
cisco Live!
TURN
IT
UP
#CiscoLive

The bridge to possible

NETCONF/YANG-Only Provision, Configure, Monitor

Mike Mikhail, Customer Delivery Architect
@MikeMikhail mamikhai@cisco.com

BRKPRG-1373

Agenda

- YANG Models, Config & Oper data
- NETCONF Protocol & Tools
- Zero-Touch Provisioning
- Initial “golden” config
- Deployment
- Functional verification

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Agenda

- ➢ YANG Models, Config & Oper data
- NETCONF Protocol & Tools
- Zero-Touch Provisioning
- Initial “golden” config
- Deployment
- Functional verification

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

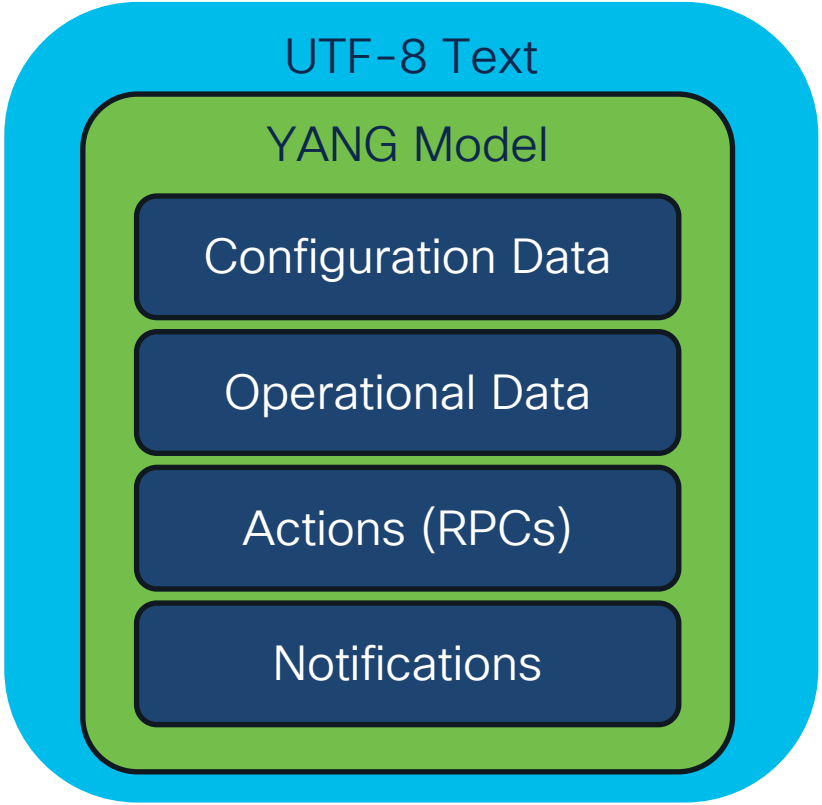
YANG Models, Config & Oper Data

YANG Language

YANG is a data modeling language

"Yet Another Next Generation" – really!

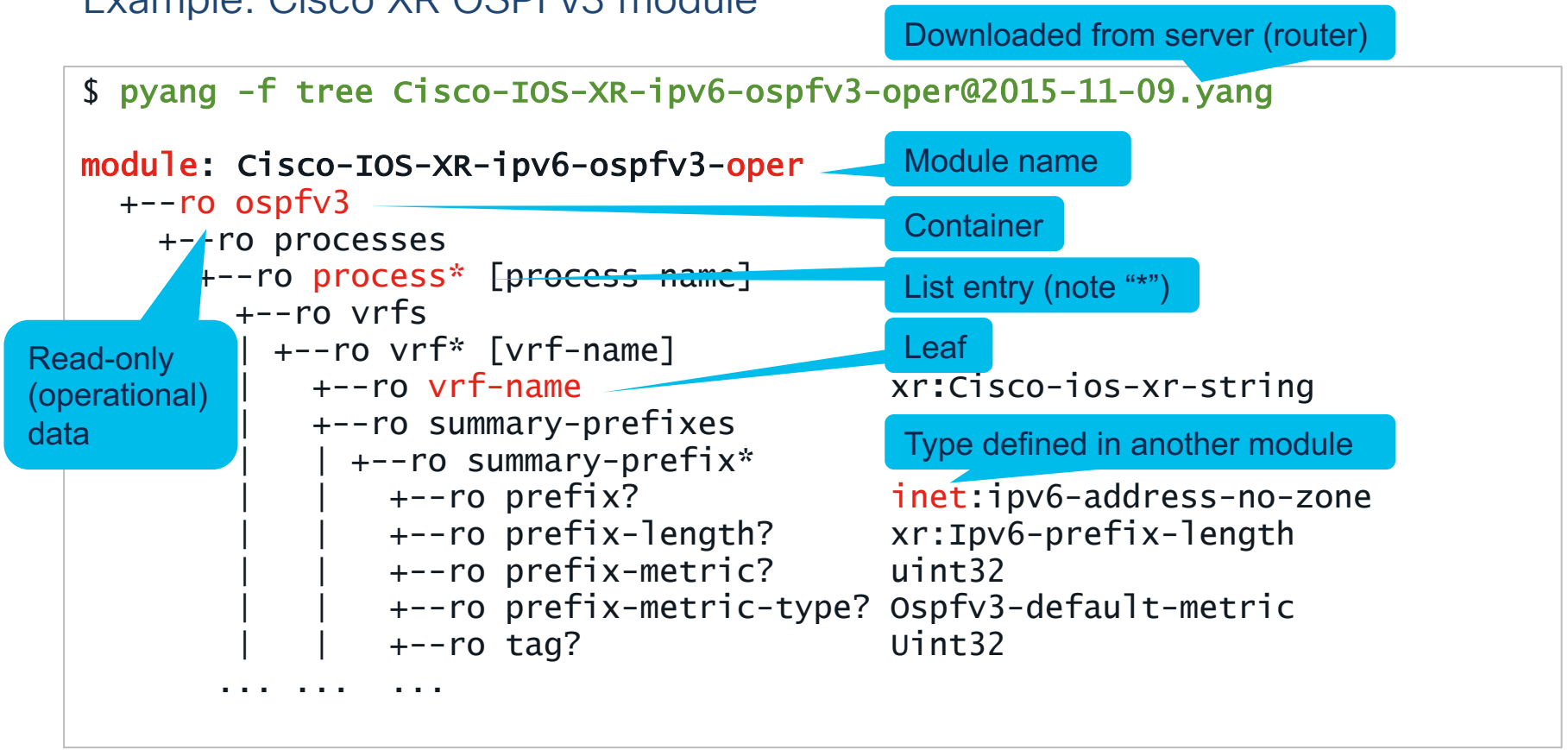
- Readable by humans and machines
- Hierarchical, modular, and extensible
- https://datatracker.ietf.org/wg/netmod/documents/



© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

YANG data models are hierarchical (trees)

Example: Cisco XR OSPFv3 module



© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

YANG models define actions (RPCs)

- Example: IETF NETCONF module

```
$ pyang -f tree ietf-netconf@2011-06-01.yang

module: ietf-netconf
  rpcs:
    +---x get-config
    |  +---w input
    |  |  +---w source
    |  |  |  +---w (config-source)
    |  |  |     +--:(candidate)
    |  |  |     |  +---w candidate?   empty {candidate}?
    |  |  |     +--:(running)
    |  |  |     |  +---w running?     empty
    |  |  |     +--:(startup)
    |  |  |        +---w startup?     empty {startup}?
    |  |  +---w filter?
    |  +--ro output
    |     +--ro data?
    +---x edit-config
   ...
```

Module name

RPC definition

Config data store types

Choice (select one of the following cases:)

Only valid if device supports “candidate datastore” feature

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

How and where to find YANG models

Where to find YANG models

- Retrieve from the YANG server, via NETCONF <get-schema> or via RESTCONF HTTP GET
- GitHub
  - https://github.com/YangModels/yang/
    - IETF, IEEE, Broadband Forum, and MEF draft and standard models
    - Vendor models for Cisco, Ciena, Huawei, and Juniper
  - https://github.com/openconfig/public/tree/master/release/models
    - OpenConfig models

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Agenda

✓YANG Models, Config & Oper data

➢NETCONF Protocol & Tools

- Zero-Touch Provisioning
- Initial “golden” config
- Deployment
- Functional verification

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

NETCONF
Protocol & Tools

NETCONF Protocol

What is NETCONF

Network management protocol

- Uses RPC's with XML encoding, usually over SSH, to:
  - Read/write/edit configuration
  - Read operational state and parameters
  - Instruct administrative actions
- Session-based
- Transactional: all or nothing (atomic) with confirmation/test/rollback
- Extensible through YANG data model augmentation
- Concept of config data stores: Candidate, Running, Start-up
- IETF https://datatracker.ietf.org/wg/netconf/documents/

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Netconf Operations

For configuration and operational data

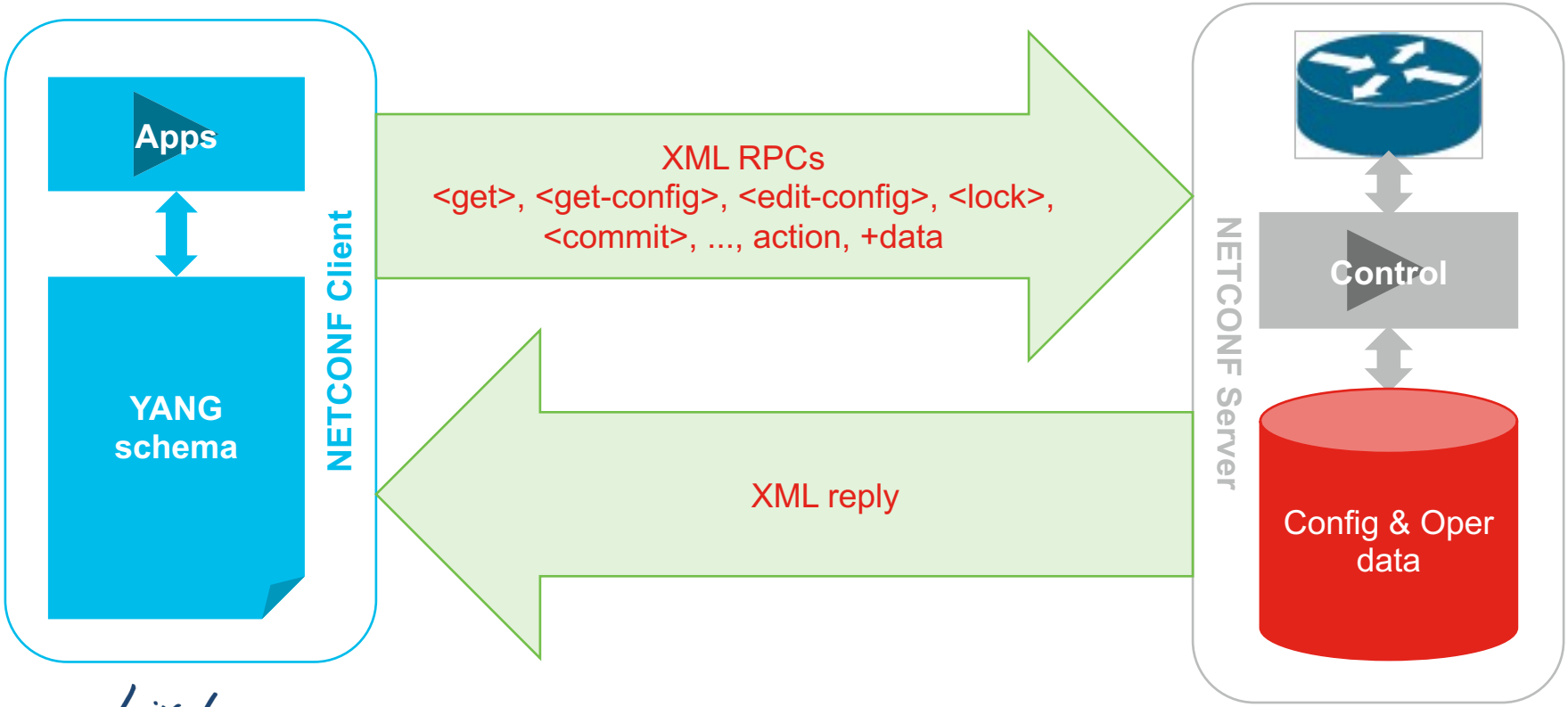
```
<get-config>
    source: running/candidate/startup
    filter: subtree/xpath match
    options: with-defaults

<get>
    filter: subtree/xpath match

<edit-config>
    target: running/candidate/startup
    operation:
        merge/replace/create/delete/remove
    options: test-option/error-option
```

```
<copy-config>
    target: startup/running/url/...
    source: startup/running/url/...

<delete-config>
    target: startup/url

<lock>/<unlock>
    target: running/candidate/startup

<close-session>

<kill-session>
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

NETCONF Mechanism

Data requests/manipulation, and actions [RFC 6241]



© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Getting YANG Modules

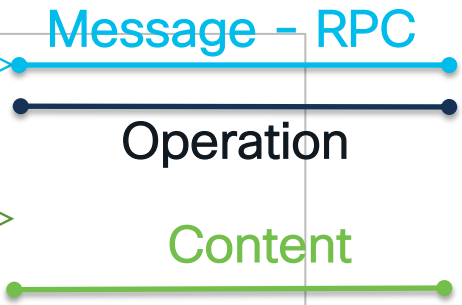
```
<rpc message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-schema xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
    <identifier>ietf-interfaces</identifier>
  </get-schema>
</rpc>
```

