YANG modely v kontexte protokolov RESTCONF a NETCONF

Práca s YANG modelmi v súvislosti s RESTCONF a NETCONF protokolmi

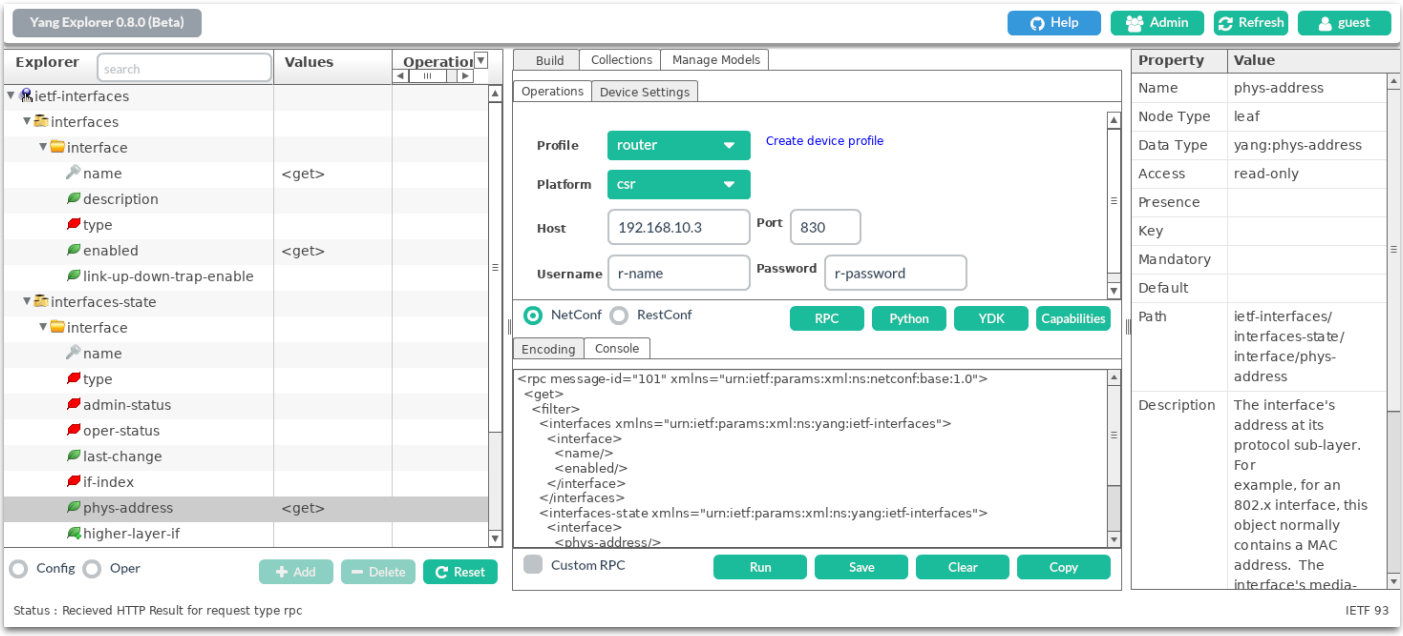
YANG modely slúžia na opis vlastností objektov. Z pohľadu počítačových sietí je možné YANG modely využiť na opis rozhraní, bežiacej konfigurácie, smerovacích protokolov a podobne. Je však potrebné podotknúť, že YANG modely definujú dátovú štruktúru príslušných objektov, pričom každý objekt (rozhranie, smerovací protokol, prístupový zoznam, ...) je opísaný príslušným YANG modelom.

YANG modely

YANG modely môžu mať rôzny pôvod. Existujú štandardizované YANG modely vytvorené štandardizačnými organizáciami ako IETF, IEEE a pod. Keďže každý výrobca si môže funkčnosť svojich riešení ľubovoľne rozširovať, môže sa stať, že štandardné YANG modely by na opis príslušného objektu neboli postačujúce, resp. by neposkytovali úplný popis. Z tohto dôvodu má každý výrobca možnosť definovať si vlastné proprietárne YANG modely (Cisco, Juniper a pod.).

