement

Administrative actions with input and output parameters

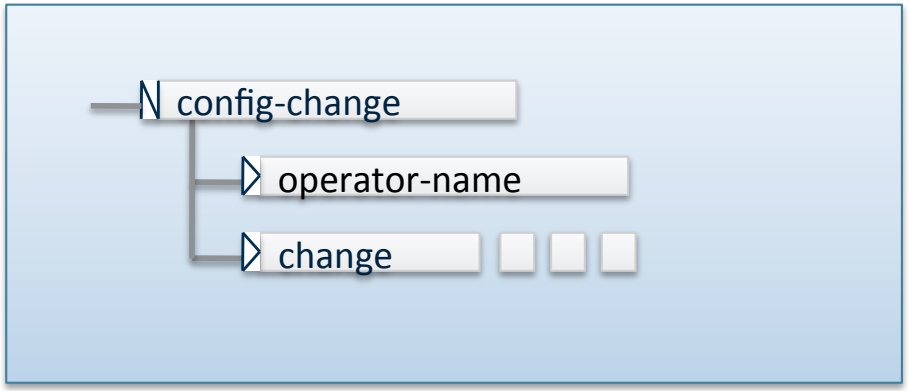
```
rpc activate-software-image {
  input {
    leaf image {
      type binary;
    }
  }
  output {
    leaf status {
      type string;
    }
  }
}
```



Notification Statement

Notification with output parameters

```
notification config-change {
  description
    "The configuration changed";
  leaf operator-name {
    type string;
  }
  leaf-list change {
    type instance-identifier;
  }
}
```



Instance-identifier values

```
<change>/ex:system/ex:services/ex:ssh/ex:port</change>
<change>/ex:system/ex:user[ex:name='fred']/ex:type</change>
<change>/ex:system/ex:server[ex:ip='192.0.2.1'][ex:port='80']</change>
```

Advanced YANG Statements

Must Statement

Restricts valid values by Xpath 1.0 expression

```
container timeout {
  leaf access-timeout {
    description "Maximum time without server response";
    units seconds;
    mandatory true;
    type uint32;
  }
  leaf retry-timer {
    description "Period to retry operation";
    units seconds;
    type uint32;
    must "current() < ../access-timeout" {
      error-app-tag retry-timer-invalid;
      error-message "The retry timer must be "
                  + "less than the access timeout";
    }
  }
}
```

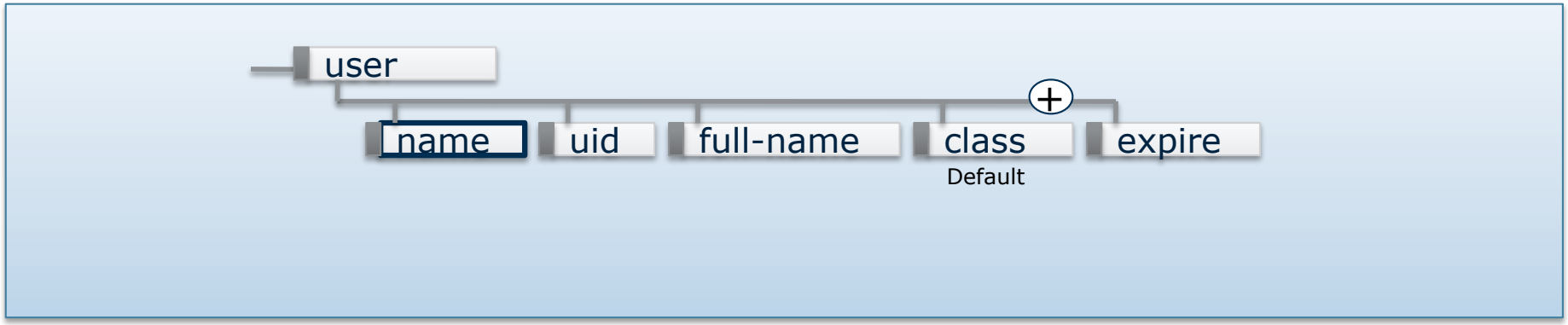
Must Statement

```
leaf interface-group-name {
  type string {
    length "1..31";
    pattern "[a-zA-Z][a-zA-Z0-9_-]*";
  }

  must "not(/sys:sys/interface[name = current()])" {

    error-message "Must be different from all interface names";
    tailf:dependency "/sys:sys/interface/name";
  }
}
```

