OF-CONFIG

Tuesday, September 6, 2016 2:25 PM

https://github.com/openvswitch/of-config -> OFCONFIG Implementation.
https://github.com/openvswitch/of-config/blob/master/README.md --> Source Code Structure.

https://tools.ietf.org/html/rfc6241 -> Client library of OF-CONFIG that comes with RYU. OF-CONFIG is an implementation of NETCONF library.

http://searchsdn.techtarget.com/tip/Can-the-NETCONF-protocol-give-OpenFlow-a-run-for-its-money - NETCONF vs OpenFlow.

https://floodlight.atlassian.net/wiki/display/floodlightcontroller/How+to+Write+Flows+for+an+OF-DPA+Switch- OFDPA support by floodlight

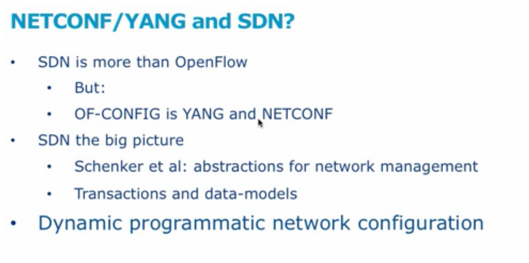
Netopeer NETCONF server implementation. https://github.com/CESNET/netopeer

https://groups.google.com/forum/#!msg/opennetworklinux/U0euINpqA7s/7x6OMz5oCwAJ - Patrick's questions on Forum.

https://rawgit.com/CESNET/libnetconf/master/doc/doxygen/html/index.html --> Documentation libnetconf library based on which NETCONF servers and clients are implemented.

http://www.netconfcentral.org/ -> Netconf tutorials, forums etc.

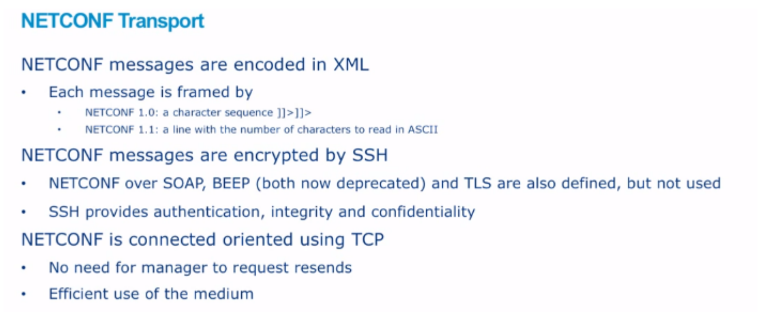
1. https://www.ietf.org/edu/documents/2012-ietf-84-netconf-yang.pdf



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

Screen clipping taken: 9/8/2016 2:15 PM



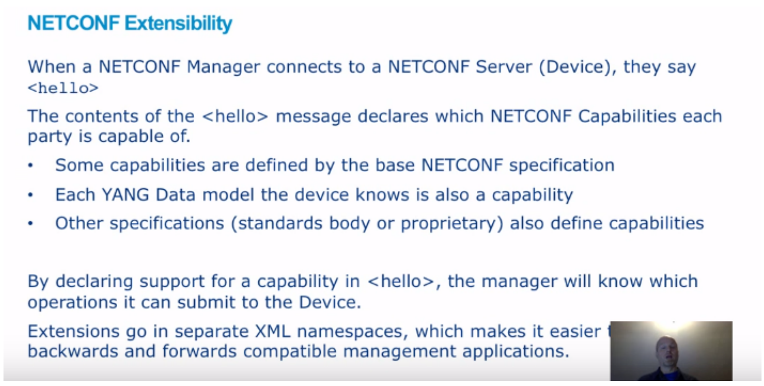
NETCONF Transport

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

Screen clipping taken: 9/8/2016 4:09 PM



NETCONF Extensibility

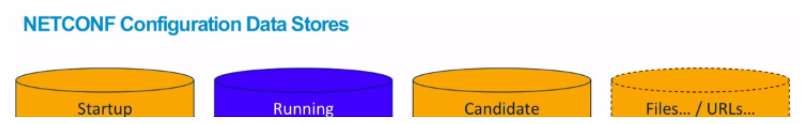
When a NETCONF Manager connects to a NETCONF Server (Device), they say <hello>

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 backwards and forwards compatible management applications.

