tail-f

Tutorial: NETCONF and YANG

Presented by Stefan Wallin, Tail-f

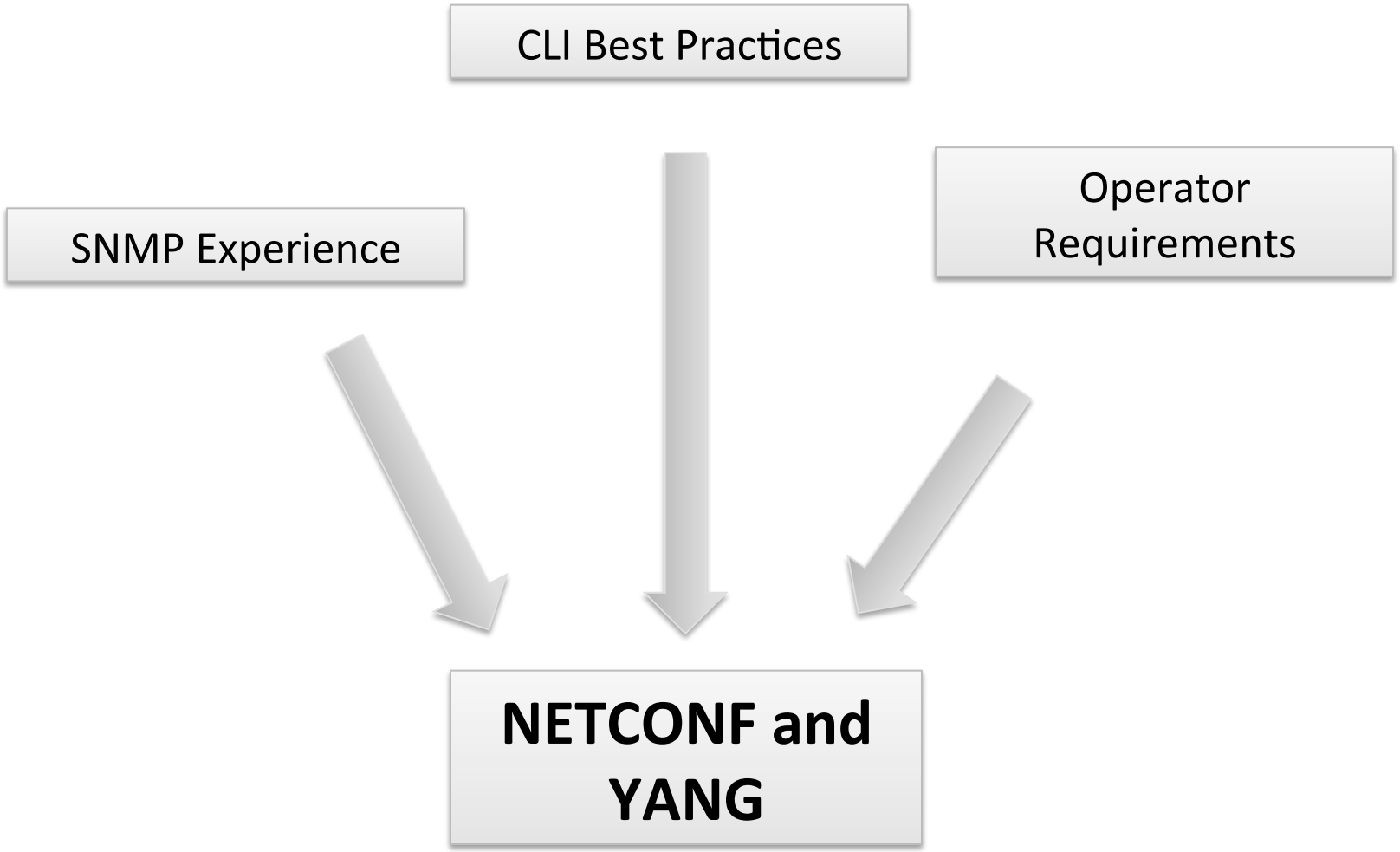
stefan@tail-f.com

Today's Topic: #1 Market Leader in Configuration Management

Origins of NETCONF and YANG (the Beginning)

- Several meetings at events in 2001 (NANOG-22, RIPE-40, LISA-XV, IETF 52)
  - Operators expressing opinion that the developments in IETF do not really address requirements configuration management.
- June of 2002, the Internet Architecture Board (IAB) held invitational workshop on Network Management [RFC3535] to
  - Identify a list of technologies relevant for network management with their strengths and weaknesses
  - Identify the most important operator needs.

Best Practices Coming Together



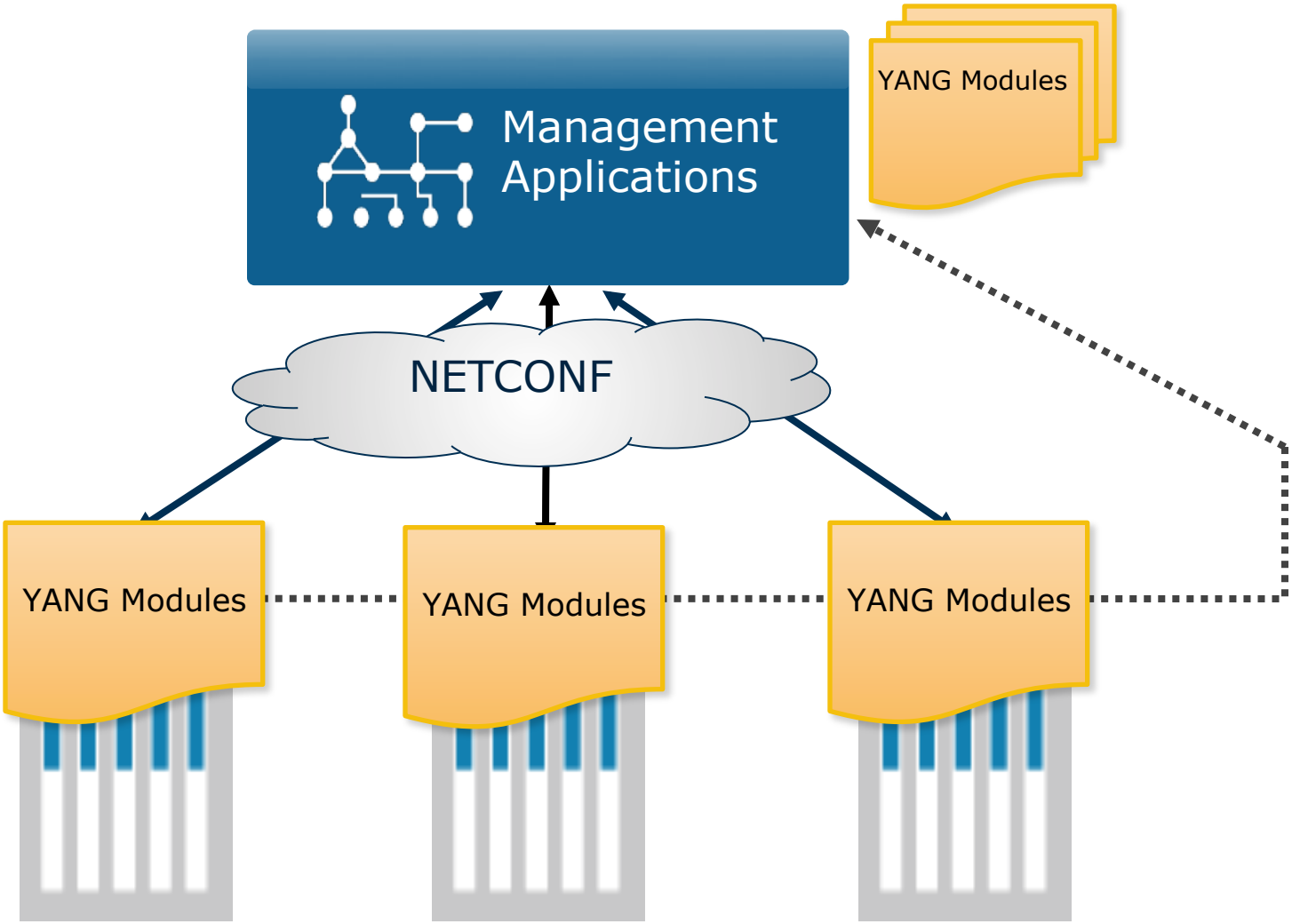
Agenda

- Quick Overview (RFC 6244, 2011)
- Background and Motivation (RFC 3535, 2003)
- NETCONF (RFC 6241, 2006, 2011)
- YANG (RFC 6020, 2010)
- Complete Example
- Demo

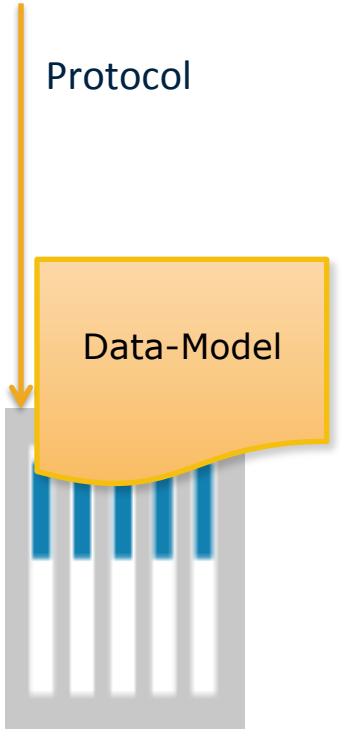
NETCONF and YANG

What is it?