Must Statement

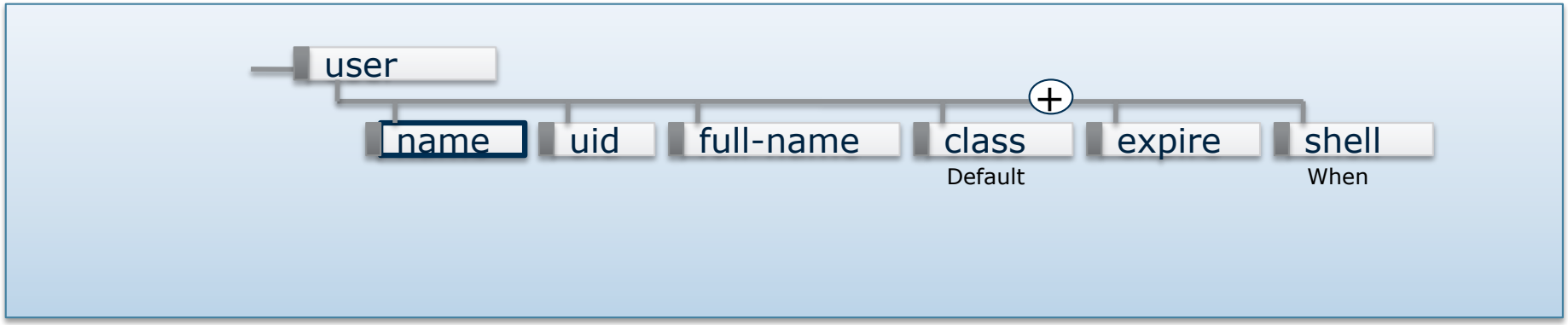
```
leaf max-weight {
  type uint32 {
    range "0..1000";
  }
  default 100;

  must "sum(/sys:sys/interface[enabled = 'true']/weight)
        < current()" {

    error-message "The total weight exceeds the configured
                   max weight";
  }
}
```

Augment Statement




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

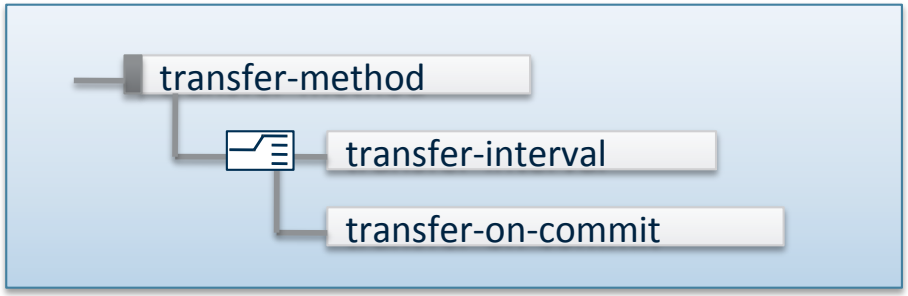
When Statement




```
augment /sys:system/sys:user {
  when "sys:class = 'wheel'";
  leaf shell {
    type string;
  }
}
```

Choice Statement

Choice allows one of several alternatives




```
choice transfer-method {
  leaf transfer-interval {
    description "Frequency at which file transfer happens";
    type uint16 {
      range "15 .. 2880";
    }
    units minutes;
  }
  leaf transfer-on-commit {
    description "Transfer after each commit";
    type empty;
  }
}
```

Choice Statement

Each alternative may consist of multiple definitions

- Either as a named or anonymous group

```
choice counters {
  case four-counters {
    leaf threshold {…}
    leaf ignore-count {…}
    leaf ignore-time {…}
    leaf reset-time {…}
  }
  container warning-only {
    …
  }
  default four-counters;
}
```