Screen clipping taken: 9/8/2016 4:48 PM



NETCONF Configuration Data Stores

NETCONF and YANG Tutorial Part 1b: Relati

OF-CONFIG Protocol 1.2 specification.

Machine generated alternative text: It is strongly
switch described in this document via OF-CONFI
interfaces and other legacy management protoco
OpenFlgW

Machine generated alternative text: The OpenFlo
OpenFlow protocol) has been configured with va
OpenFlow Configuration Protocol (OF-CONFIG) i

Open Flow -> Operates at the Flow time-sca
OF-CONFIG -> need not work at this time-sc

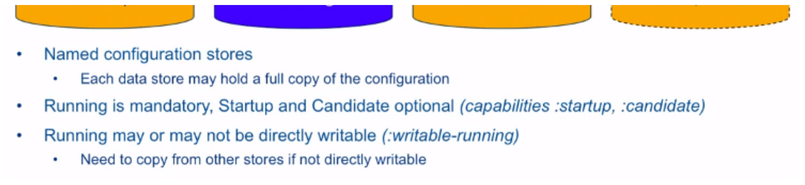
Machine generated alternative text: The service w
OpenFl Configuration Point; No assumptions are

[redacted]

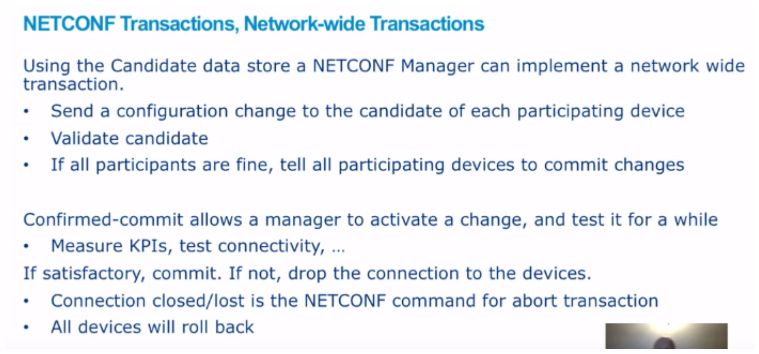
Machine generated alternative text: 5.1 OpenFlov
can act an as operational context for an OpenFlo
which may be associated with an OpenFlow Logi
configures one or more OpenFlow Capable Switc
Logical Switch An OpenFlow Logical Switch is a s
specific OpenFlow Controller. An OpenFlow Logi

Screen clipping taken: 9/9/2016 3:40 PM

Machine generated alternative text: 5.4 OpenFlov
with an OpenFlow Capable Switch and may be as
is a queuing resource of an OpenFlow Logical Sw
OpenFlow switch. 5.4.2 OpenFlow Port An OpenF
OpenFlow specification as the port component o
switch or a logical port on a physical or virtual sw
OpenFlow Logical Switches via the OpenFlow pro
describes specific switch forwarding behaviors co



- 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

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

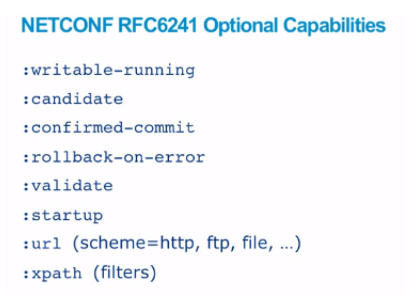
- Measure KPIs, test connectivity, …

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

Screen clipping taken: 9/8/2016 5:01 PM

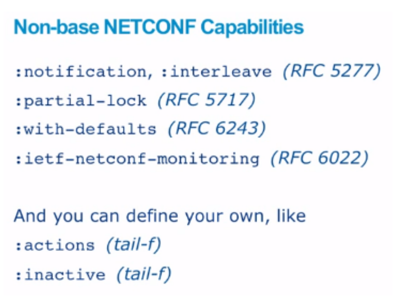
Screen clipping taken: 9/8/2016 5:04 PM



NETCONF RFC6241 Optional Capabilities

```
:writable-running
:candidate
:confirmed-commit
:rollback-on-error
:validate
:startup
:url (scheme=http, ftp, file, …)
:xpath (filters)
```