```
<rpc-reply message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
    module ietf-interfaces {
        //ietf-interfaces yang module contents here ...
    }
  </data>
</rpc-reply>
```

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

NETCONF Stack

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source><running/><source>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet3</name>
        </interface>
      </interfaces>
    </filter>
  </get-config>
</rpc>
```



© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

<get-config> Response

Message - RPC

Content

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>GigabitEthernet3</name>
        <description>tor1 e1/47</description>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>172.200.200.100</ip>
            <netmask>255.255.255.0</netmask>
          </address>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"></ipv6>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

<edit-config>

```
<rpc message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet3</name>
            <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
              <address>
           <ip>1.1.1.1</ip>
           <netmask>255.255.255.0</netmask>
             </address>
          </ipv4>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

Message - RPC

Operation

Content

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

<edit-config> Response

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="102">
  <ok/>
</rpc-reply>
```

Message - RPC

Content

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

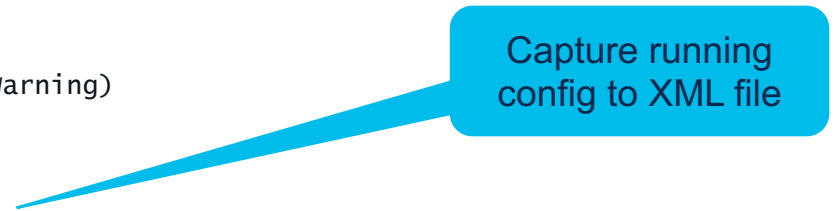
ncclient
Python Library for NETCONF

NCCLIENT: Introduction

https://pypi.python.org/pypi/ncclient

- A Python library for NETCONF client-initiated RPCs and NETCONF server events
- “pip install ncclient”
- Python2 and Python3 (>=3.5); https://pypi.python.org/pypi/ncclient

```
$ ./get-config.py host username password

import sys, os, warnings, time
warnings.simplefilter("ignore", DeprecationWarning)
from ncclient import manager

today=time.strftime("%y%m%d")

def get_config(host, user, password):
    with manager.connect(host=host, port=830, username=user, password=password, hostkey_verify=False) as m:
        c = m.get_config(source='running').data_xml
        with open("config-" + today + "-" + "%s.xml" % host, 'w') as f:
            f.write(c)

if __name__ == '__main__':
    get_config(sys.argv[1], sys.argv[2], sys.argv[3])
```



© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

NCCLIENT RPC <get> Example

Capture node running config to file

```
$ ./get-config.py 10.101.124.1 cisco cisco

$ ll config-170502-10.101.124.1.xml
-rw-rw-r-- 1 cisco cisco 65444 May  2 13:43 config-170502-10.101.124.1.xml

$ more config-170502-10.101.124.1.xml
<?xml version="1.0" encoding="UTF-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <crypto xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-crypto-sam-cfg">
   <ssh xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-crypto-ssh-cfg">
    <server>
     <v2/>
     <netconf>830</netconf>
     <netconf-vrf-table>
      <vrf>
       <vrf-name>default</vrf-name>
       <enable/>
      </vrf>
     </netconf-vrf-table>
    </server>
   </ssh>
  </crypto>
  <mac-sec-keychains xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-lib-keychain-macsec-cfg">
   <mac-sec-keychain>
    <chain-name>CISCO</chain-name>
...SNIPPED...
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Agenda

✓YANG Models, Config & Oper data

✓NETCONF Protocol & Tools

➢Zero-Touch Provisioning

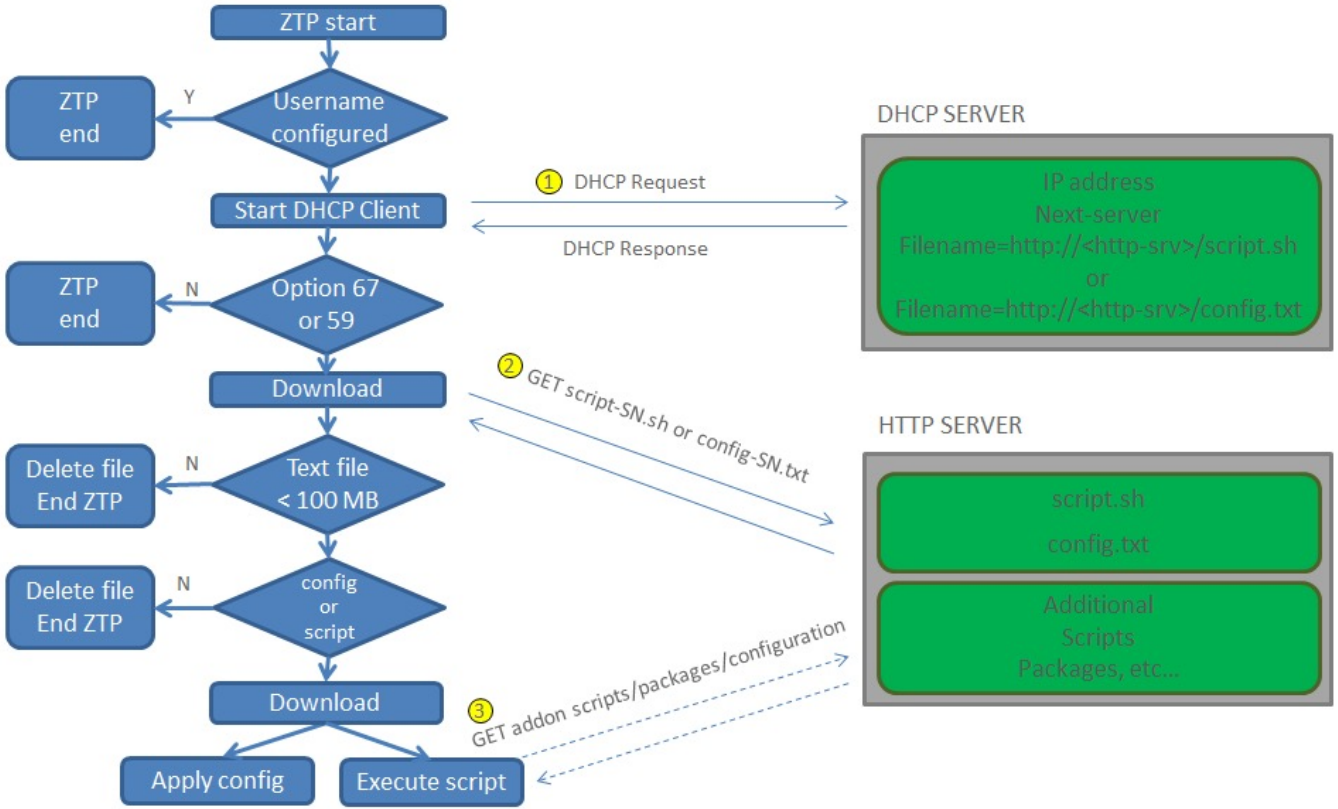
- Initial “golden” config
- Deployment
- Functional verification

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Zero-Touch Provisioning

cisco Live!

ZTP Tools & Process: DHCP & HTTP



© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

ZTP: Outcomes and Process

- Upgrade: Download and install packages
- Script: Download and run a shell script
- Configure: Download and commit a config file
- You need to:
  - ✓Test
  - ✓Verify: Error checking – build in process
  - ✓Validate: Validate the node functionality [control and forwarding] – automate

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

ZTP: Prepare node for NETCONF