YANG modely je možné nájsť na: https://github.com/YangModels/yang, kde sa nachádzajú štandardizované aj výrobcami definované modely. Keďže sa bude v našom prípade pracovať s Cisco IOS XE, tak budeme používať práve modely pre tento typ operačného systému, ktoré je možné nájsť na: https://github.com/YangModels/yang/tree/master/vendor/cisco/xe/1661.

YANG dátové modely sú napísané využitím YANG jazyka, ktorý žiaľ nie je pre bežného človeka veľmi čitateľný. Pre zjednodušenie práce s YANG modelmi preto vznikli rôzne nástroje, ktoré umožňujú prepis zložitého YANG zápisu do, pre človeka, prijateľnejšej podoby. Jedným z nástrojov je YANG-Explorer, zobrazený na obrázku 1.1:



Obrázok 1.1 Rozhranie nástroja Yang Explorer

Nástroj je však možné využívať len na OS Linux a preto sa ním nebudeme v ďalších častiach zaoberať.

Pre neskoršie využitie budeme potrebovať nasledujúce informácie:

- **Module**: ietf-interfaces
- **Namespace**: urn:ietf:params:xml:ns:yang:ietf-interfaces
- **Container**: interfaces

Práve tieto údaje neskôr využijeme na vytváranie RESTCONF URI a na vytváranie NETCONF filtrov:

```
module ietf-interfaces {

   namespace "urn:ietf:params:xml:ns:yang:ietf-interfaces";
   prefix if;

   import ietf-yang-types {
       prefix yang;
   }
   organization
      "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
```

```
container interfaces {
    description
      "Interface configuration parameters.";

    list interface {
      key "name";

      description
        "The list of configured interfaces on the device.
         The operational state of an interface is available in the
         /interfaces-state/interface list.  If the configuration of a
         system-controlled interface cannot be used by the system
         (e.g., the interface hardware present does not match the
         interface type), then the configuration is not applied to
         the system-controlled interface shown in the
         /interfaces-state/interface list.  If the configuration
         of a user-controlled interface cannot be used by the system,
         the configured interface is not instantiated in the
         /interfaces-state/interface list.";

     leaf name {
        type string;
        description
          "The name of the interface.
```

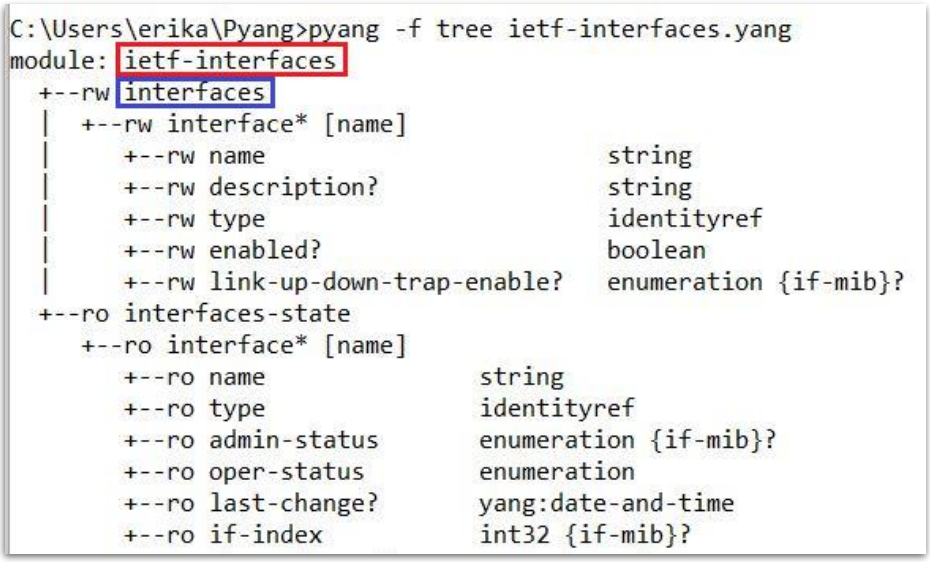
Keďže je na prvý pohľad vidieť, že YANG jazyk nie je veľmi čitateľný a zároveň sú modely opísané YANG jazykom aj značne rozsiahle – napr. samotný model pre rozhranie má 725 riadkov, tak z tohto dôvodu budeme pre zjednodušenie práce, pochopenie a vizualizáciu týchto modulov používať nástroj **PYANG.** Tento nástroj umožňuje zobraziť YANG model v stromovej štruktúre, pričom uvádza aj charakteristiku jednotlivých atribútov (dátový typ a možnosť prístupu read/write). Nástroj PYANG stačí nainštalovať využitím príkazového riadka príkazom:

```
$ pip install --no-binary pyang pyang
```