Screen clipping taken: 9/8/2016 5:24 PM



Non-base NETCONF Capabilities

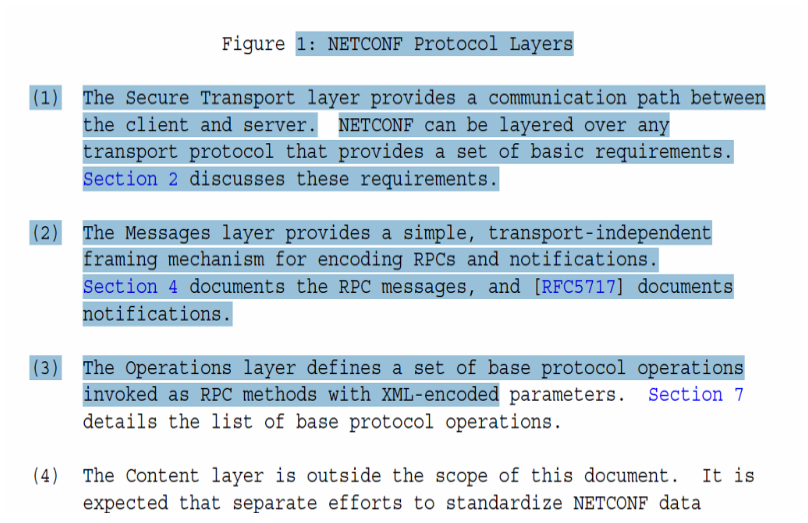
```
:notification, :interleave (RFC 5277)
:partial-lock (RFC 5717)
:with-defaults (RFC 6243)
:ietf-netconf-monitoring (RFC 6022)
```

And you can define your own, like

```
:actions (tail-f)
:inactive (tail-f)
```

Screen clipping taken: 9/8/2016 5:25 PM

Netconf protocol overview. RFC 6241 as reference



```
                 Figure 1: NETCONF Protocol Layers

   (1)  The Secure Transport layer provides a communication path between
        the client and server.  NETCONF can be layered over any
        transport protocol that provides a set of basic requirements.
        Section 2 discusses these requirements.

   (2)  The Messages layer provides a simple, transport-independent
        framing mechanism for encoding RPCs and notifications.
        Section 4 documents the RPC messages, and [RFC5717] documents
        notifications.

   (3)  The Operations layer defines a set of base protocol operations
        invoked as RPC methods with XML-encoded parameters.  Section 7
        details the list of base protocol operations.

   (4)  The Content layer is outside the scope of this document.  It is
        expected that separate efforts to standardize NETCONF data
```

for switch behavior so that implementers can per
be scalably represented in a single OpenFlow tab

Screen clipping taken: 9/9/2016 3:43 PM

2 kinds of configuration requireme

Machine generated alternative text: protocol. The
[redacted] switches. The specification of OF-CC
compatibility.

Screen clipping taken: 9/9/2016 3:44 PM

Machine generated alternative text: 1 Instantiatio

Machine generated alternative text: does not hav
one or more OpenFlow data planes and can assig
resources like managementportmaynotbeasslgne

Machine generated alternative text: 6.1.1 Instanti
capable switch is capable of hosting one or more
capable switch owns all the resources of the switc
instantiate one more OpenFlow data planes and
resources like management port may not be assig
(Connection Setup) of [1] indicates that an OpenF
the process of setting up a connection between t
three barameters that need to be configured in a
connection to the switch (described in (Il). port nu
he transport protocol to use, either TLS or port nu
provide means for configuring these parameters.