```
cisco@mamikhai-ubuntu:~$ more /var/www/html/ztp/pe125-script.sh
#!/bin/bash

source ztp_helper.sh
config_file='/disk0:/ztp/tmp/config.txt'
config_log='/disk0:/ztp/customer/config-log.txt'

/bin/touch $config_log

if [ -f $config_file ]; then
  /bin/rm -f $config_file
else
  /bin/touch $config_file
fi

echo 'username cisco' >> $config_file
echo ' group root-lr' >> $config_file
echo ' group cisco-support' >> $config_file
echo ' secret cisco' >> $config_file
echo 'interface MgmtEth0/RP0/CPU0/0' >> $config_file
echo ' ipv4 address 192.168.30.125 255.255.255.0' >> $config_file
echo ' no shutdown' >> $config_file
echo 'netconf-yang agent' >> $config_file
echo ' ssh' >> $config_file
echo 'ssh server v2' >> $config_file
echo 'ssh server netconf vrf default' >> $config_file
```

File start:
#!/bin/bash == script

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

ZTP: Prepare node for NETCONF - Continued

```
xrapply_with_reason 'Initial ZTP config' $config_file


if [[ -z $(xrcmd "show crypto key mypubkey rsa") ]]; then
    echo "1024" | xrcmd "crypto key generate rsa"
else
    echo -ne "yes\n 2048\n" | xrcmd "crypto key generate rsa"
fi


xrcmd 'show running-config' >> $config_log
xrcmd 'show configuration failed' >> $config_log
xrcmd 'show crypto key mypubkey rsa' >> $config_log
```

+ can prep ssh ☺ [for NETCONF]

```
cisco@mamikhai-ubuntu:~$ ll /var/www/html/ztp/
total 40
drwxr-xr-x 2 root root 4096 Oct 22 08:52 ./
drwxr-xr-x 3 root root 4096 Aug  7 11:26 ../
-rw-r--r-- 1 root root  513 Oct 18 19:46 pe125-config-initial.txt
-rw-r--r-- 1 root root 8503 Oct 18 08:14 pe125-config.txt
.
-rw-r--r-- 1 root root 1090 Oct 22 08:52 pe125-script.sh
```

Served by HTTP server

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

DHCP Server: Initial Parameters & Pointer

```
cisco@mamikhai-ubuntu:~$ more /etc/dhcp/dhcpd.conf
.
# option definitions common to all supported networks...
option domain-name "cisco.com";
option domain-name-servers 171.70.168.183, 64.102.6.247;

default-lease-time 600;
max-lease-time 7200;

subnet 192.168.30.0 netmask 255.255.255.0 {
}

.


host PE125 {
  hardware ethernet 00:50:56:85:da:18;
  fixed-address 192.168.30.125;
  option routers 192.168.30.1;
  filename "http://192.168.30.101/ztp/pe125-script.sh";
}
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