Identity Statement

Identities for modeling families of related enumeration constants

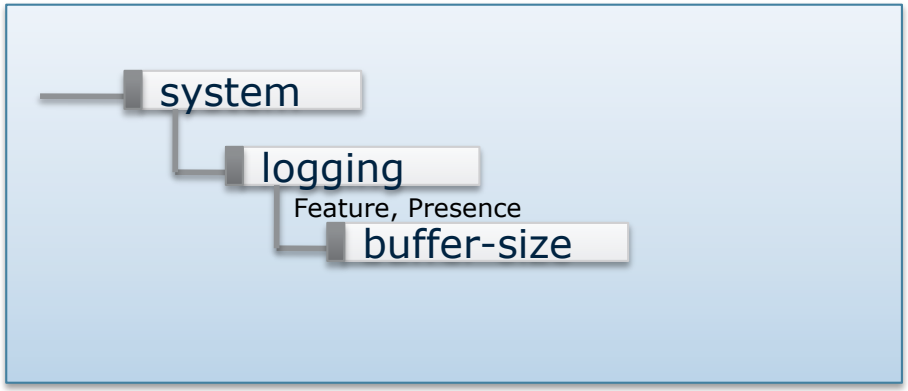
```
module phys-if {
…
  identity ethernet {
    description
      "Ethernet family of PHY
       interfaces";
  }
  identity eth-1G {
    base ethernet;
    description "1 GigEth";
  }
  identity eth-10G {
    base ethernet;
    description "10 GigEth";
  }
```

```
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";
    }
  }
```

Feature Statement

Mark data as conditional

```
feature has-local-disk {
  description
    "System has a local file
     system that can be used
     for storing log files";
}

container system {
  container logging {
    if-feature has-local-disk;
    presence "Logging enabled";
    leaf buffer-size {
      type filesize;
    }
  }
}
```



The features supported by a system are meant to stay relatively constant. Adding a feature is comparable to a hardware upgrade.

Deviations

Systems should conform to standard YAMs

- Still, if a device can't, that can be properly declared

```
deviation /base:system/base:user/base:type {
  deviate add {
    default "admin"; // new users are 'admin' by default
  }
}
deviation /base:system/base:name-server {
  deviate replace {
    max-elements 3;
  }
}
```

YANG Modeling Strategy

Constraints

- On-board XPATH parser not required
  - XPATH is more formal than English, that's all
- Debate*: "Configuration validity must not depend on state"*
  - Are operators right?
  - What are the implications?

- `min-elements`
- `max-elements`
- `range`
- `key/unique`
- `leafref`
- `must`
- `when`

Structure

- Model to mimic an existing interface?
- Model for which interfaces?
  - Yang for NETCONF only?
  - All interfaces uses the Yang model?
  - Multiple models with transformations?
  - Multiple models that allow writing?
- What goes in each namespace?
  - One, a few or many namespaces?
- No configuration inside operational data
  - SNMP transient configuration

So what is configuration?

- What is in the factory default?
- Any probing at system boot?
- Do you want pre-provisioning?
- Do you want to support rollbacks?
- Inserting a board, is that an act of configuration?
- Do you need spontaneous reconfiguration?
  - Where does the configuration data originate from?

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

Versioning Yang Modules (cont'd)

You may:

- Add or update a *status* statement: *current* ➔ *deprecated* ➔ *obsolete*
- Expand allowed *range*, *length*, *pattern*
- Add a *default* (but not change it!)
- Add *units*, add or update a *reference*

You may:

- Change *mandatory* to false (or remove it)
- Remove or relax *min-* & *max-elements*
- Add or clarify a *description* without changing the semantics
- Change state data to optional configuration
- Remove an if-feature

IETF Modeling Recommendations

- Describe
  - Module, security considerations, references, definitions, …
- Top level
  - Only one data definition (typ. container) on top level
  - Top level data definitions optional
- Enumerations that grow
  - Use identityref instead of enumeration

IETF Modeling Recommendations

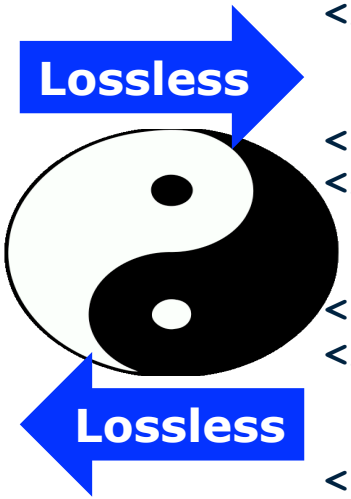
- Specific types
  - Use ranges, length restrictions, patterns
  - Do not use signed types unless negative values needed
- Standard types
  - When possible, use standard types, e.g. 'yang-types' defined in RFC 6021
  - Make your own reusable 'type modules'