Screen clipping taken: 9/9/2016 4:04 PM

NOTE: Open Flow data planes are

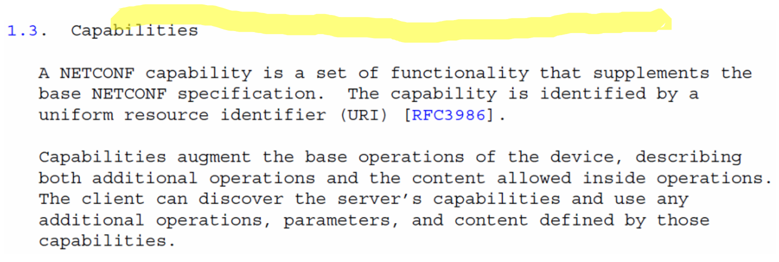
NOTE:

Machine generated alternative text: It should be
ilable and configurable via both OpenFlow and O

models will be undertaken.

Screen clipping taken: 9/10/2016 10:50 AM

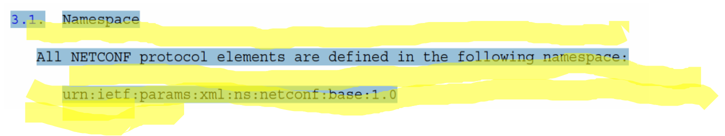
NOTE: Content layer is where data model comes into picture.


1.3. Capabilities

A NETCONF capability is a set of functionality that supplements the base NETCONF specification. The capability is identified by a uniform resource identifier (URI) [RFC3986].

Capabilities augment the base operations of the device, describing both additional operations and the content allowed inside operations. The client can discover the server's capabilities and use any additional operations, parameters, and content defined by those capabilities.

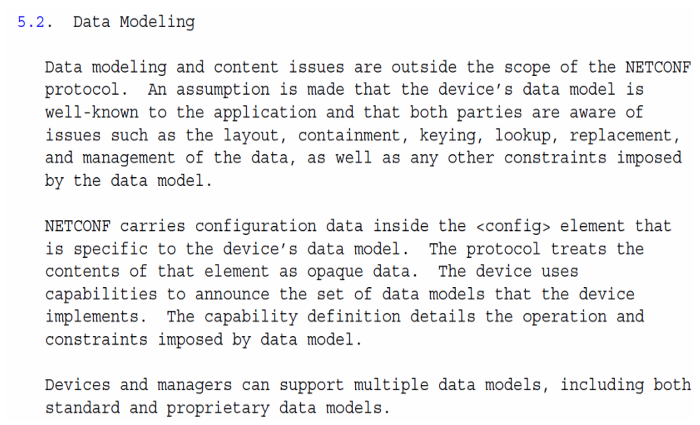
Screen clipping taken: 9/10/2016 11:05 AM


3.1. Namespace

All NETCONF protocol elements are defined in the following namespace:

urn:ietf:params:xml:ns:netconf:base:1.0

Screen clipping taken: 9/10/2016 11:44 AM

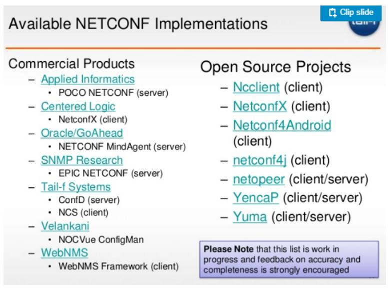

5.2. Data Modeling

Data modeling and content issues are outside the scope of the NETCONF protocol. An assumption is made that the device's data model is well-known to the application and that both parties are aware of issues such as the layout, containment, keying, lookup, replacement, and management of the data, as well as any other constraints imposed by the data model.

NETCONF carries configuration data inside the <config> element that is specific to the device's data model. The protocol treats the contents of that element as opaque data. The device uses capabilities to announce the set of data models that the device implements. The capability definition details the operation and constraints imposed by data model.