ZTP Node Ready for NETCONF

```
<?xml version="1.0" encoding="UTF-8"?><data
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
.
   <interface-configuration>
    <active>act</active>
    <interface-name>MgmtEth0/RP0/CPU0/0</interface-name>
    <ipv4-network xmlns="http://cisco.com/ns/yang/Cisco-IOS-
XR-ipv4-io-cfg">
     <addresses>
      <primary>
       <address>192.168.30.125</address>
       <netmask>255.255.255.0</netmask>
      </primary>
     </addresses>
    </ipv4-network>
   </interface-configuration>
.
  <ssh xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-crypto-
ssh-cfg">
   <server>
    <v2/>
    <netconf-vrf-table>
     <vrf>
      <vrf-name>default</vrf-name>
      <enable/>
     </vrf>
    </netconf-vrf-table>
   </server>
  </ssh>
.
```

```
<netconf-yang xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-
man-netconf-cfg">
   <agent>
    <ssh>
     <enable/>
    </ssh>
   </agent>
  </netconf-yang>
.
  <aaa xmlns="http://tail-f.com/ns/aaa/1.1">
   <authentication>
    <users>
     <user>
      <name>cisco</name>
      <uid>9000</uid>
      <gid>100</gid>
      <password>$1$glU0$OEyQD/4ePFuNrZ2d0xtQo0</password>
      <ssh_keydir>/var/confd/homes/cisco/.ssh</ssh_keydir>
      <homedir>/var/confd/homes/cisco</homedir>
     </user>
    </users>
    <groups>
     <group>
      <name>aaa-r</name>
      <gid>100</gid>
      <users>%%__system_user__%%</users>
     </group>
     <group>
      <name>admin-r</name>
      <gid>100</gid>
      <users>%%__system_user__%%</users>
.
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Agenda

- ✓YANG Models, Config & Oper data
- ✓NETCONF Protocol & Tools
- ✓Zero-Touch Provisioning
- ➢Initial “golden” config
- Deployment
- Functional verification

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Initial ”golden” config

https://github.com/mikemikhail/NETCONF-ZTP-config-retrieval-and-push

Get Config, Schema, Modules

Using NETCONF XML RPCs with filters

get-config.py:

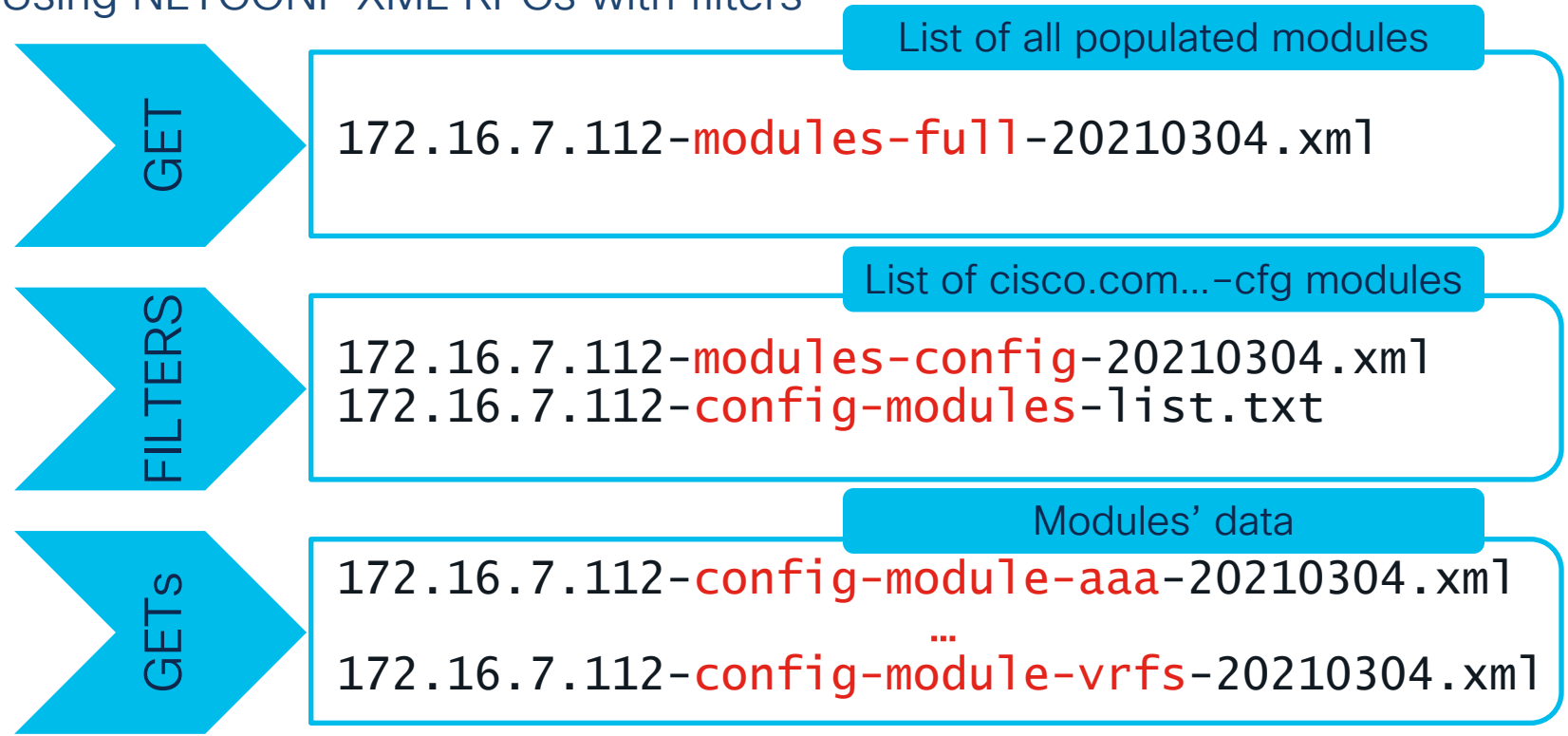
- Read and save full config => 172.16.7.112-config-full-20210304.xml
- list of exposed models => 172.16.7.112-modules-full-20210304.xml
- **list of config native models => 172.16.7.112-modules-config-20210304.xml**
- list of "other" models => 172.16.7.112-modules-non-config-20210304.xml
- **filters to retrieve each module => 172.16.7.112-config-filter-...-20210304.xml**
- **retrieved config of each module 172.16.7.112-config-module-...-20210304.xml**

https://github.com/mikemikhail/NETCONF-ZTP-config-retrieval-and-push

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Get Config, Schema, Modules

Using NETCONF XML RPCs with filters



© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

The Data

Read: *host*-config-full-*date*.xml

```
cisco@ubuntu-89:/opt/live/ztp$ more 172.16.7.112-config-full-20210304.xml
<?xml version="1.0" encoding="UTF-8"?><data
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <host-names xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-cfg">
   <host-name>PE112</host-name>
  </host-names>
  <mpls-oam xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-mpls-oam-cfg">
   <enable-oam/>
  </mpls-oam>
  <ssh xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-crypto-ssh-cfg">
   <server>
    <v2/>
    <netconf-vrf-table>
     <vrf>
      <vrf-name>default</vrf-name>
      <enable/>
     </vrf>
    </netconf-vrf-table>
   </server>
  </ssh>
  .
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Read: *host*-modules-config-*date*.xml

```
cisco@ubuntu-89:/opt/live/ztp$ more 172.16.7.112-modules-config-20210304.xml
  <host-names xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-cfg" />
  <mpls-oam xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-mpls-oam-cfg" />
  <ssh xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-crypto-ssh-cfg" />
  <clock xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-infra-clock-linux-cfg" />
  <mpls-lsd xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-mpls-lsd-cfg" />
  <cdp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cdp-cfg" />
  <parser xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-parser-cfg" />
  <netconf xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-man-xml-ttyagent-cfg" />
  <router-static xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ip-static-cfg" />
  <groups xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-group-cfg" />
  <mpls-te xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-mpls-te-cfg" />
  <isis xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-clns-isis-cfg" />
  <ipv4-acl-and-prefix-list xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-acl-cfg"
/>
  <telemetry-model-driven xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-telemetry-model-
driven-cfg" />
  <rsvp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ip-rsvp-cfg" />
  <ipv4-network-global xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-ma-cfg" />
  <netconf-yang xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-man-netconf-cfg" />
  <bgp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-bgp-cfg" />
  .
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Construct: *host*-config-filter-vrfs-*date*.xml

```
cisco@ubuntu-89:/opt/live/ztp$ more 172.16.7.112-config-filter-vrfs-20210304.xml
<filter>
  <vrfs xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-rsi-cfg" />
</filter>
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Read: *host*-config-module-vrfs-*date*.xml

```
cisco@ubuntu-89:/opt/live/ztp$ more 172.16.7.112-config-module-vrfs-20210304.xml
<config>
  <vrfs xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-rsi-cfg">
   <vrf>
    <vrf-name>JOE</vrf-name>
    <create/>
    <afs>
     <af>
      <af-name>ipv4</af-name>
      <saf-name>unicast</saf-name>
      <topology-name>default</topology-name>
      <create/>
      <bgp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-bgp-cfg">
       <import-route-targets>
        <route-targets>
         <route-target>
          <type>as</type>
          <as-or-four-byte-as>
           <as-xx>0</as-xx>
           <as>65001</as>
           <as-index>65400</as-index>
           .
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

List: *host*-config-module-list-*date*.txt