NETCONF and YANG in Context

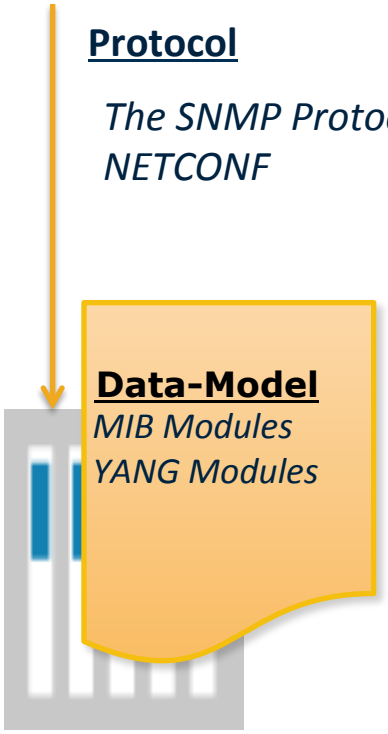


What is a Data-Model? What is a Network Management Protocol?



- 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. SOAP, REST, …

NETCONF – A Protocol to Manipulate Configuration

- IETF network management protocol
- Distinction between configuration and state data
- Multiple configuration data stores (candidate, running, startup)
- Configuration change validations
- Configuration change transactions
- Selective data retrieval with filtering
- Streaming and playback of event notifications
- Extensible remote procedure call mechanism

Why you should care:

NETCONF provides the fundamental programming features for comfortable and robust automation of network services

YANG – A Data Modeling Language for Networking

- Human readable, and easy to learn representation
- Hierarchical configuration data models
- Reusable types and groupings (structured types)
- Extensibility through augmentation mechanisms
- Supports definition of operations (RPCs)
- Formal constraints for configuration validation
- Data modularity through modules and sub-modules
- Well defined versioning rules

```
list interface {
        key "name";
        unique "type location";

        leaf name {
          type string;
          reference
            "RFC 2863: The Interfaces Group MIB - ifName";
        }

        leaf description {
          type string;

...

  container statistics {
          config false;
          leaf discontinuity-time {
            type yang:date-and-time;
          }

          leaf in-octets {
            type yang:counter64;
            reference
              "RFC 2863: The Interfaces Group MIB - ifHCInOctets";
          }
```

Why you should care:

YANG is a full, formal contract language with rich syntax and semantics to build applications on

Who ?

NETCONF

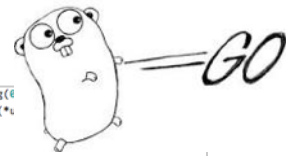
- Phil Shafer, Rob Enns
  - Juniper
- Jürgen Schönwälder
  - Jacobs University
- Martin Björklund
  - Tail-f
- Andy Bierman
  - Yumaworks
- Ken Crozier Eliot Lear
  - Cisco Systems
- Ted Goddard
  - IceSoft
- Steve Waldbusser
- Margaret Wasserman
  - Painless Security, LLC

YANG

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

Language Bindings


```ruby
require 'net/netconf'

puts "NETCONF v#{Netconf::VERSION}"

login = { :target => 'jeap', :port => 2022,
  :username => "admin", :password => "admin" }

Netconf::SSH.new( login ){ |dev|

  config = dev.rpc.get_config
```




```go
        s, err := netconf.DialSSH(flag.Arg(0
                netconf.SSHConfigPassword(*u
        if err != nil {
                panic(err)
        }

        defer s.Close()

        fmt.Printf("Server Capabilities: '%+v'\n", s.ServerCapabilities)
        fmt.Printf("Session Id: %d\n\n", s.SessionID)
```


```python
from ncclient import manager

# use unencrypted keys from ssh-agent or ~/.ssh keys, and rely on known_hosts
with manager.connect_ssh("host", username="user") as m:
    assert(":url" in m.server_capabilities)
    with m.locked("running"):
        m.copy_config(source="running", target="file:///new_checkpoint.conf")
        m.copy_config(source="file:///old_checkpoint.conf", target="running")
```

```perl
my $jnx = new Net::Netconf::Manager(%deviceinfo);
unless (ref $jnx) {
    croak "ERROR: $deviceinfo{hostname}: failed to connect.\n";
}
```


```java
    public static Configuration getJunosConfiguration(NodeSet
        Element config = null;
        for (Element elem : configs) {
            if (elem.name.equals("configuration")) {
                config = elem;
                break;
            }
        }
        return (Configuration)config;
    }
```

Momentum

A common, standard configuration management p
data modeling language is needed in order t
automation required by TeraStream. NETCONF and YANG are suitable because they are self-contained, in the sense that no additional offline knowledge about the syntax and transport of configuration exchange is needed. A CLI falls short since there is no formal data model and no formal vendor-neutral syntax. SNMP (Simple Network Management Protocol) falls short because it requires the application to have knowledge about in which order things can be created or modified.

DTAG TeraStream Device Management Interface Requirements

Deutsche Telekom

CISCO

ERICSSON

JUNIPER NETWORKS

BROCADE

IETF

CableLabs

ONF OPEN NETWORKING FOUNDATION

MEF METRO ETHERNET FORUM

tail-f

OPEN DAYLIGHT

MG SOFT

YumaWorks Automating the Internet

Standards background, motivation and history

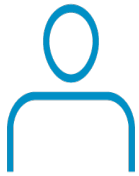
RFC 3535: Operators' problems and requirements on network management

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%+

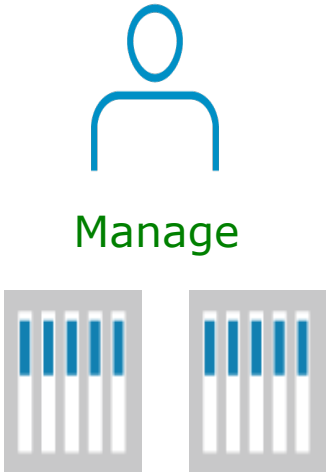


configuration

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

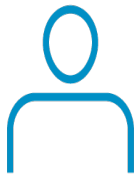


Operator Requirement #2-3/14

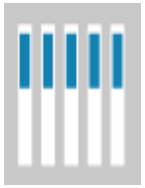
2. It is necessary to make a **clear distinction** between **configuration data**, data that describes **operational state and statistics**.