Devices and managers can support multiple data models, including both standard and proprietary data models.

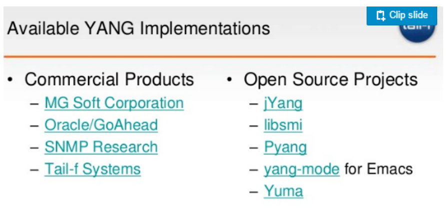
[A 30-minute Introduction to NETCONF and YANG](http://www.slideshare.net/cmoberg/a-30minute-introduction-to-netconf-and-yang) http://www.slideshare.net/cmoberg/a-30minute-introduction-to-netconf-and-yang


Available NETCONF Implementations

Commercial Products
- Applied Informatics
  - POCO NETCONF (server)
- Centered Logic
  - NetconfX (client)
- Oracle/GoAhead
  - NETCONF MindAgent (server)
- SNMP Research
  - EPIC NETCONF (server)
- Tail-f Systems
  - ConfD (server)
  - NCS (client)
- Velankani
  - NOCVue ConfigMan
- WebNMS
  - WebNMS Framework (client)

Open Source Projects
- Ncclient (client)
- NetconfX (client)
- Netconf4Android (client)
- netconf4j (client)
- netopeer (client/server)
- YencaP (client/server)
- Yuma (client/server)

Please Note that this list is work in progress and feedback on accuracy and completeness is strongly encouraged


Available YANG Implementations

- Commercial Products
  - MG Soft Corporation
  - Oracle/GoAhead
  - SNMP Research
  - Tail-f Systems
- Open Source Projects
  - jYang
  - libsmi
  - Pyang
  - yang-mode for Emacs
  - Yuma

Hereafter YUMA works NETCONF material.

NETCONF NETCONF features are identified with "capability" URIS A session starts with an exchange of capability lists by both peers Network Manager NETCONF Client Managed Device NETCONF Server The server advertises all its capabilities, assigns a session ID The

since the primary purpose of OpenFlow Configur
synchronization of the data models cand semanti

Screen clipping taken: 9/9/2016 4:34 PM

Data Model:
Full XML Schema is provided as a data-mod

Machine generated alternative text: 8.1 YANG Mc
unplementation of the OF-CONFIG data model. I
not normative for this specification. The YANG m
explanations in this section. Most of the constrai
YANG module by syntax elements already built i
using the YANG module to reduce implementatic
including those that are not expressible by the Y/

Screen clipping taken: 9/9/2016 6:17 PM

Machine generated alternative text: 8.2 Core Data
openF10w C mhgntia-' Cmtoller Coe* IN Sw•ith S
OF-CONFIG Data Model

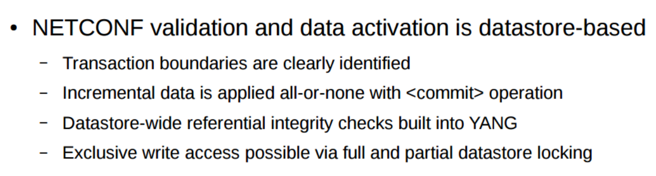
Screen clipping taken: 9/9/2016 6:42 PM

3 Scope OFÆONFG is on the following functions
more OpenFlow contro"ers The configuration of
aspects Of (e.g. up/down) Configuration of ceriti
OpenFlow Logical Switches and Open Flow Contr
Logical Switch Configuration Of a St•nall Set Of t
Functionality introduced in OF-CONFIG 1.1-1 incl

Screen clipping taken: 9/10/2016 10:34 AM client advertises the protocol versions it supports - All optional server features and supported YANG module information can be discovered by the client

NETCONF Sessions NETCONF is session-based - Secure transport required SSH/TCP is mandatory; TLS/TCP is optional - Transport must provide a user identity to the server for the session (for authorization purposes) Why use sessions? - Some procedures require multiple protocol operations - Some automatic transaction cleanup done when session is dropped or killed