```
cisco@ubuntu-89:/opt/live/BRKPRG-1373$ more 172.16.7.112-config-module-list-20210304.txt
172.16.7.112-config-module-host-names-20210304.xml
172.16.7.112-config-module-mpls-oam-20210304.xml
172.16.7.112-config-module-ssh-20210304.xml
172.16.7.112-config-module-clock-20210304.xml
172.16.7.112-config-module-mpls-lsd-20210304.xml
172.16.7.112-config-module-cdp-20210304.xml
172.16.7.112-config-module-router-static-20210304.xml
172.16.7.112-config-module-mpls-te-20210304.xml
172.16.7.112-config-module-isis-20210304.xml
172.16.7.112-config-module-ipv4-acl-and-prefix-list-20210304.xml
172.16.7.112-config-module-telemetry-model-driven-20210304.xml
172.16.7.112-config-module-rsvp-20210304.xml
172.16.7.112-config-module-ipv4-network-global-20210304.xml
172.16.7.112-config-module-netconf-yang-20210304.xml
172.16.7.112-config-module-bgp-20210304.xml
172.16.7.112-config-module-mpls-ldp-20210304.xml
172.16.7.112-config-module-call-home-20210304.xml
172.16.7.112-config-module-aaa-20210304.xml
172.16.7.112-config-module-keychains-20210304.xml
172.16.7.112-config-module-interface-configurations-20210304.xml
172.16.7.112-config-module-routing-policy-20210304.xml
.
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

The Code

get-config.py: Parameters

```
#! /usr/bin/env python
# Reads XR device configuration to a file
# mamikhai@cisco.com

import sys, os, warnings, re
warnings.simplefilter("ignore", DeprecationWarning)
from ncclient import manager
import time
from datetime import datetime

# target NETCONF server
server = '172.16.7.112'

logfile = 'get_config.log'
tracefile = 'get_config.trace.log'

cap_cnf_file = server + '-cap-cnf-' + str(datetime.now().strftime('%Y%m%d')) + '.xml'


user ='cisco'
password ='cisco'
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

get-config.py: Read, File, Make Lists!

```
if __name__ == '__main__':
    with manager.connect(host=server, port=830, username=user, password=password, \
        look_for_keys=False, hostkey_verify=False, \
        unknown_host_cb=my_unknown_host_cb) as m:


        config_full_file = server + '-config-full-' + \
               str(datetime.now().strftime('%Y%m%d')) + '.xml'
        config = m.get_config(source='running').data_xml
        with open(config_full_file, 'w') as f:
            f.write(config)
        f.close()


        modules_full_file = server + '-modules-full-' + \
               str(datetime.now().strftime('%Y%m%d')) + '.xml'
        modules_config_file = server + '-modules-config-' + \
               str(datetime.now().strftime('%Y%m%d')) + '.xml'
        modules_non_config_file = server + '-modules-non-config-' + \
               str(datetime.now().strftime('%Y%m%d')) + '.xml'
.
```

Get full config

Files of lists of YANG models

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

get-config.py: Read, File, Make Lists! -Cont.

```
.
        with open(modules_full_file, 'w') as modules_full, \
               open(modules_config_file, 'w') as modules_config, \
               open(modules_non_config_file, 'w') as modules_non_config,\
               open(config_full_file, 'r') as config_full:

            for line in config_full:
                if re.match(r'^  <[^/]', line):
                    modules_full.write(line)
                    if re.match(r'^  <[^/].*cisco\.com.*cfg', line):
                        modules_config.write(re.sub('>', ' />', line))
                    else:
                        modules_non_config.write(line)

        config_list_file = server + '-config-module-list-' + \
               str(datetime.now().strftime('%Y%m%d')) + '.txt'
```

Find module xlmns

Xlmns of native models

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

get-config.py: Loop Through, Read, File

```
        with open(modules_config_file, 'r') as modules, open(config_list_file, 'w')  \
                as config_list:
            module = modules.readline()
            while module:
                filter_file = server + '-config-filter-' + \
                        (module.partition('<')[2]).partition(' xmlns')[0] + '-' + \
                        str(datetime.now().strftime('%Y%m%d')) + '.xml'
                with open(filter_file, 'w') as f:
                    f.write('<filter>\n')
                    f.write(module)
                    f.write('</filter>')

                config_module_file = server + '-config-module-' + \
                        (module.partition('<')[2]).partition(' xmlns')[0] + '-' + \
                        str(datetime.now().strftime('%Y%m%d')) + '.xml'
                config_list.write(config_module_file + '\n')
                with open(config_module_file, 'w') as f:
#                    f.write('<config>\n')
                    c = m.get_config(source='running', \
                        filter=open(filter_file, 'r').read()).data_xml
                    c = re.sub(r'<\?xml.*>', '<config>', c)
                    c = re.sub(' </data>', '</config', c)
                    f.write(c)
#                    f.write('</config>')
                    module = modules.readline()
```

For each native model

Construct a filter file

Read, file config

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Agenda

- ✓YANG Models, Config & Oper data
- ✓NETCONF Protocol & Tools
- ✓Zero-Touch Provisioning
- ✓Initial “golden” config
- ➢Deployment
- Functional verification

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Deployment

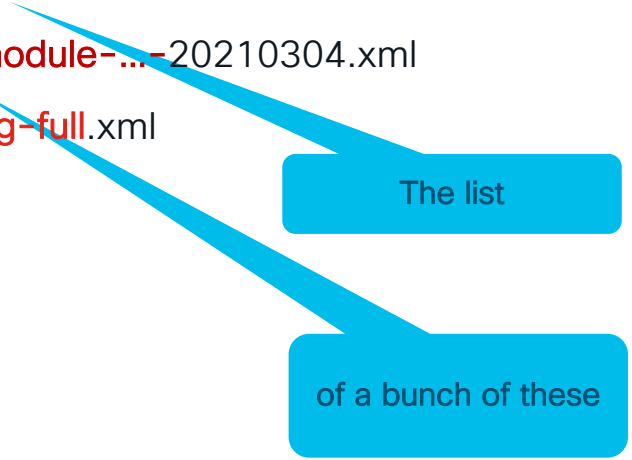
https://github.com/mikemikhail/NETCONF-ZTP-config-retrieval-and-push

Push Config, Log Return

Using NETCONF XML RPCs

edit-config-modules.py:

- Loop through list of config models <= 172.16.7.112-config-module-list-20210304.txt
- Push+commit each config module <= 172.16.7.112-config-module-...-20210304.xml
- Log return error/<ok/> => 172.16.7.112-response-edit-config-full.xml



https://github.com/mikemikhail/NETCONF-ZTP-config-retrieval-and-push

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

RPC Response

Push Configs, Return Status