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
```

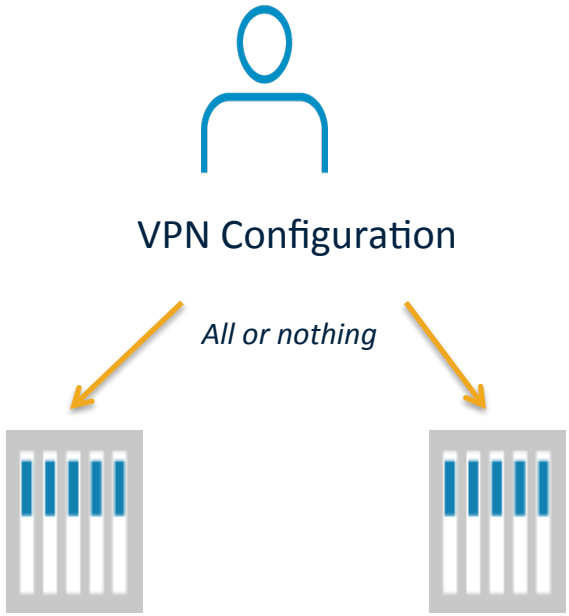


Operator Requirement #4-5/14

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



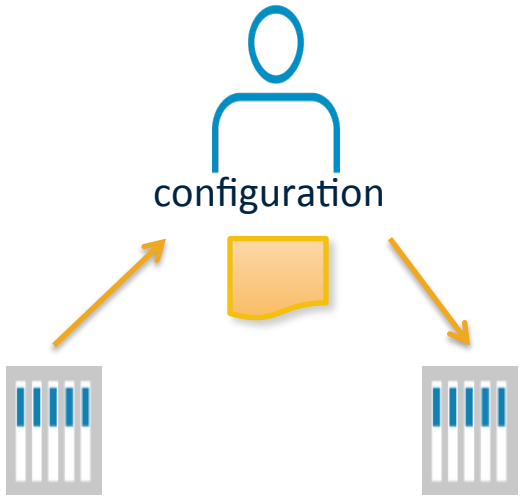
Operator Requirement #6-7/14

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

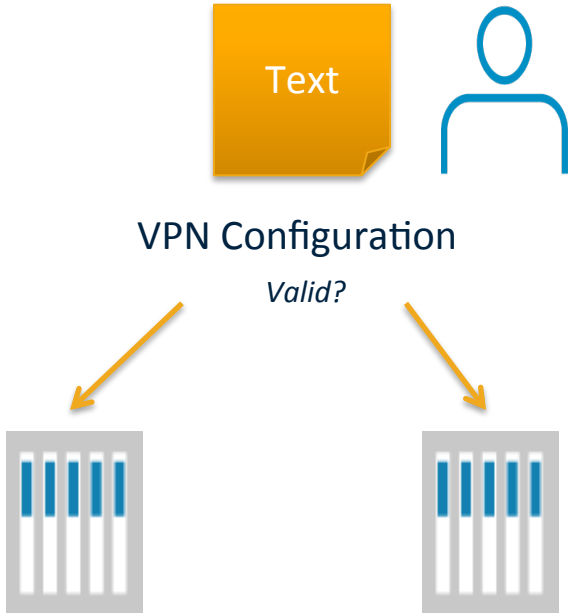


Operator Requirement #8, 10/14

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

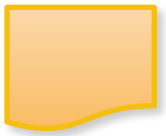


Operator Requirement #9/14

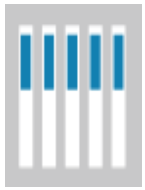
9. Network wide configurations are typically stored in central master databases and transformed into formats that can be pushed to devices, either by generating sequences of CLI commands or complete configuration files that are pushed to devices. There is no **common database schema** …, although the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

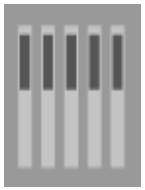
It is desirable to extract, document, and standardize the common parts of these network wide configuration database schemas.

- Standardized data models



Interfaces Data-Model



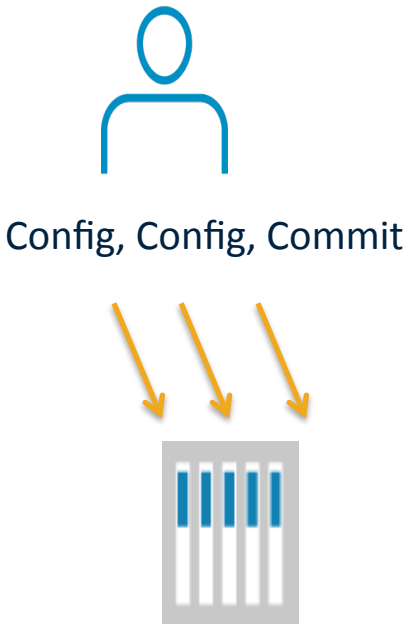


Operator Requirement #13/14

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

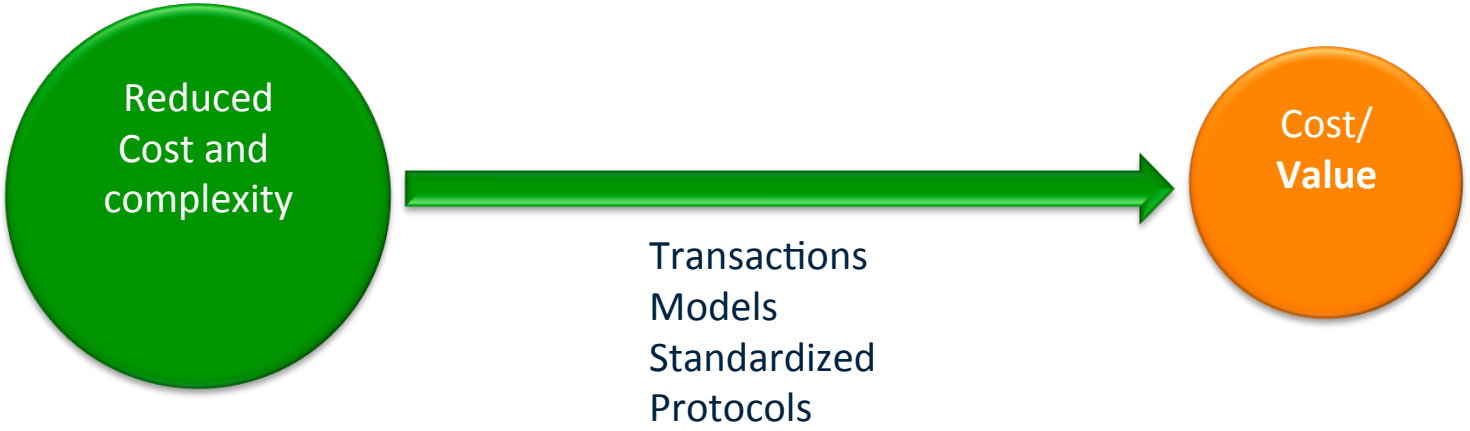
- Role-Based Access Control (RBAC)
  - Data oriented
  - Task oriented

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



Introduction to NETCONF

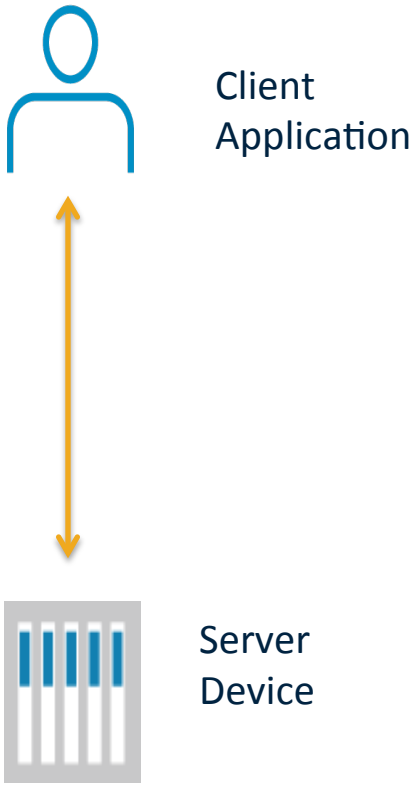
What makes NETCONF/YANG different?

	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

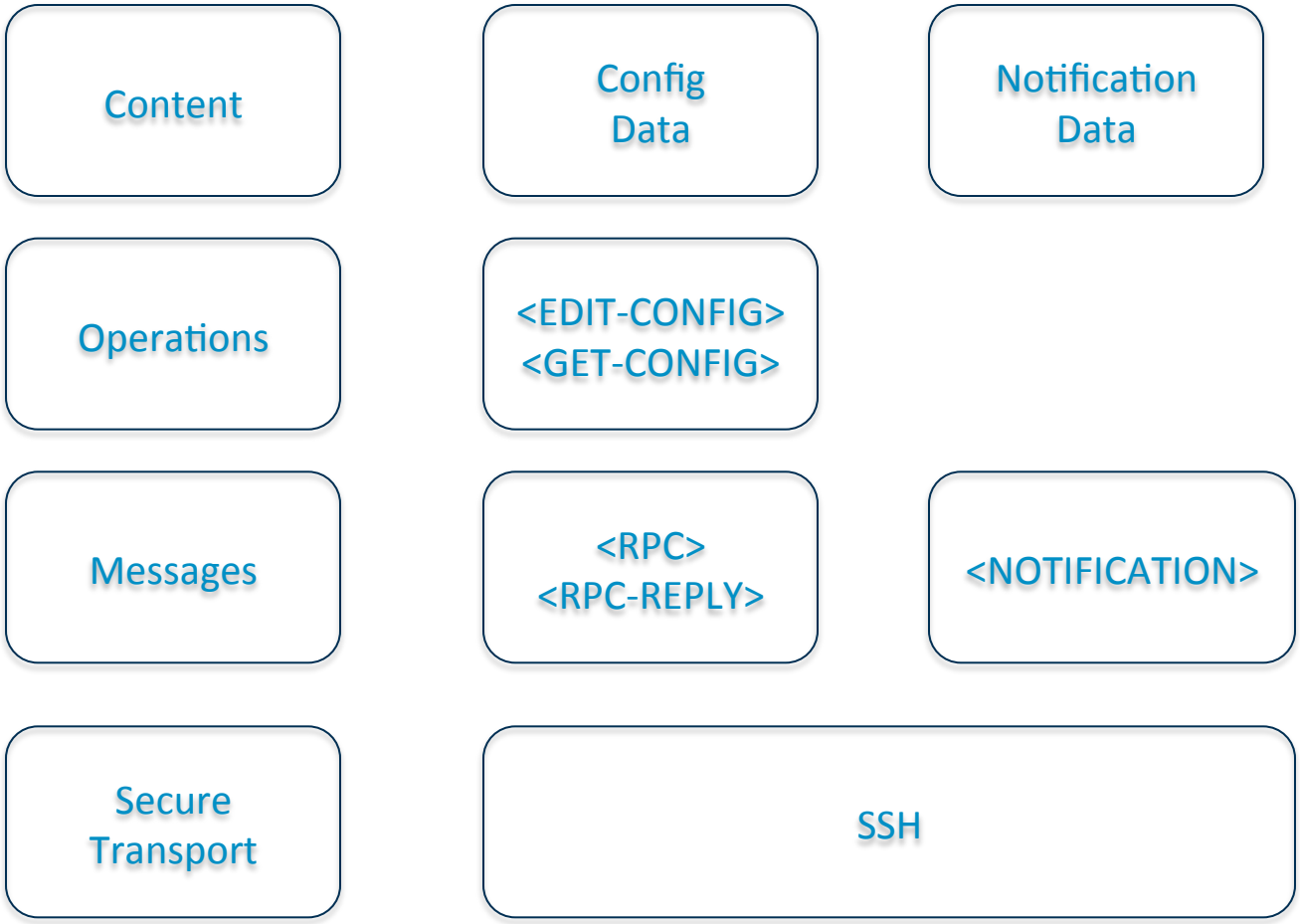
"RESTConf"

NETCONF Features

- API to configure network devices
- XML RPC mechanism
- Connection-oriented (SSH)
- Closely mirrors the device functionality
- Capability Discovery
- Separation of configuration and state data
- Sub-tree filtering



Layers



NETCONF Capabilities

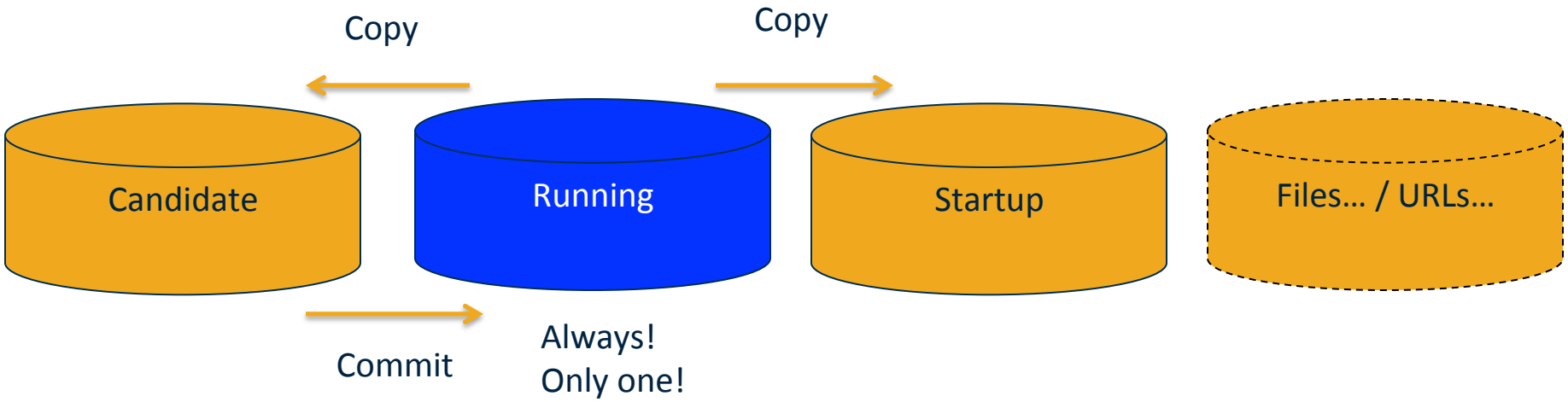
- A capability is a set of functionality that supplements base NETCONF spec
- Capabilities augment:
  - Additional operations
  - Content allowed inside these operations
- Capabilities advertised by server during session establishment

- Base NETCONF specification provides very restricted set of operations for lightweight server implementations