Yin, Yang and Yang Extensions

Yin – the XML representation of Yang

YANG 日

```
container dhcp {
  leaf defaultLeaseTime {
    type xs:duration;
    default PT600S;
  }
  leaf maxLeaseTime {
    type xs:duration;
    default PT7200S;
  }
  leaf logFacility {
    type loglevel;
    default local7;
  }
```



YIN 月

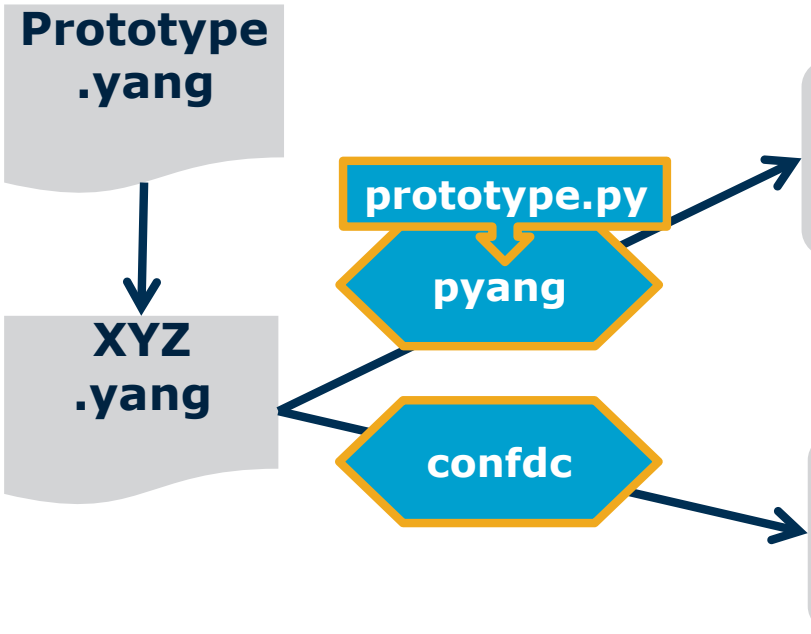
```
<container name="dhcp">
  <leaf name="defaultLeaseTime">
    <type name="xs:duration"/>
    <default value="PT600S"/>
  </leaf>
  <leaf name="maxLeaseTime">
    <type name="xs:duration"/>
    <default value="PT7200S”/>
  </leaf>
  <leaf name="logFacility">
    <type name="loglevel"/>
    <default value="local7"/>
  </leaf>
```

YANG Extensions - Prototype

```
module prototype {
  namespace "http://tail-f.com…
  prefix prototype;

  extension create {
    argument keylist;
  }
  extension value {
    argument value;
  }
  extension keylist-value {
    argument keylist-value;
  }
}
```

```
module xyz {
  list y {
    config false;
    tailf:cdb-oper;
    prototype:create "b=1,c=0";
    prototype:create "b=6,c=0";
    prototype:create "b=6,c=1";
    key "b c";
    leaf b { type int32; }
    leaf c { type int32; }
    leaf d {
      type string;
      prototype:keylist-value
        "b=1,c=0,cool";
      prototype:keylist-value
        "b=6,c=1,green";
    }
```