```
cisco@ubuntu-89:/opt/live/ztp$ more 172.16.7.112-response-edit-config-full.xml
172.16.7.112-config-module-host-names-20210304.xml<?xml version="1.0"?>
<rpc-reply message-id="urn:uuid:c1ab88ce-ace9-4876-9877-b5c085ad2fc0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:
ietf:params:xml:ns:netconf:base:1.0">
 <ok/>
</rpc-reply>
<?xml version="1.0"?>
<rpc-reply message-id="urn:uuid:9cba9235-6189-4f81-b012-6218d9b04463"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:
ietf:params:xml:ns:netconf:base:1.0">
 <ok/>
</rpc-reply>
172.16.7.112-config-module-mpls-oam-20210304.xml<?xml version="1.0"?>
<rpc-reply message-id="urn:uuid:efef7e0a-0ca5-4fd0-9d0c-b9b094447765"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:
ietf:params:xml:ns:netconf:base:1.0">
 <ok/>
</rpc-reply>
<?xml version="1.0"?>
<rpc-reply message-id="urn:uuid:2c7a3ec4-aab9-4201-9c61-f8845d737059"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:
ietf:params:xml:ns:netconf:base:1.0">
 <ok/>
</rpc-reply>
.
```

edit-config

commit

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

The Code

edit-config-modules.py: Loop Throug, Push

```
#! /usr/bin/env python2.7
.
response = '172.16.7.112-response-edit-config-full.xml'

def demo(host, user):
    with manager.connect(host=host, port=830, username='cisco', password='cisco', \
                hostkey_verify=False, timeout=120) as m, \
         open('172.16.7.112-config-module-list-20210304.txt', 'r') as config_list:
         config_module = config_list.readline().split('\n')[0]

         while config_module:
             print config_module
             c = m.edit_config(open(config_module, 'r').read(), format='xml', \
                target='candidate', default_operation='merge')
             print c
             with open(response, 'w') as f:
                 f.write(str(c))
             c = m.commit()
             print c
             with open(response, 'a') as f:
                 f.write(str(c))
             config_module = config_list.readline().split('\n')[0]

if __name__ == '__main__':
    demo(sys.argv[1], os.getenv("USER"))
```

For every native model

Find its config file

edit-config

commit

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Agenda

- ✓YANG Models, Config & Oper data
- ✓NETCONF Protocol & Tools
- ✓Zero-Touch Provisioning
- ✓Initial “golden” config
- ✓Deployment
- ➢Functional verification

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Functional verification

Functional Indicators & Success Criteria

Verify: is deployed node functional?

1. Read operational functional indicators
   - GET number/state/specifics of: IGP – BGP – RIB – uplinks – prefixes
2. What to do if failed
   - If not success: retry/reload/reconfigure/redeploy? – optional edit-config? – notify?

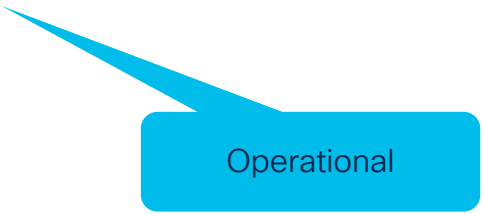
© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

The Data

RPC Filter to <get> Operational Parameters

Read ISIS IGP adjacencies

```
cisco@mamikhai-ubuntu:/opt/ncclient/live$ more filter-oper-router-isis-neighbors.xml
<isis xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-clns-isis-oper">
 <instances>
   <instance>
     <instance-name>ISIS</instance-name>
     <neighbors>
       <neighbor>
         <neighbor-state>isis-adj-up-state</neighbor-state>
       </neighbor>
     </neighbors>
   </instance>
 </instances>
</isis>
```



© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

RPC Filter to <get> Operational Parameters

... and global IPv4 routing table

```
cisco@mamikhai-ubuntu:/opt/ncclient/live$ more filter-oper-rib-ipv4-default.xml
<rib xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ip-rib-ipv4-oper">
 <vrfs>
   <vrf>
     <vrf-name>default</vrf-name>
     <afs>
       <af>
         <af-name>IPv4</af-name>
         <safs>
           <saf>
             <saf-name>Unicast</saf-name>
             <ip-rib-route-table-names>
               <ip-rib-route-table-name>default</ip-rib-route-table-name>
             </ip-rib-route-table-names>
           </saf>
         </safs>
       </af>
     </afs>
   </vrf>
 </vrfs>
</rib>
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

RPC Content to <edit-config>

Add an interface

```
cisco@mamikhai-ubuntu:/opt/ncclient/live$ more edit-config-add-loopback55.xml
<config>
 <interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
   <interface-configuration>
     <active>act</active>
     <interface-name>Loopback55</interface-name>
     <interface-virtual/>
     <description>ENABLED ONLY IF NEIGHBOR ADJACENCY FAILS</description>
     <ipv4-network xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-io-cfg">
       <addresses>
         <primary>
           <address>172.16.255.55</address>
           <netmask>255.255.255.255</netmask>
         </primary>
       </addresses>
     </ipv4-network>
   </interface-configuration>
 </interface-configurations>
</config>
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

RPC Content to <edit-config>

... or delete it!

```
cisco@mamikhai-ubuntu:/opt/ncclient/live$ more edit-config-delete-loopback55.xml
<config>
 <interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
   <interface-configuration nc:operation="delete">
     <active>act</active>
     <interface-name>Loopback55</interface-name>
   </interface-configuration>
 </interface-configurations>
</config>
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Sample Code

Sample Script : Read, Evaluate, and Change

Using NETCONF XML RPCs

A Python routine, using ncclient, and customized XML calls:

- Read operational data: IGP adjacencies; and global routing table
- Check conditions: Number of adjacencies; & a specific prefix in RIB
- If conditions are met: edit configuration; or no action
- Logs in both brief and detailed traces
- Repeat every n seconds