Následne na zobrazenie konkrétneho YANG modelu stačí v CMD zadať: `pyang -f tree <nazov.yang>` napr.:

```
$ pyang -f tree ietf-interfaces.yang
```

Výpis nástroja PYANG je znázornený na obrázku 1.2:

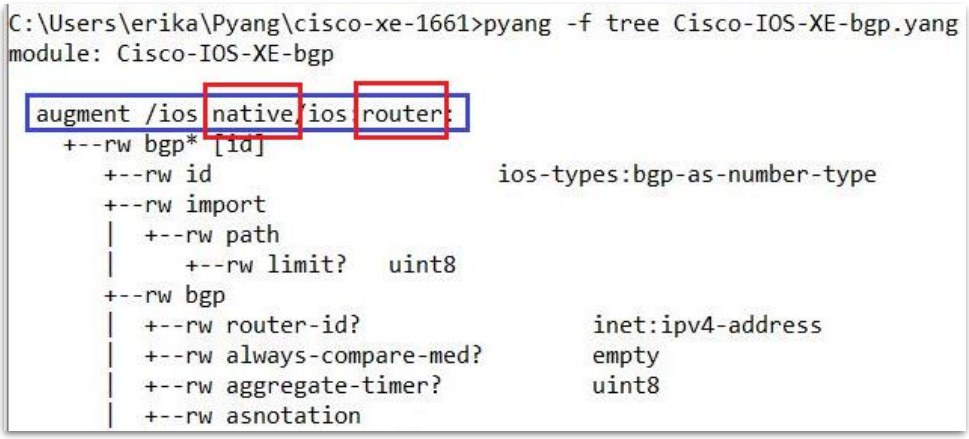


Obrázok 1.2 Výpis z nástroja PYANG pre modul interfaces

Červenou farbou je znázornený názov modulu a modrou je znázornený názov kontajnera. V tomto výpise nám chýba použitý *namespace*, ktorý bude potrebný pri vytváraní NETCONF filtrov. Využitím pôvodného YANG modelu, bude možné potrebné chýbajúce údaje dodatočne vyhľadať.

K zobrazenému YANG modelu je možné pristupovať priamo využitím názvu kontajnera. Nie všetky YANG modely sú ale takto priamočiaro spísané. Niektoré modely využívajú rozšírenia (*augment*), ktorými sa odkazujú na iné YANG modely.

Ukážka takéhoto YANG modelu je znázornená na obrázku 1.3:



*Obrázok 1.3 Výpis z nástroja PYANG, využívajúci rozšírenie do kontajnera **router***

Ako je možné vidieť, modul pre protokol BGP neobsahuje koreňový kontajner, ale hneď sa odkazuje na rozšírenie, konkrétne do modulu ***native*** a kontajneru ***router***. Niektoré YANG modely sú opísané len týmto štýlom. Čo to pre nás znamená? Znamená to, že informácie o protokole BGP nie je možné nájsť využitím tohto dedikovaného modelu, no je ich možné nájsť v iných modeloch, s ktorými vieme pracovať priamočiaro, ako bolo možné vidieť v prípade YANG modelu pre rozhrania.