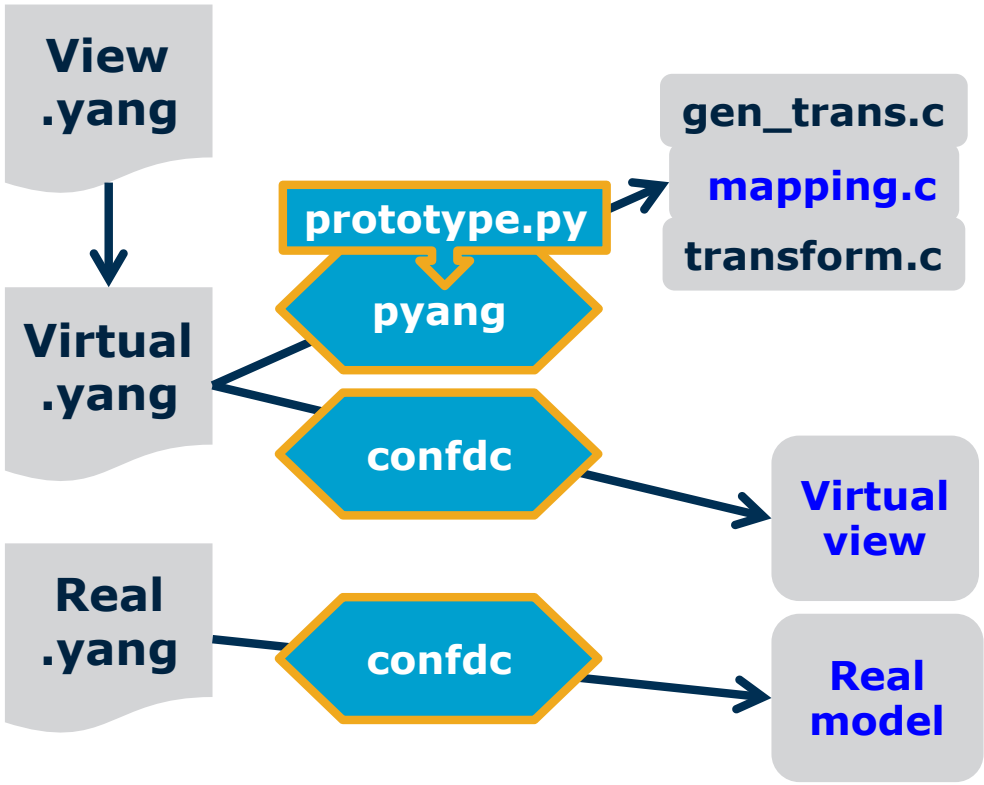
Yang Extensions - Prototype




```
c "/xyz:x/xyz:y{1 0}"
c "/xyz:x/xyz:y{6 0}"
c "/xyz:x/xyz:y{6 1}"
s "/xyz:x/xyz:y{1 0}/xyz:d" "cool"
s "/xyz:x/xyz:y{6 1}/xyz:d" "green"
```

```
ubuntu# show x
B  C  D
-------------
1  0  cool
6  0  -
6  1  green
```

Yang Extensions – View Transform Plugin




```
list interface-problem {
  view:pathmap "/iface";
  view:keymap
    "ifname2ifidx_keymap";
  view:filter
    "interface_problem_filter";
  key name;
  leaf name {
    type interface-type;
  }
}
```

Example:
Router Device Model (part of)

Example: Quagga Types

```
submodule quagga_common {
    belongs-to quagga {
        prefix quagga;
    }
…
    include quagga_top;
    typedef PrefixLengthIPv4 {
        type uint8 { range 0 .. 32; }
    }
…
    grouping inetNetworkAddressIPv4 {
        leaf address {
            type inet:ipv4-address;
            mandatory true;
        }
        leaf prefix-length {
            type PrefixLengthIPv4;
            mandatory true;
        }
    }
    typedef MacAddressType {
        type string {
            length 17;
            pattern "([0-9a-fA-F][0-9a-fA-F]:){5}
                     ([0-9a-fA-F][0-9a-fA-F])";
        }
    }
```

```
typedef ACLLowRangeType {
    type uint32 { range 1 .. 99; }
}
typedef ACLHighRangeType {
    type uint32 { range 1300 .. 1999; }
}
typedef ACLExtType {
    type union {
        type ACLExtLowRangeType;
        type ACLExtHighRangeType;
    }
}
typedef InfiniteType {
    type enumeration {
        enum infinite;
    }
}
typedef DateTimeInfiniteType {
    type union {
        type yang:date-and-time;
        type InfiniteType;
    }
}
```