Screen clipping taken: 9/14/2016 1:24 PM



- NETCONF validation and data activation is datastore-based
  - Transaction boundaries are clearly identified
  - Incremental data is applied all-or-none with <commit> operation
  - Datastore-wide referential integrity checks built into YANG
  - Exclusive write access possible via full and partial datastore locking

NETCONF data persistence is datastore-based Entire configuration datastore saved in NV-storage in implementation- specific manner - Optional "startup" capability allows access and manual control over configuration used at the next reboot

NETCONF has default statement Mandatory to implement for server developers If the leaf does not exist in the configuration datastore, the default value MUST be used instead

NETCONF has 3 types of default handling - Controlled by server and advertised in the "with-defaults" capability URI Normal retrieval of the configuration will not have any leaf that the server treats as a default leaf trim mode: if the value matches the YANG default value, the server treats the leaf as a default leaf, and will not create it in the configuration datastore explicit mode: if the value is explicitly set by the client then it is stored, even if it matches the YANG default value. report-all: the server stores all leafs in the configuration and does not remove any default leafs from the configuration datastore

• YANG All protocol operations including actions are defined in YANG rpc radiusAccServConfigReset { description " leaf server-state { config false; type enumeration { } - There could be a data object defined to monitor the status of the initialization, but: read-only status object not dual purpose

Source Code OF-CONFIG

https://github.com/openvswitch/of-config/

urce Codes Source codes are placed in the serwe the paragraphs above. Ofc- agent (I) agent . c ag implementation netconf c Ofc transAPI module C

Screen clipping taken: 9/11/2016 9:30 AM

D:\SourceCode\OFCONF\of-config\server\o

/ * overall structure providing content of this mod libnetconf * / struct transapi = { . version = 6, . ini . close ofc_transapi_close, zeet_state ofc_status_cl TRANSAPI CLBCKS ORDER DEFAULT, .data clbks & rpc_clbks = &ofc_rpc_clbks, .ns_mapping ofc_nan config_modified = . err-opt . file clbks - NULL,

https://tools.ietf.org/html/draft-ietf-netcon

NETCc*F and Configuration nodels draft -ietf -ne defines configuration data model RESTCONF con configuration Of the NET CONF and RESTCONF i supported, "hat ports the Servers listen on, authe

Screen clipping taken: 9/11/2016 5:28 PM

NOTE: Refer to ietf-netconf-server.yang in D

Screen clipping taken: 9/15/2016 11:27 AM

There are 3 standard configuration datastores, used in combinations to support different transaction and persistence behavior within the server running: mandatory representation of the current running configuration - candidate: optional scratchpad used to collect edits to apply all-or-none startup: optional non-volatile storage representation of the configuration that will be used at the next reboot. There are 4 common combinations or editing models identified with capabilities - direct (:writable-running), direct + startup (:writable-running + :startup) - indirect (:candidate), indirect + startup (:candidate + :startup)

Direct Editing Models :writable-running edit-config Automatic NV-save copy-config - Edits take effect immediately and are automatically saved to NV-storage :writable-running + :startup copy-config edit-config copy-config - Edits take effect immediately and are manually saved to NV-storage

Indirect Editing Models edit-config Automatic NV-save copy-config - Edits are collected then applied all-or-none, and automatically saved to NV-storage :candidate + :startup copy-config edit-config copy-config - Edits are collected then applied all-or-none, and manually saved to NV-storage

Other Editing Models • :writable-running + :candidate Generally not supported because there is no standard mechanism to re- sync the candidate to the current configuration and resolve collisions if the running datastore is edited independently of the candidate. Private "candidate" - Vendor-specific candidate that is not shared by all clients (like the standard candidate datastore) - Has same re-sync and collision issues as concurrent editing of running and candidate datastores

NETCONF Message Encoding The "namespace" statement value in the YANG module is used as the XML namespace for an element or qualified attribute The YANG identifier is used as the local-name module example-mod { namespace • http://example.corn/ns/example-mod"; prefix "ex"; rpc test { } <rpc <test