Záver: Cez nástroj PYANG máme možnosť pozrieť na stromovú štruktúru dát. V prípade, že model začína kontajnerom, vieme dáta získavať priamo využitím daného modelu. V prípade ak je v modeli využité rozšírenie, tak je potrebné na získanie daných dát využiť iné modely, v ktorých vieme dané dáta získať priamo cez kontajner.

Zatiaľ sme si vysvetlili čo YANG model je a ako sa s ním pracuje. Nasledujúce dve časti sa budú venovať využitiu YANG dát v súvislosti s protokolmi RESTCONF a NETCONF.

RESTCONF a práca s YANG objektmi

Protokol RESTCONF sa používa na prístup k zariadeniu využitím REST API. Nevyhnutnosťou je správne vytvorenie URI identifikátoru, na základe YANG modelu. Keďže protokolom RESTCONF prenášame citlivé konfiguračné dáta, tak je samotný transport realizovaný využitím HTTPS protokolu.

URI má štandardne nasledujúci zápis:

https://<ip_adresa>/restconf/data/modul:kontajner/<špecifický_atribút_sekcie>

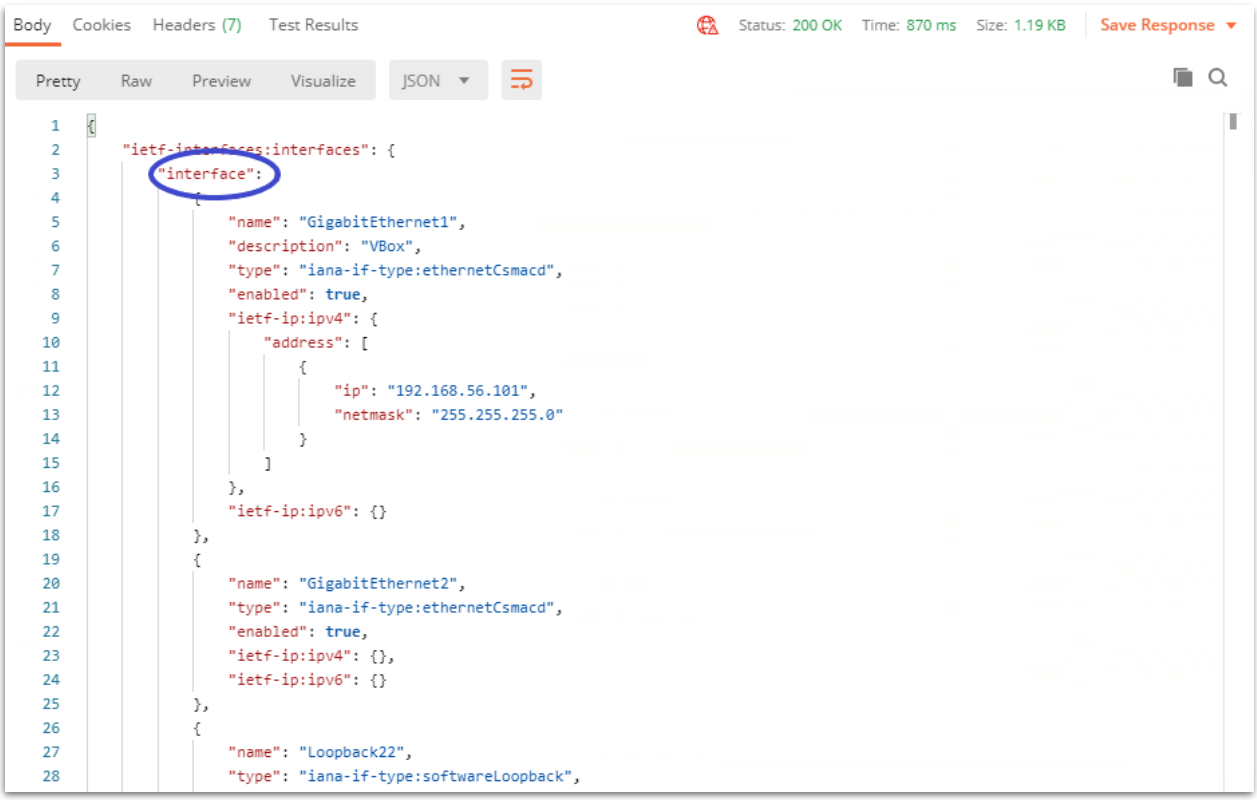
Na základe HTTP metódy (GET, PUT, DELETE) vieme vykonať príslušnú akciu na daných dátach (čítať, modifikovať, mazať). Čo konkrétne je možné robiť, zistíme využitím nástroja PYANG, ktorý pri každom atribúte uvádza, či je možné daný atribút len čítať (RO) alebo aj meniť (RW). Formát prenášaných dát je JSON alebo XML, odporúčané je ale využívať JSON formátovanie. V prípade potreby je možné využívať aj filtrovanie údajov spôsobom, že za kontajnerom v URI za znakom „**/**“ uvádzame názvy „podkontajnerov“ a teda zobrazený výsledok je kratší a jednoduchší na spracovanie.