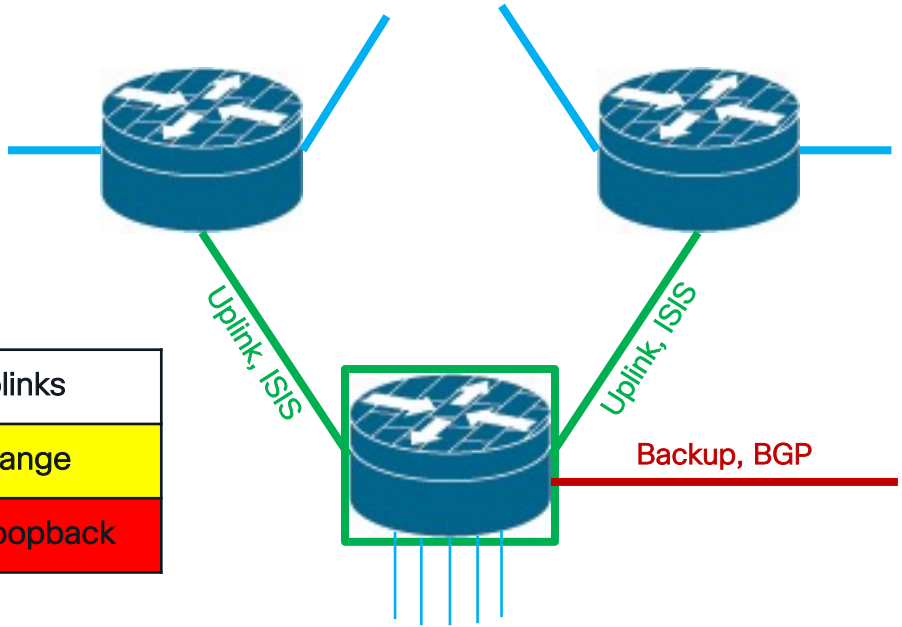
https://github.com/mikemikhail/if_modify

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Topology and Conditions

https://github.com/mikemikhail/if_modify

Conditions	>= 2 uplinks	<2 uplinks
Loopback up	delete loopback	no change
Loopback down	no change	add loopback



© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Script Part A

Imports, RPC XML content, and conditions

```
#! /usr/bin/env python
# Sample if_modify routine. This uses
ncclient to send NETCONF XML RPC's
# Reads ISIS adjacency status and an IPv4
prefix every cycle from XR
# If conditions are met, a loopback is
added/deleted or no change
# Two logs appended, one detailed, the
other is a one liner per interval
# mamikhai@cisco.com

import sys, os, warnings
warnings.simplefilter("ignore",
DeprecationWarning)
from ncclient import manager
import time
from datetime import datetime
```

```
# target NETCONF server
server = '10.101.112.1'
# time between checks in seconds
t = 300.000
# filters and responses
filter1 = 'filter-oper-router-isis-
neighbors.xml'
filter2 = 'filter-oper-rib-ipv4-default.xml'
response1 = 'isis.xml'
response2 = 'rib.xml'
action1 = 'edit-config-delete-loopback55.xml'
action2 = 'edit-config-add-loopback55.xml'

# minimum IGP adjacencies for no need for
loopback
min_adj = 2

logfile = 'if_modify.log'
tracefile = 'if_modify.trace.log'

user ='cisco'
password ='cisco'
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Script Part B

Function to: Read, check info, log

Response filename

To be logged

To be counted

Filter filename

```
def check_oper(subtree_filter, response, trace_string, count_string):
    count = 0
    trace = open(tracefile,'a')
    op_time = str(datetime.now())
    trace.write(op_time + ' ' + str(trace_string) + ' ' + str(count_string) + '\n')

    # Get data, record
    c = m.get(filter = ('subtree', open(subtree_filter, 'r').read()))
    with open(response, 'w') as f:
        f.write(str(c))

    # Check for target data, record, count
    with open(response, 'r') as f:
        for line in f:
            if trace_string and (trace_string in line):
                trace.write(line)
            if count_string in line:
                count +=1
    trace.close()

    return count
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Script Part C

Call read, act? ...

```
if __name__ == '__main__':
    # Log start time
    trace = open(tracefile,'a')
    log = open(logfile,'a')
    op_time = str(datetime.now())
    trace.write(op_time + ' start' + '\n')
    log.write(op_time + ' start' + '\n')
    trace.close()
    log.close()

    with manager.connect(host=server, port=830, username=user, password=password) as m:
        # Endless cycle
        while True:
            log = open(logfile, 'a')
            op_time = str(datetime.now())

            # Read ISIS adjacencies
            adj = check_oper(filter1, response1, '<system-id>',
            '<neighbor-state>isis-adj-up-state</neighbor-state>')

            log.write(op_time + ' adjacencies: ' + str(adj))
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Script Part C – Continued

Call read, act? …

```
# Check for presence of a specfic interface in IPv4 global RIB
route = check_oper(filter2, response2, None, \
    '<interface-name>Loopback55</interface-name>')

# If adjacencies back to normal and loopback is in RIB, delete loopback55
if adj>= min_adj:
    if route:
        m.edit_config(open(action1, 'r').read(), format='xml', \
            target='candidate', default_operation='merge')
        m.commit()
        log.write('; loopback55 deleted')
# If adjacencies drop and loopback is not in RIB, configure loopback55
elif not route:
    m.edit_config(open(action2, 'r').read(), format='xml', \
            target='candidate', default_operation='merge')
    m.commit()
    log.write('; loopback55 configured')
log.write('\n')
log.close()

# Wait for next cycle
time.sleep(t)
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

and Logs

A summary log, and a detailed trace

```
cisco@mamikhai-ubuntu:/opt/ncclient/live$ tail -n 11 if_modify.log
2021-01-24 19:36:06.168655 start
2021-01-24 19:36:07.969928 adjacencies: 2
2021-01-24 19:41:09.386943 adjacencies: 2
2021-01-24 19:46:10.112325 adjacencies: 2
2021-01-24 19:51:11.302043 adjacencies: 2
2021-01-24 19:56:13.023289 adjacencies: 2
2021-01-24 20:01:14.860292 adjacencies: 1; loopback55 configured
2021-01-24 20:06:18.217079 adjacencies: 1
2021-01-24 20:11:19.275329 adjacencies: 1
2021-01-24 20:16:20.168785 adjacencies: 2; loopback55 deleted
2021-01-24 20:21:22.977216 adjacencies: 2
```

```
cisco@mamikhai-ubuntu:/opt/ncclient/live$ tail -n 20 if_modify.trace.log
.
2021-01-24 20:11:19.275401<system-id><neighbor-state>isis-adj-up-state</neighbor-state>
      <system-id>0101.0010.1001</system-id>
2021-01-24 20:11:19.396763 None<interface-name>Loopback55</interface-name>
2021-01-24 20:16:20.168861<system-id><neighbor-state>isis-adj-up-state</neighbor-state>
      <system-id>0101.0010.1001</system-id>
      <system-id>0101.0010.2001</system-id>
2021-01-24 20:16:20.465473 None<interface-name>Loopback55</interface-name>
.
```


© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

We have covered...

- ✓YANG Models, Config & Oper data
- ✓NETCONF Protocol & Tools
- ✓Zero-Touch Provisioning
- ✓Initial “golden” config
- ✓Deployment
- ✓Functional verification

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

Continue your education

- Demos in the Cisco campus
- Meet the engineer 1:1 meetings
- Walk-in labs
- Related sessions

© 2021 Cisco and/or its affiliates. All rights reserved. Cisco Public

The bridge to possible

Thank you


cisco Live!
TURN
IT
UP
#CiscoLive