```
with manager.connect(host=host, port=22, username=user) as m:
    assert(":validate" in m.server_capabilities)
    m.edit_config(target='running', config=snippet,
                  test_option='test-then-set')
```

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:confirmed-commit:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
    <capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
    <capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
    <capability>urn:ietf:params:xml:ns:yang:ietf-inet-types?module=ietf-inet-types&amp;revision=2013-07-15</capability>
    <capability>urn:ietf:params:xml:ns:yang:ietf-netconf-acm?module=ietf-netconf-acm&amp;revision=2012-02-22</capability>
    <capability>urn:ietf:params:xml:ns:yang:ietf-yang-types?module=ietf-yang-types&amp;revision=2013-07-15</capability>
  </capabilities>
  <session-id>14</session-id>
</hello>
```

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

Common Operations

Data Manipulation

- `<get>`
- `<get-config>`
- `<edit-config>`
- `<copy-config>`
- `<delete-config>`
- `<discard-changes>` *(:candidate)*

Session Management

- `<close-session>`
- `<kill-session>`

Locking

- `<lock>`
- `<unlock>`

Transaction Management

- `<commit>` *(:candidate, :confirmed)*
- `<cancel-commit>` *(:confirmed)*

Schema Management

- `<get-schema>` *(:monitoring)*

RPC Extensions

- `<rpc>`

Anatomy of NETCONF Sessions

Ambitious version:

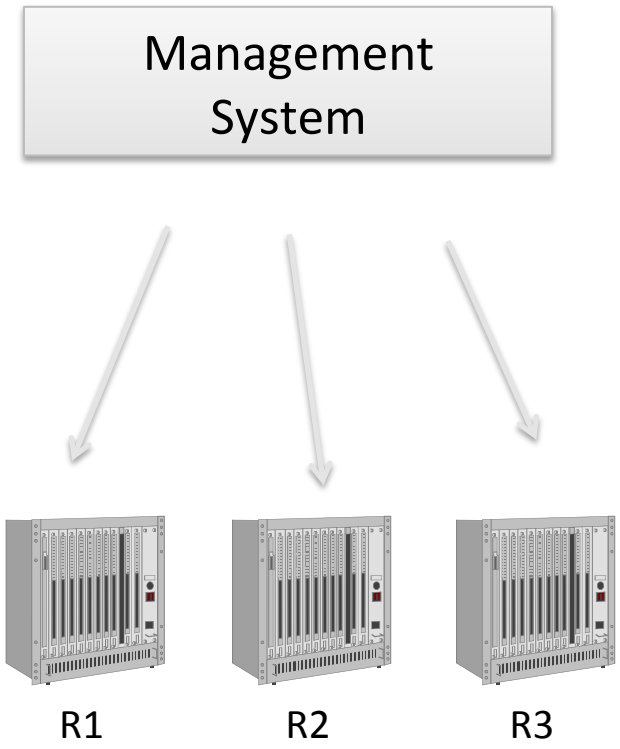
- **Hello** exchange including capabilities
- **Lock** running
- **Lock** candidate
- **Discard** changes on candidate
- **Edit** config on candidate
- **Commit** confirmed (with timeout)
- **Confirm** commit
- **Copy** running to startup
- **Unlock** candidate
- **Unlock** running

Short version:

- **Hello** exchange including capabilities
- **Edit** config on running database


```
with manager.connect(host=host, port=22, username=user) as m:
    if(":candidate" in m.server_capabilities):
        with m.locked(target='candidate'):
            m.discard_changes()
            ...
    else:
        m.edit_config(target='running', config=cfg)
```

Distributed Transactions (for Bonus Points)



1. Connect to and lock R1, R2, R3
2. Edit candidate databases and commit with timeout
3. (Optionally) do assurance checks during timeout
4. Confirm commit, copy to startup and release locks

Transaction context manager simply kills all sessions on communication failure, failed commits -> Rollback

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:1.1" message-id="7">
  <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 exists

Error if item to delete does not exist

Ok if item to delete does not exist

url can be used instead of inline config (:url capability)

NETCONF <copy-config> Operation

- Copy and replace 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>
```

NETCONF capabilities

- :writable-running
  - <edit-config> and <copy-config> can have candidate as a <target>
- :candidate
  - Share “scratch-pad”
  - <commit> to running
  - <discard-changes>
- :confirmed-commit
  - <confirmed> parameter to commit
  - <cancel-commit>
- :validate
  - <validate>
  - <test-option> parameter to the <edit-config>
- :startup
- :url
- :xpath

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>
```

NETCONF FILTERS

NETCONF Filters

- Namespace Selection
- Containment Nodes
- Selection Nodes
- Content Match Nodes
- Attribute Match Expression
- XPATH (capability)

NETCONF Filters: Namespace selection

```
<filter type="subtree">
  <top xmlns="http://example.com/schema/1.2/config"/>
</filter>
```

Selection Node

An empty leaf node within a filter

```
<top/>
<top></top>
```

NETCONF Filters: Containment

```
<filter type="subtree">
  <top xmlns="http://example.com/schema/1.2/config">
     <users/>
  </top>
</filter>
```

NETCONF Filters: Content Match

```
<filter type="subtree">
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>fred</name>
        </user>
      </users>
    </top>
  </filter>
```

NETCONF Filters: Attribute Match

NOTIFICATIONS

Event Notifications

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

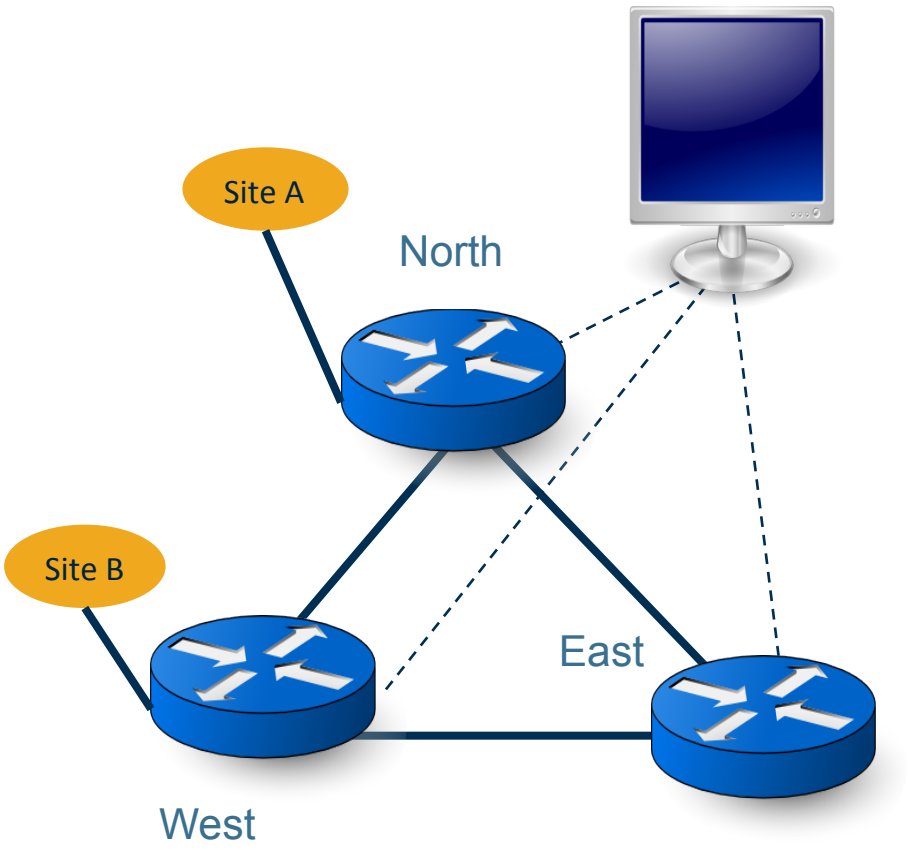
ⓘ www.rfc-editor.org/rfc/rfc5277.txt

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

Example:

VPN provisioning using NETCONF Network-wide Transactions

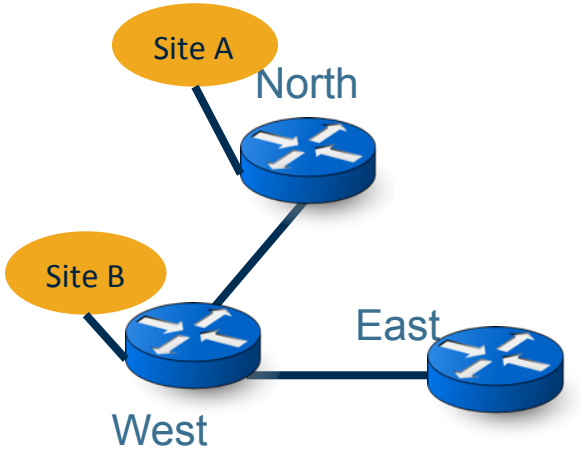
VPN Scenario



- 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>
```

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>
```

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>
```

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.
```