Odporúčanie: Ak chceme nejaký atribút meniť, je vhodné najskôr zistiť jeho formát cez GET metódu. Prijaté dáta je možné následne využiť v metóde PUT, kde do časti *Body* vložíme JSON v správnom formáte. Presne kvôli tomu sa odporúča vykonať najskôr GET, ktorý nám ukáže správne formátovanie. Z pohľadu administrátora už následne stačí len zmeniť požadované hodnoty.

Príklady: Nasleduje niekoľko príkladov, ktoré sa budú venovať ukážkam tvorby URI pre získanie požadovaných objektov zo zariadenia v YANG formáte a ich filtrovanie.

Na získanie údajov o rozhraní je potrebné vyhľadať modul, ktorý bude tieto údaje poskytovať (napr. *ietf-interfaces*) a určiť názov kontajnera (*interfaces*). Oba tieto údaje je možné vyčítať z PYANG nástroja. Výsledná URI je teda:

https://192.168.2.6/restconf/data/ietf-interfaces:interfaces/

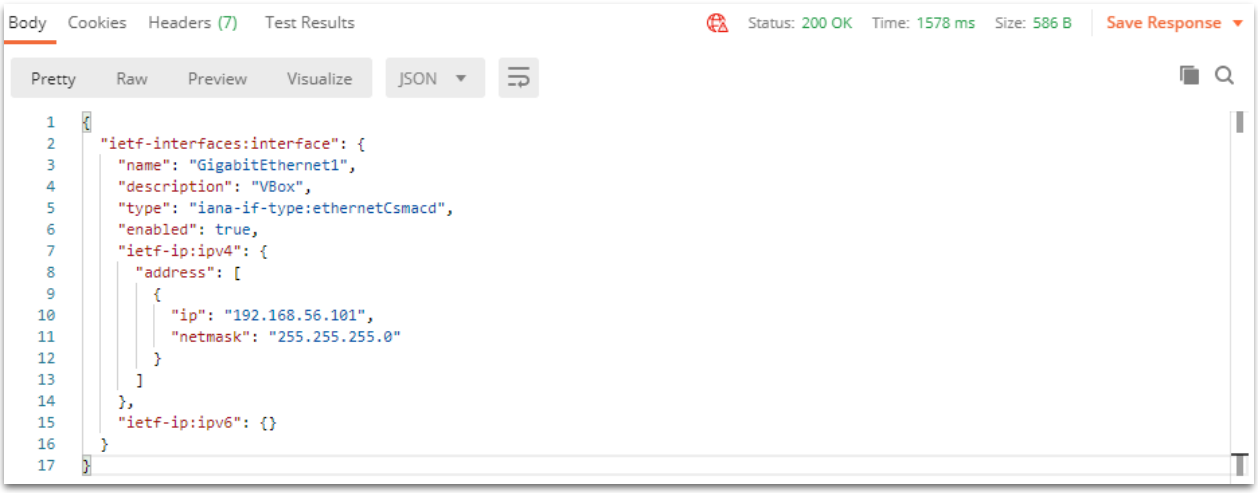


Obrázok 1.4 Prostredie Postman – odpoveď GET požiadavky pre všetky rozhrania, v JSON formáte