Remote Procedure Call (RPC) • Each NETCONF message must be a valid XML instance document • All operations use a NETCONF specific RPC mechanism • SNMP does not have a SET PDU that is allowed to return data NETCONF Client NETCONF Server • A mandatory "message-id" attribute must be included in an <rpc> element • All operations are defined with the YANG "rpc" statement

<rpc> Input Parameters The "input" statement in the "rpc" definition specifies any input parameters for the protocol operation Each data-def staternent within the input statement is encoded as a child node Of the method node. The "input" node is not encoded. module example-mod { namespace " hap ://example. corn/ns/example-mod"; prefix "ex"; rpc test { input { leat count { type int32; } <rpc message-id="101" <test xmlns='tlttp Wexample.corn/ns/example-mod" > leaf msg { type string; } <count>42</count>

Screen clipping taken: 9/15/2016 11:33 AM

Without Output • If no "output" statement is in the "rpc" definition , then an element is returned if the operation succeeds. module example-mod { namespace "http://example.com/ns/example-mo&, prefix "ex'; rpc test { input { leaf count { type uint32; } leaf msg { type string; } l/ no output section!! <rpc-reply xmlns="urn:iett:params:xml:ns:netconf:base:l.o"> <0k

<rpc-replY> Output Parameters • The "Output" statement in the "rpc" definition specifies any output parameters for the protocol operation Each data-def statement within the Output statement is encoded as a child node of the "rpc-reply" node. The "output" node is not encoded. mod"e exanvle-mod (•httpWex .corn/nslexanWe-mod': prefix •e*' ; rpc test (input count (type uint32:) msg type string;) mnput (leaf memory-errors (type uint32: <rpc-reply <memory-errors xmlns=•http://example.com/ns/example•mod" > 3

for Errors It any errors occur, then the server is expected to retum 1 or Il-lore <rpc-error> child nodes within the <rpc-reply> elernent The server is allowed to include "output' data in addition to nodes But <0k> and <rpc-error> cannot both be returned! The server is allowed to include <rpc-error> elements as descendant nodes within output data (e.g., back-end instrumentation retrieval failed) <rpc-reply contents explained later!!! rpc-reply>

Base Protocol Operations Description Name get-config Retrieve some or all of a config datastore edit-config Edit some or all of a config datastore copy-config Copy contents from/to a config datastore delete-config Remove all contents of a config datastore lock Start exclusive write access of a datastore unlock Stop exclusive write access of a datastore Retrieve config and/or state data get close-session Cause your session to close kill-session Force another session to close NETCONF/YANG Tutorial (c) 2014 YumaWorks, Inc.

<edit-confiq> 5 edit operations (create, merge, replace, delete, remove) Parameter target default- operation test-option error-option config Description Configuration datastore to edit Default edit mode; Default (merge) Values (merge, replace, none) Values (test-then-set, set, test-only); D: (set) Values (stop-on-error, continue-on-error, rollback-on-error); Default (stop-on-error) Portion of the configuration to edit

Using the candidate datastore Edit phase <edit-config> used to collect edits in the (shared scratchpad) candidate datastore Global <lock> and <unlock> should be used to prevent multi-client collisions Commit phase <commit> used to apply scratchpad all-or-none to the running datastore Commit with rollback <commit> w/ <confirmed> parameter for automatic rollback unless confirming <commit> sent by client <cancel-commit> used to force manual rollback Cancel an edit <discard-changes> used to remove any edits from the candidate and re-sync with the running datastore

Screen clipping taken: 9/15/2016 6:14 PM

Using the running datastore Edit phase <edit-config> used directly on the running datastore Global <lock> and <unlock> should be used to prevent multi-client collisions Subtree-specific <partial-lock> and <partial-unlock> may be supported by the server Commit phase - There is no commit phase; The edits take effect immediately Commit with rollback The server may support the "rollback-on-error" <error-option> to support all-or-none Cancel an edit - There is no way to cancel an edit

Screen clipping taken: 9/15/2016 6:14 PM