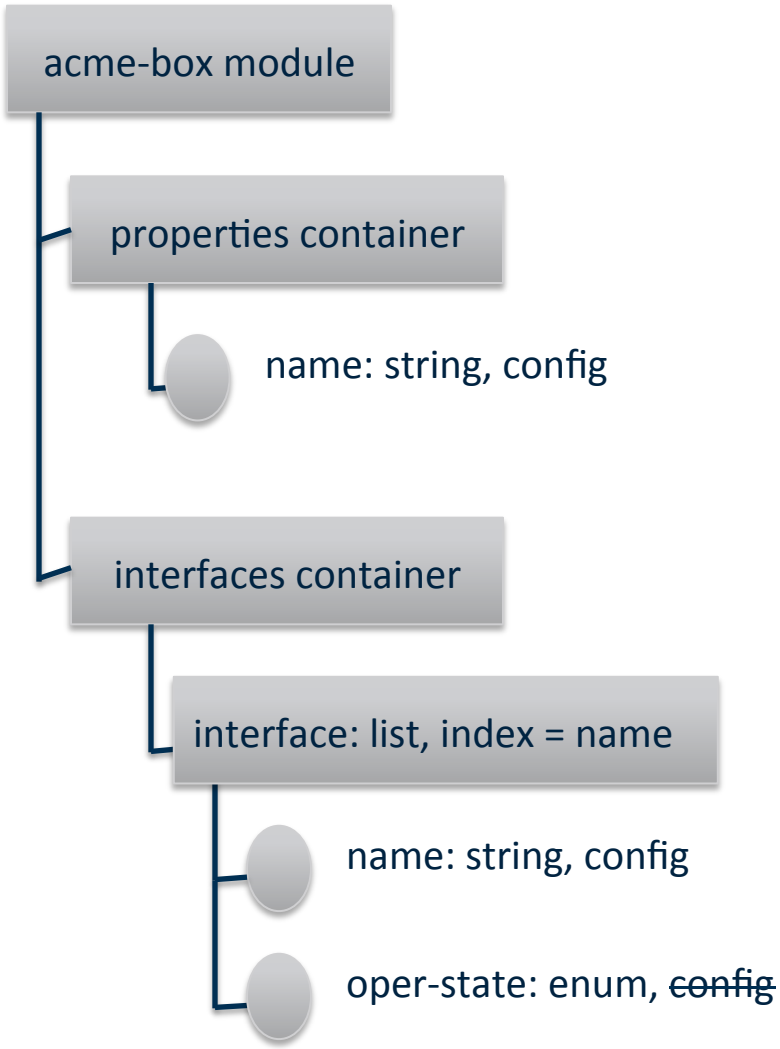
Common NETCONF Use Cases

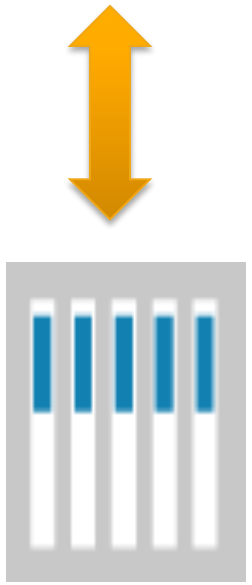
- Network-wide transactions
- Applying and testing a configuration
- Testing and rejecting a configuration
- Rollback when device goes down
- Transactions requiring all devices to be up
- Backlogging transactions
- Synchronizing

YANG Overview and Examples

YANG ?

- Data modeling language
  - Configuration data
  - State data
- Tree structure
- Close to device
- Managing device features
- Data and Types
- Constraints
- Augmentation
- Reusable structures
- Extensions
- SMI translation
- XML
  - NETCONF Transport Encoding
  - YANG – XML Model mapping



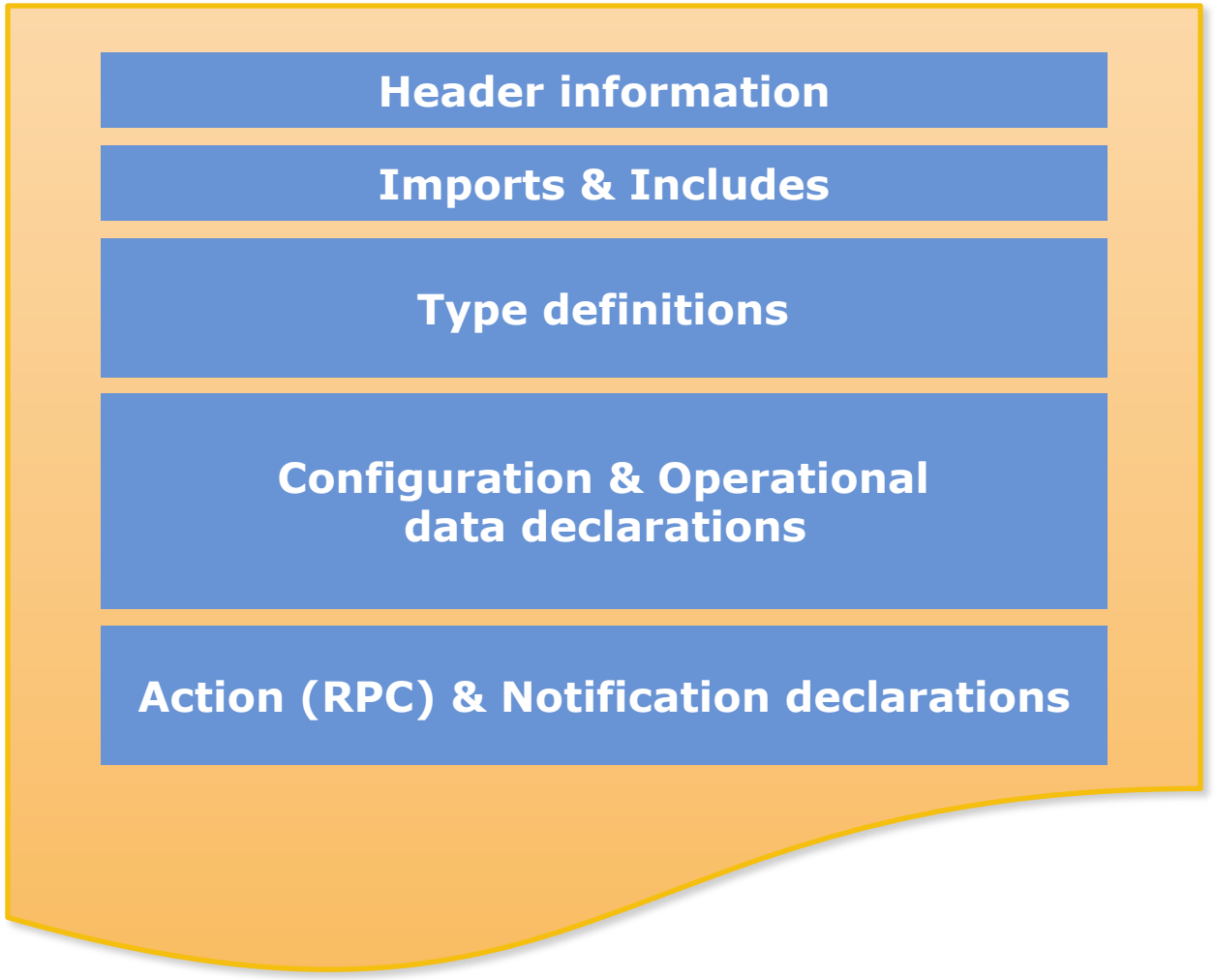


YANG ?

```
module acme-box {
  namespace "http://acme.net/yang/box";
  prefix "box";

  container properties {
    leaf name {
      type string;
    }
  }
```

```
  container interfaces {
    list interface {
      key name;
      leaf name {
        type string;
      }
      leaf oper-state {
        config false;
        type enumeration {
          enum enabled;
          enum disabled;
        }
      }
    }
  }
}
```

YANG Module Contents



YANG Header

```
module acme-box {
  namespace "http://acme.net/yang/box";
  prefix "box";

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

  organization "ACME Inc.";
  contact "joe@acme.net";
  description "Magic box";
  revision "2014-04-12" {
    description "For RIPE";
  }
```

URI

YANG Data Definitions

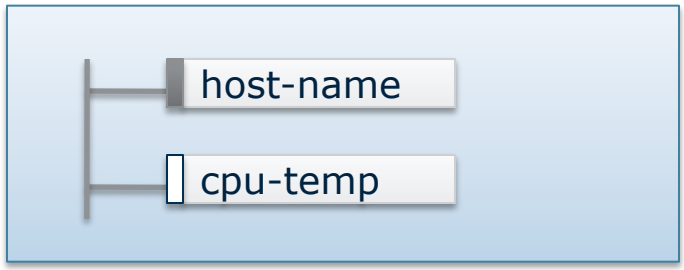
Data Modeling Nodes

- Leaf
- Leaf-List
- Container
- List

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";
}
```

NETCONF XML:

```
<host-name>my.example.com</host-name>
```

cpu-temp **not** returned in NETCONF get-config

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”

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";
}
```

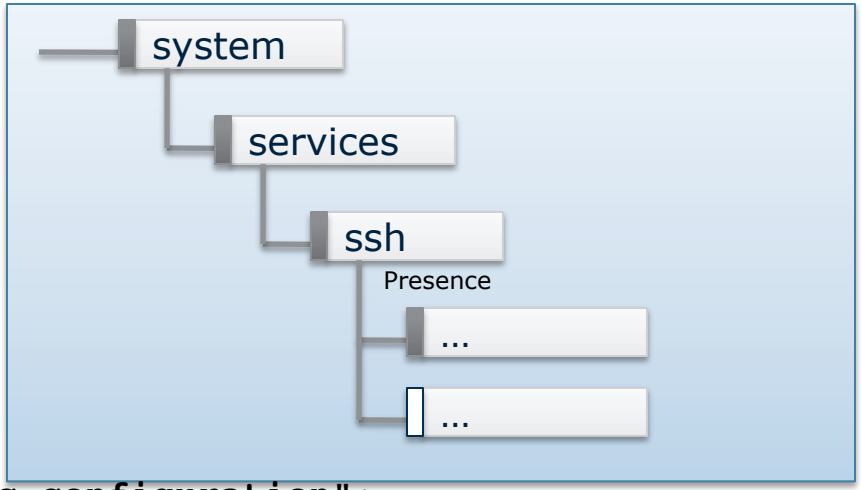
NETCONF operations to insert first, last, before, after