Example: Quagga Interface Configuration

```
submodule zebra_confd_if {
    belongs-to quagga {
        prefix quagga;
    }
    include confd_top;
    include confd_common;
    typedef BandwidthRangeType {
        type uint64 { range "1 .. 10000000000"; }
    }
    augment "/system/interface" {
        leaf bandwidth {
            type BandwidthRangeType;
        }
        leaf link-detect {
            type boolean;
            default false;
        }
        leaf multicast {
            type MulticastType;
            default if-default;
        }
        leaf up {
            type boolean;
            default true;
        }
    }
}
```

```
submodule zebra_confd_if_ip {
    belongs-to quagga {
        prefix quagga;
    }
    import ietf-inet-types {
        prefix inet;
    }
    include confd_top;
    include confd_common;
    augment "/system/interface/ip" {
        list address {
            key "address prefix-length";
            leaf address {
                type inet:ipv4-address;
            }
            leaf prefix-length {
                type PrefixLengthIPv4;
            }
            leaf label {
                type string;
            }
        }
    }
}
```

Example: Quagga Status & Notification

```
submodule zebra_confd_stats_memory {
    belongs-to quagga {
        prefix quagga;
    }
    import tailf-common {
        prefix tailf;
    }
    include confd_top;
    include confd_common;
    augment "/system/stats/memory" {
        container zebra {
            config false;
            tailf:callpoint "zebraGetMemory";
            uses memoryStatusType;
        }
    }
}
```

```
notification saExpired {
    leaf tunnelName {
        type leafref {
            path "/system/vpn/ipsec/tunnel/name";
        }
        mandatory true;
    }
    leaf spi {
        type int64;
        mandatory true;
    }
}
```

Example:
Q-in-Q Service Model

Example: Q-in-Q Service Model (complete)

```
module qinq-service {
  namespace "http://com/example/qinq-service";
  prefix qinq;

  import tailf-common { prefix tailf; }
  import tailf-ncs    { prefix ncs;   }

  list qinq {
    key name;
    leaf name { type string; }

    leaf s-vlan {
      mandatory true;
      type uint32 { range 1..4094; }
    }

    must "not (/qt:qinq[name != current()/name]
                      [s-vlan = current()/s-vlan])" {
      error-message "The s-vlan must be unique within
                     the network"
    }

    list edge-switch {
      key switch;
      leaf switch {
        type leafref {path /ncs:devices/ncs:device/ncs:name;}
      }
      list edge-interface {
        key interface;
        leaf interface { type string; }
        leaf-list c-vlan {
          type uint32 { range 1..4094; }
        }
      }
      leaf-list trunk-interface {
        min-elements 1;
        type string;
      }
    }
    list core-switch {
      key switch;
      leaf switch {
        type leafref {path /ncs:devices/ncs:device/ncs:name;}
      }
      leaf-list trunk-interface {
        min-elements 1;
        type string;
      }
    }
  }
}
```

tail-f

Demos

tail-f

Current IETF Status

NETCONF RFC Overview

- RFC 3535 Informational: Background
- RFC 6244 NETCONF+Yang Architectural Overview
- RFC 6241 Base NETCONF Protocol
- RFC 6242, 4743-4744, 5539 Transport Mappings
- RFC 5277 Notifications
- RFC 5717 Partial Locking
- RFC 6243 With defaults
- RFC 6470 Base Notifications
- RFC 6536 NETCONF Access Control Model