NETCONF XML:

```
<domain-search>high.example.com</domain-search>
<domain-search>low.example.com</domain-search>
```

Container Statement

Groups related leafs and containers



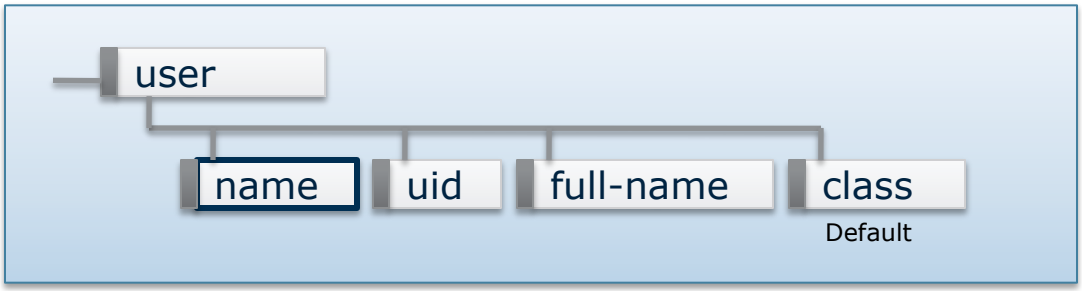

```
container system {
  container services {
    container ssh {
      presence "Enables SSH";
      description "SSH service specific configuration";
       // more leafs, containers and other things here...
    }
  }
}
```

NETCONF XML:

```
<system>
   <services>
      <ssh>
      </ssh>
   </services>
</system>
```

Presence containers explicitly created/deleted by NETCONF client. They also represent config “themselves”. “Normal” containers have no meaning, just organization of data.

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;
  }
}
```

NETCONF XML:

```
<user>
  <name>glocks</name>
  …
</user>
<user>
  <name>snowey</name>
  …
</user>
```

Non-config lists can skip key
Given at create!

NETCONF operations to insert first, last, before, after

Putting things together

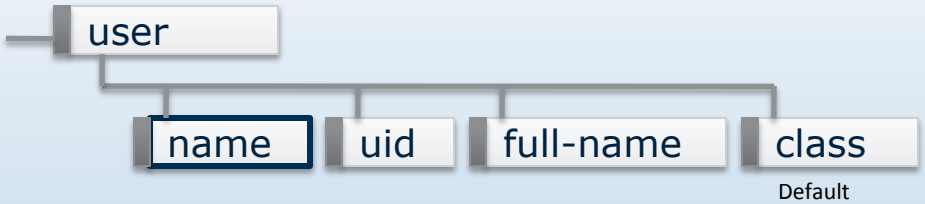
```
module acme-system {
        namespace "http://acme.example.com/system";
        prefix "acme";

        organization "ACME Inc.";
        contact "joe@acme.example.com";
        description
            "The module for entities implementing the

        revision 2007-06-09 {
            description "Initial revision.";
        }
```

```
container system {
    leaf host-name {
        type string;
        description "Hostname for this system";
    }

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

    container login {
        leaf message {
            type string;
            description
                "Message given at start of login session";
        }

        list user {
            key "name";
            leaf name {
                type string;
            }
            leaf full-name {
                type string;
            }
            leaf class {
                type string;
            }
        }
    }
}
```

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



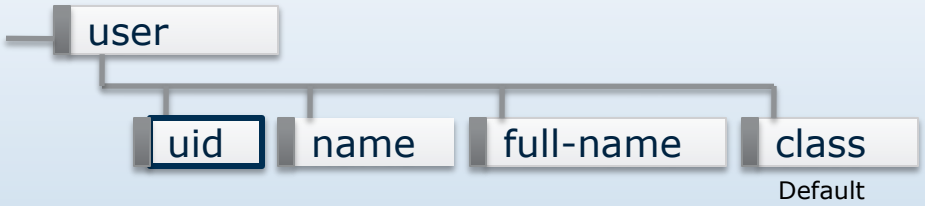
yang	1010	Yan Goode	admin
hawk	1152	Ron Hawk	oper
ling	1202	Lin Grossman	viewer

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

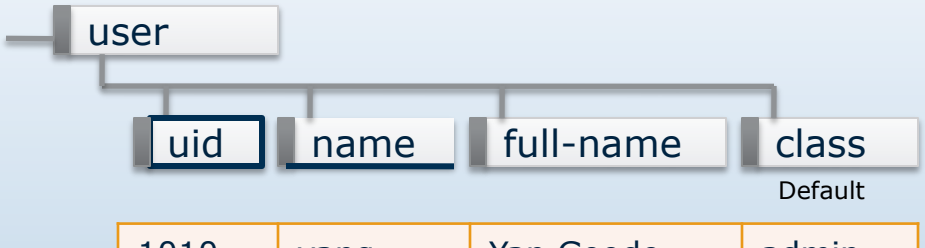


1010	yang	Yan Goode	admin
1152	hawk	Ron Hawk	oper
1202	ling	Lin Grossman	viewer

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

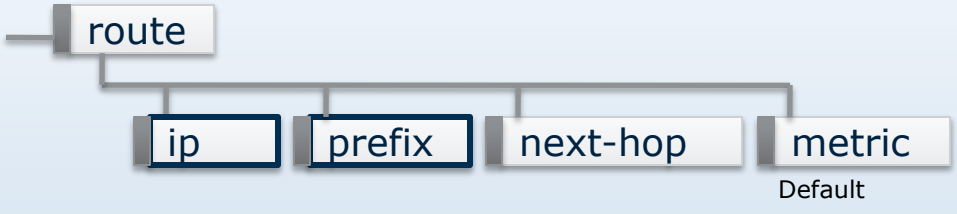
Non- key fields can also be declared unique.

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



16.40.0.0	16	220.40.0.1	20
16.42.0.0	16	82.193.16.1	40
16.42.0.0	24	82.193.16.6	50

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[ip='16.40.0.0'][prefix='16']/next-hop
 = 220.40.0.1
```

Key order significant

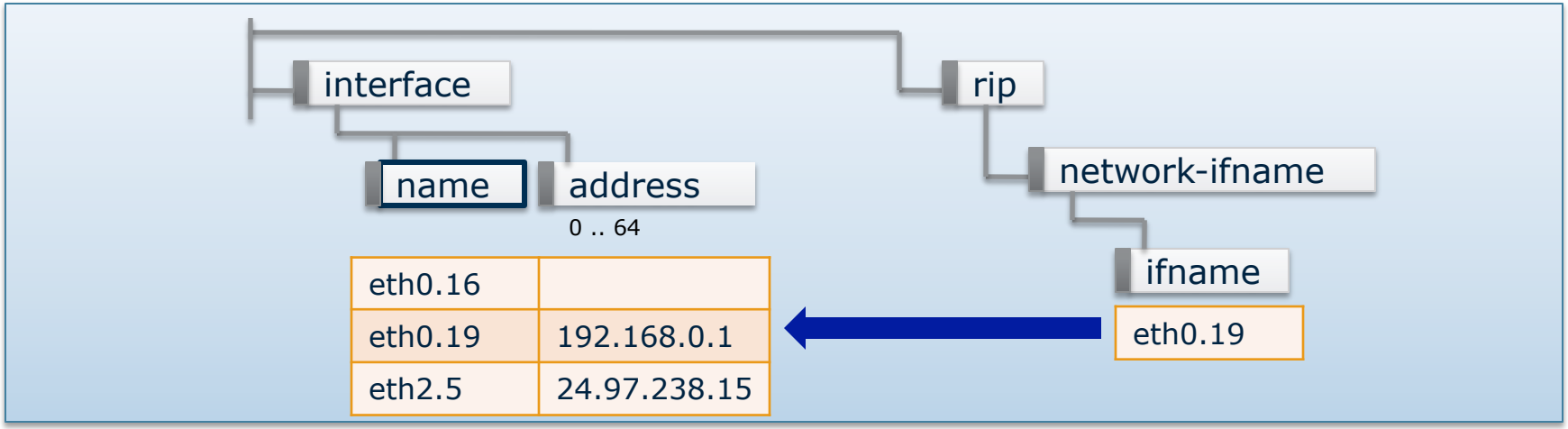
Leafref

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

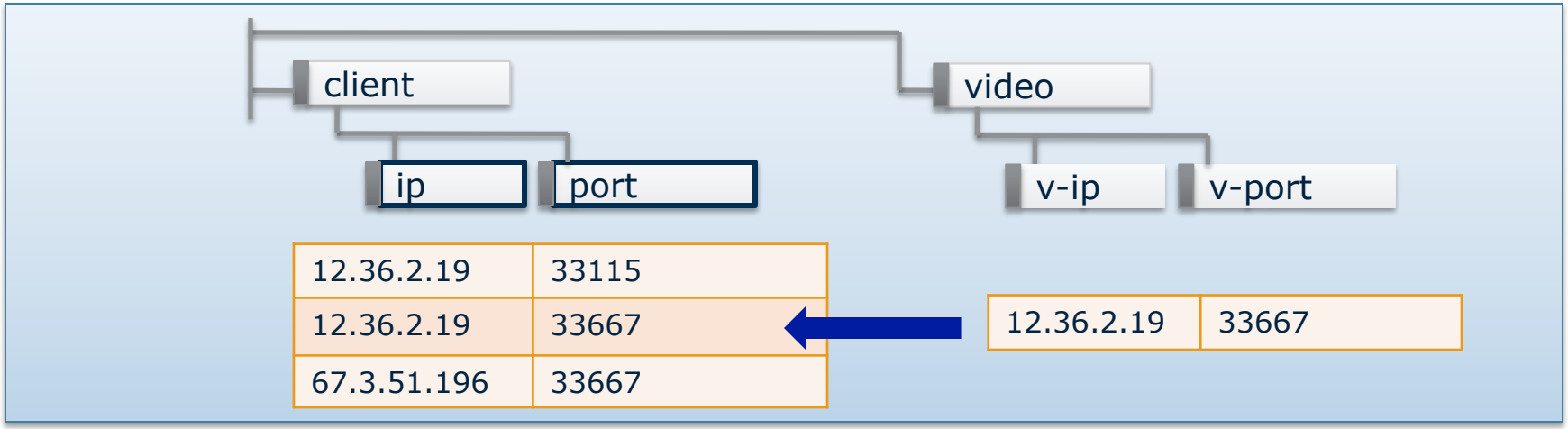
Leafref



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

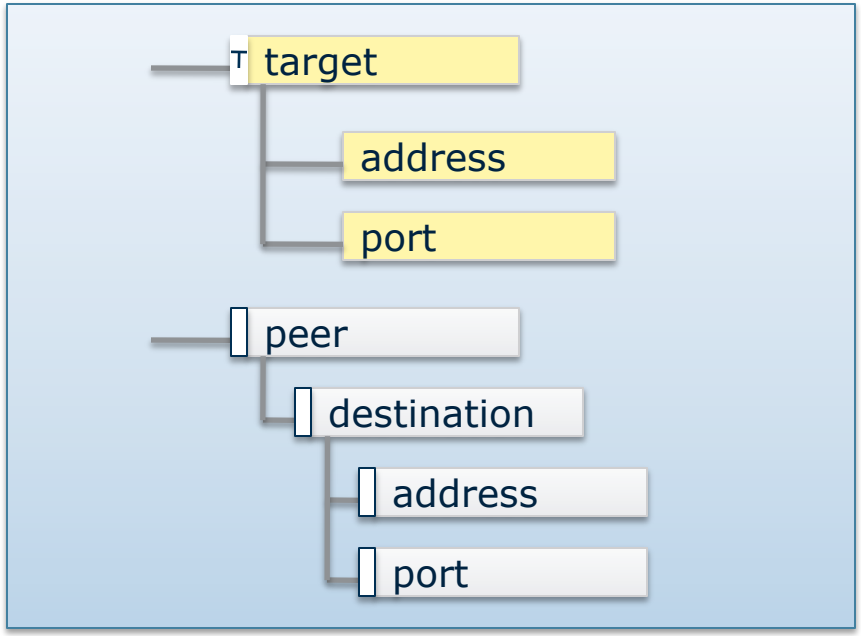
```
container video {
  leaf v-ip {
    type leafref {
      path "/client/ip";
    }
  }
  leaf v-port {
    type leafref {
      path "/client
[ip=current()/../v-ip]/port";
    }
  }
  leaf v-stream {
    type leafref {
      path "/client
[ip=current()/../v-ip]
[port=current()/../v-port]
/stream";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

```
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";


    }
  }
}
```

Grouping Statement

Think "macro expansion"

```
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;
  }
}
```



Groupings can be refined when used

Grouping Statement with Refine

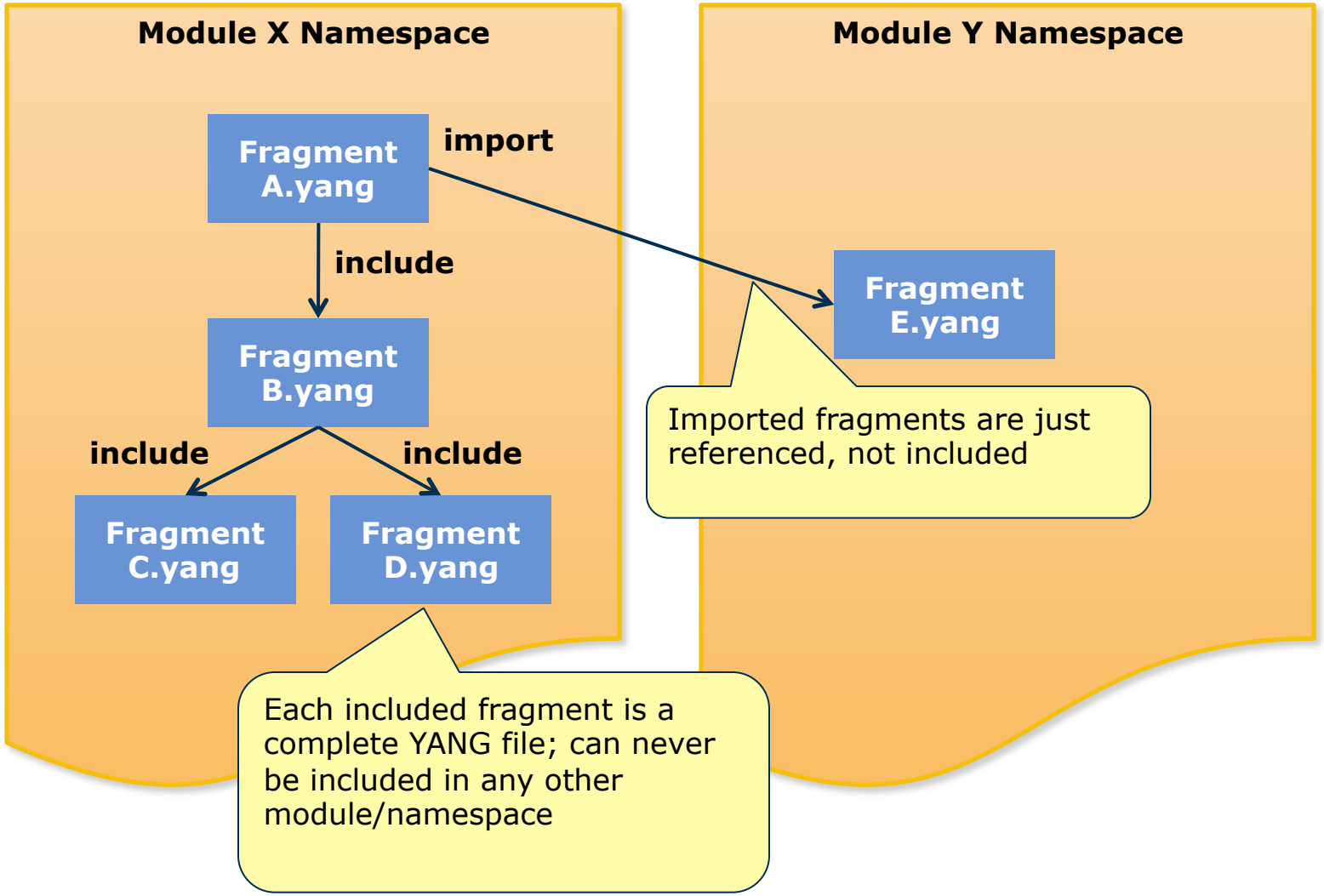
Groupings may be refined when used

```
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;
      }
    }
  }
}
```

IMPORT AND INCLUDE

Imports & Includes



Submodules

```
module acme-module {
  namespace "…";
  prefix acme;

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

```
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

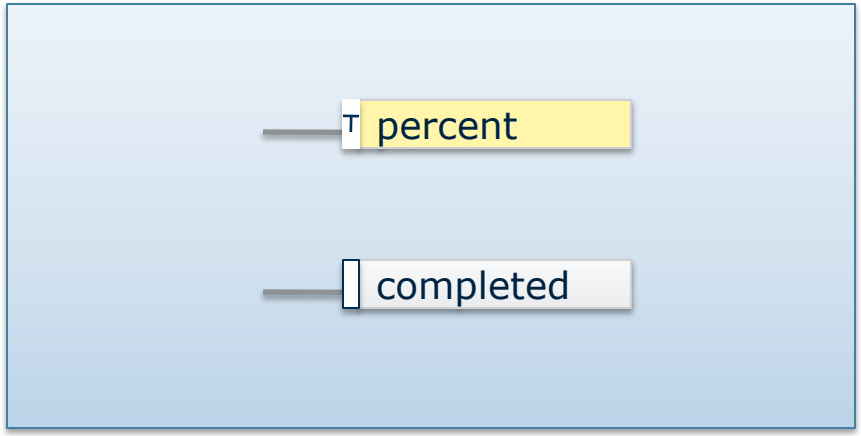
- Most YANG elements have a data type
- Type may be a base type or derived type
  - Derived types may be simple typedefs or groupings (structures)
  - There are 20+ base types to start with
- And more in modules like
  - ietf-yang-types, RFC 6021

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;
}
```