ⓘ https://datatracker.ietf.org/wg/netconf/charter/

ⓘ www.rfc-editor.org/rfc/rfcXXXX.txt

YANG RFC Overview

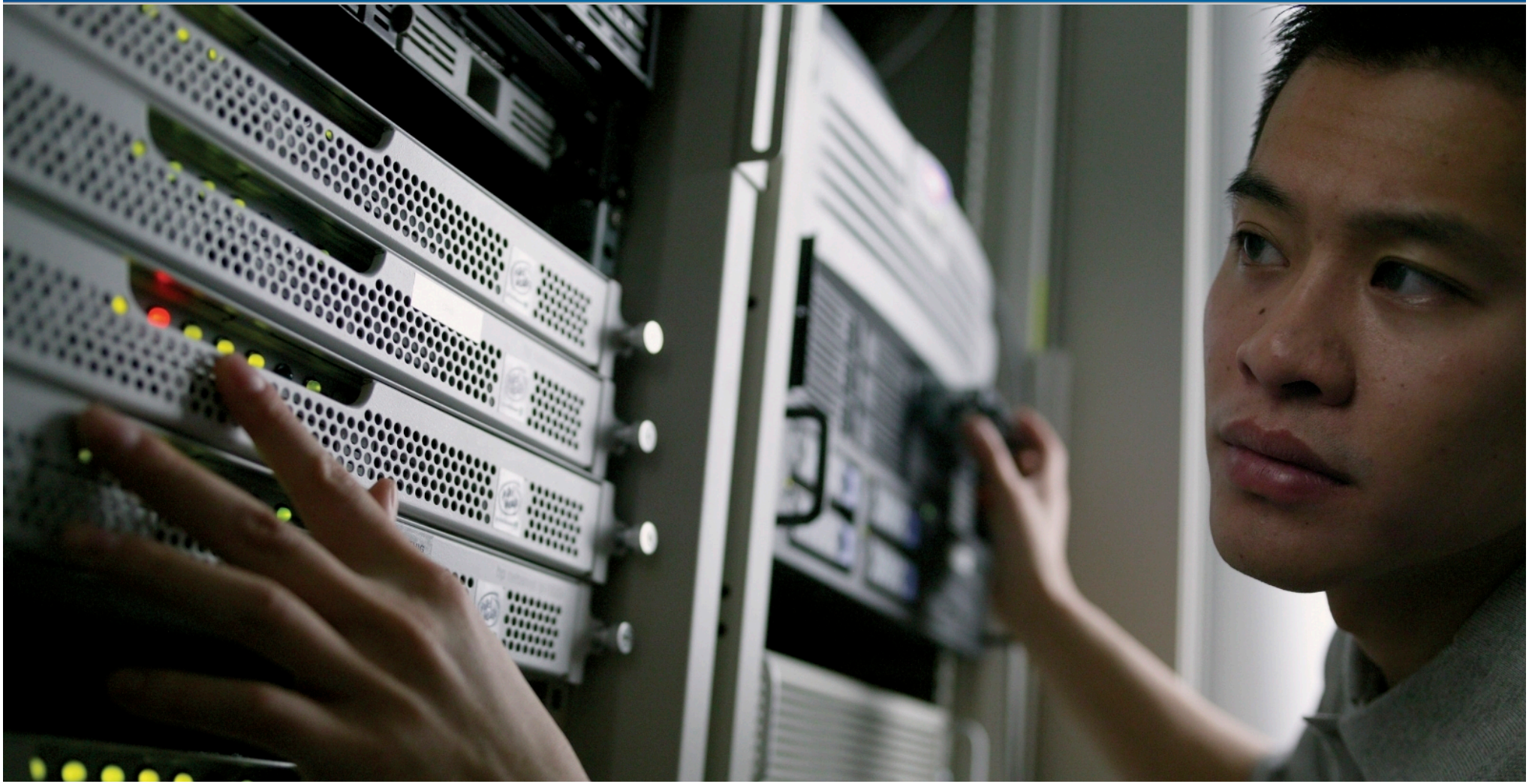
- RFC 6020 YANG Base Specification
- RFC 6021 YANG Types
- **RFC 6087 Guidelines for YANG Authors and Reviewers**
- RFC 6110 Mapping and Validating YANG
- **RFC 6244 NETCONF+Yang Architectural Overview**
- RFC 6643 Translation of SMIv2 MIBs to YANG

Start Here!

- https://datatracker.ietf.org/wg/netmod/charter/
- https://www.ietf.org/iesg/directorate/yang-doctors.html
- http://www.yang-central.org/

Rather than risking getting things wrong, ask the YANG Doctors

tail-f




tail-f

SDN and NETCONF/YANG

- a happy marriage

http://www.slideshare.net/martin_casado/sdn-abstractions

The Future of Networking,

and the Past of Protocols

Scott Shenker

with Martin Casado, Teemu Koponen, Nick McKeown

(and many others….)

But Talk Is Not Primarily About SDN

- Main focus is on:

The Role of Abstractions in Networking

Weak Evolutionary Foundations

- Ongoing innovation in systems software
  - New languages, operating systems, etc.
- Networks are stuck in the past
  - Routing algorithms change very slowly
  - Network management extremely primitive

NETCONF/YANG and SDN?

- SDN is more than OpenFlow
  - But:
  - OF-CONFIG is YANG and NETCONF
- SDN the big picture
  - Schenker et al: abstractions for network management
  - Transactions and data-models
- Dynamic programmatic network configuration