Can be scoped

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

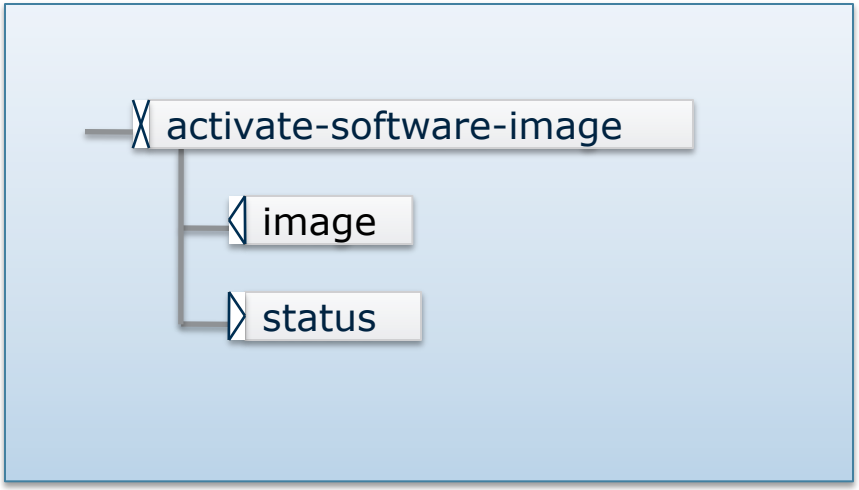
ⓘ www.rfc-editor.org/rfc/rfc6021.txt

YANG RPCs & Notifications

RPC Statement

Administrative actions with input and output parameters

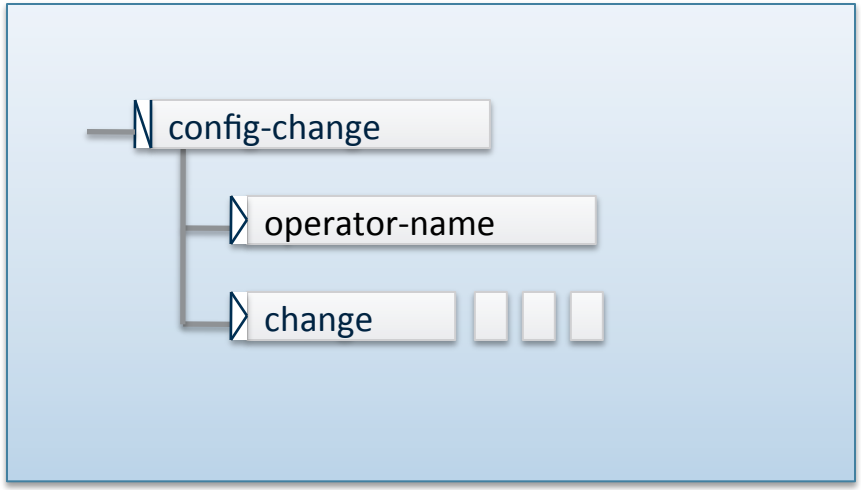
```
rpc activate-software-image {
  input {
    leaf image {
      type binary;
    }
  }
  output {
    leaf status {
      type string;
    }
  }
}
```



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

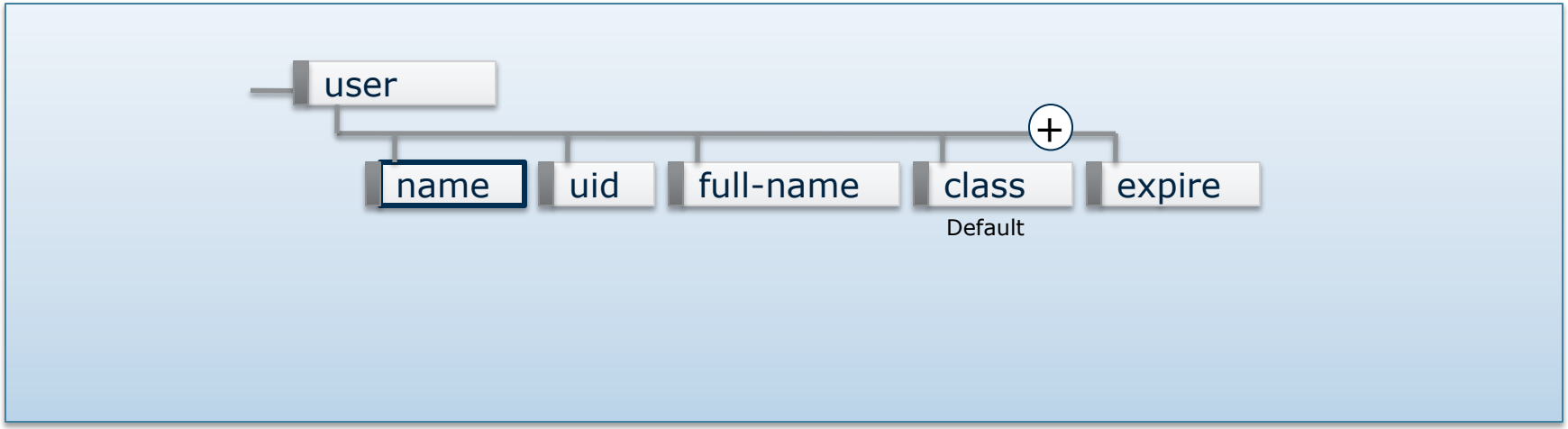
Restricts valid values by Xpath 1.0 expression

```
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

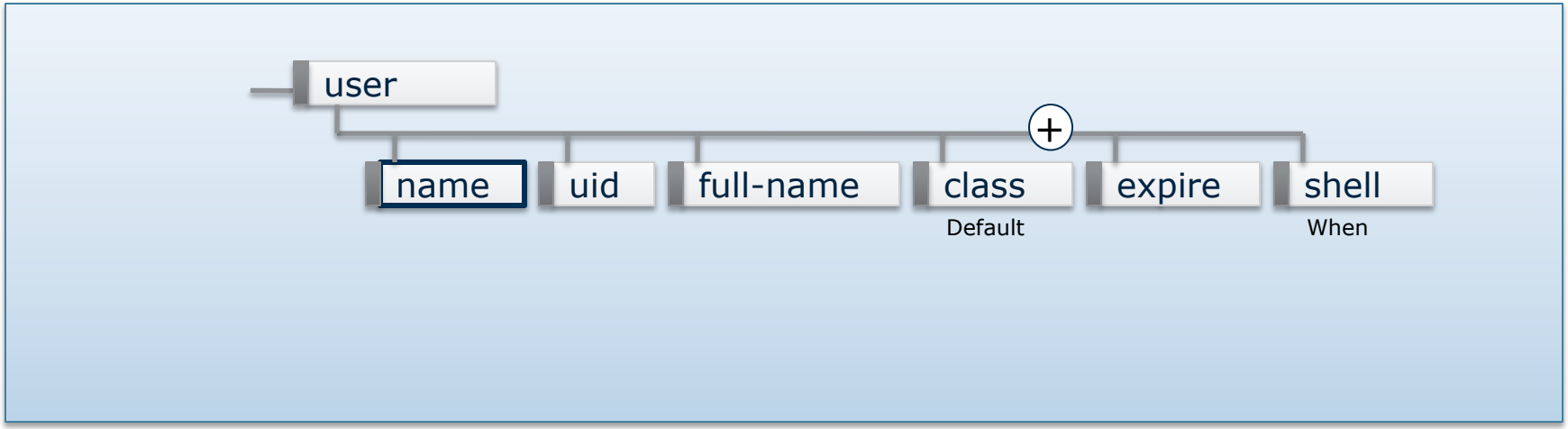
```
leaf max-weight {
  type uint32 {
    range "0..1000";
  }
  default 100;

  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;
  }
}
```

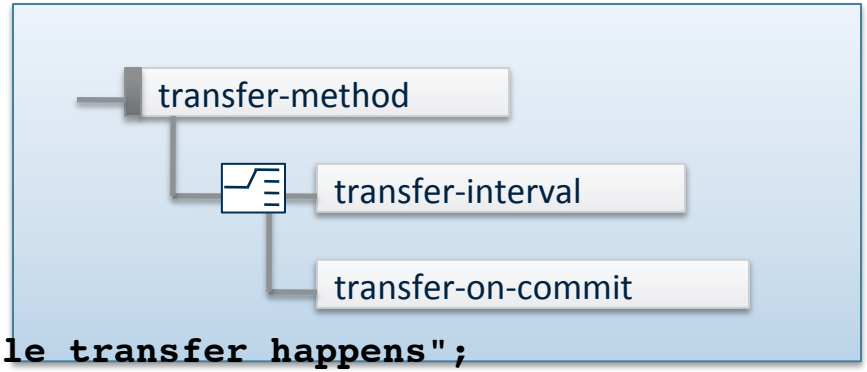
When Statement




```
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;
}
```

- Only in schema tree
- Not in the data tree or NETCONF
- Device handles deletion of “other” case when case is created.

Identity Statement

Identities for modeling families of related enumeration constants

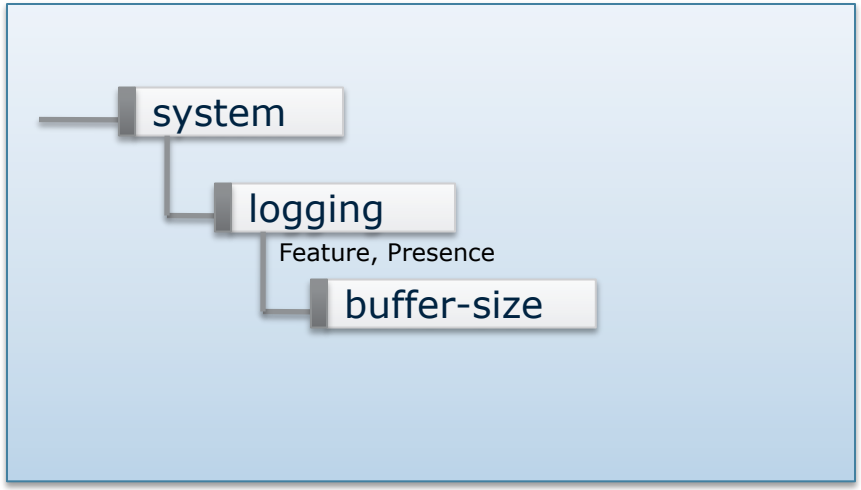
```
module phys-if {
…
  identity ethernet {
    description
  }
  identity eth-1G {
    base ethernet;
  }
  identity eth-10G {
    base ethernet;
  }
```

```
module newer {
…
  identity eth-40G {
    base phys-if:ethernet;
  }
  identity eth-100G {
    base phys-if:ethernet;
  }
…
  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 Yang Modules

- 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;
  }
}
```

Current IETF Status

IETF Activities

NETCONF Working Group

- *Rechartering in progress*
- NETCONF over TLS
- Reverse SSH

Maybe:

- RESTCONF
- DHCPv6 option for server discovery
- Efficiency extensions

NETMOD Working Group

- Interface configuration
- IP address management
- Basic routing management
- System management (i.e. MIB-II)
- SNMP Configuration

Maybe:

- Topologies
- ACLs
- OSPF

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

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

ⓘ https://www.ietf.org/iesg/directorate/yang-doctors.html

ⓘ http://www.yang-central.org/


tail-f