Ako je možné vidieť na obrázku 1.4, operácia GET nám vrátila údaje o všetkých rozhraniach, zapísané v JSON formáte, ktorých štruktúra je definovaná YANG modelom. Môžeme si všimnúť, že kontajner pozostáva zo zoznamu rozhraní, pričom tento zoznam je označený názvom „interface“. Výstup PYANG nástroja nám ukazuje, že ku konkrétnemu prvku zoznamu vieme pristupovať cez kľúč, ktorý je totožný s názvom rozhrania. Teda, ak nás zaujímajú informácie o konkrétnom rozhraní GigabitEthernet1, tak stačí obohatiť URI o „/interface=GigabitEthernet1“.

Nasledujúci výpis znázorňuje výsledné URI a obrázok 1.5 očakávanú odpoveď:

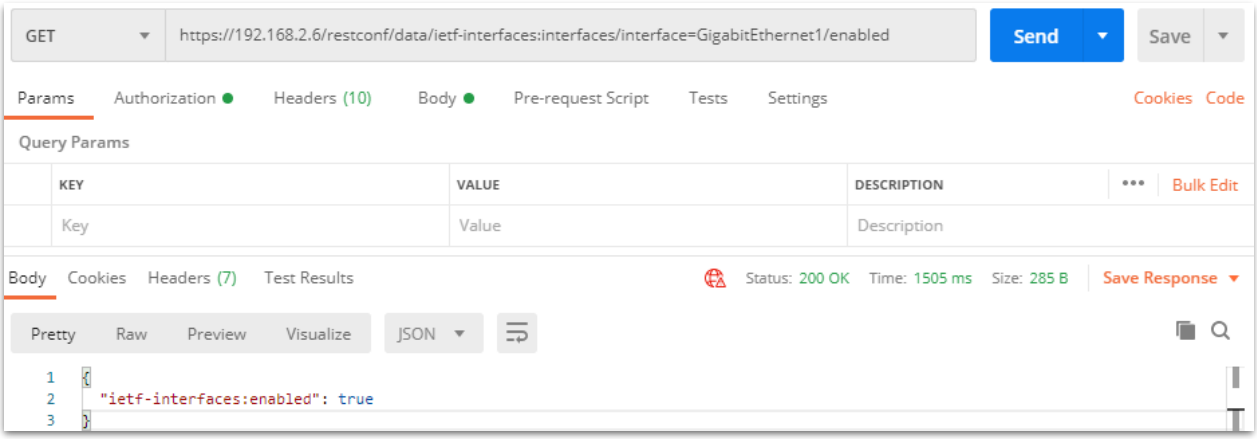
```
https://192.168.2.6/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet1
```



Obrázok 1.5 Prostredie Postman – odpoveď GET požiadavky pre konkrétne rozhranie GigabitEthernet1

Podobne môžeme pokračovať ďalej. Ak potrebujeme ešte konkrétnejší údaj, stačí cez lomku vstúpiť do nižšej úrovne. Ak by sme chceli filtrovať len stav rozhrania, tak do URI stačí doplniť „/enabled“, čo znázorňuje nasledujúci výpis a obrázok 1.6:

```
https://192.168.2.6/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet1/enabled
```



Obrázok 1.6 Prostredie Postman – odpoveď GET požiadavky pre stav konkrétneho rozhrania GigabitEthernet1

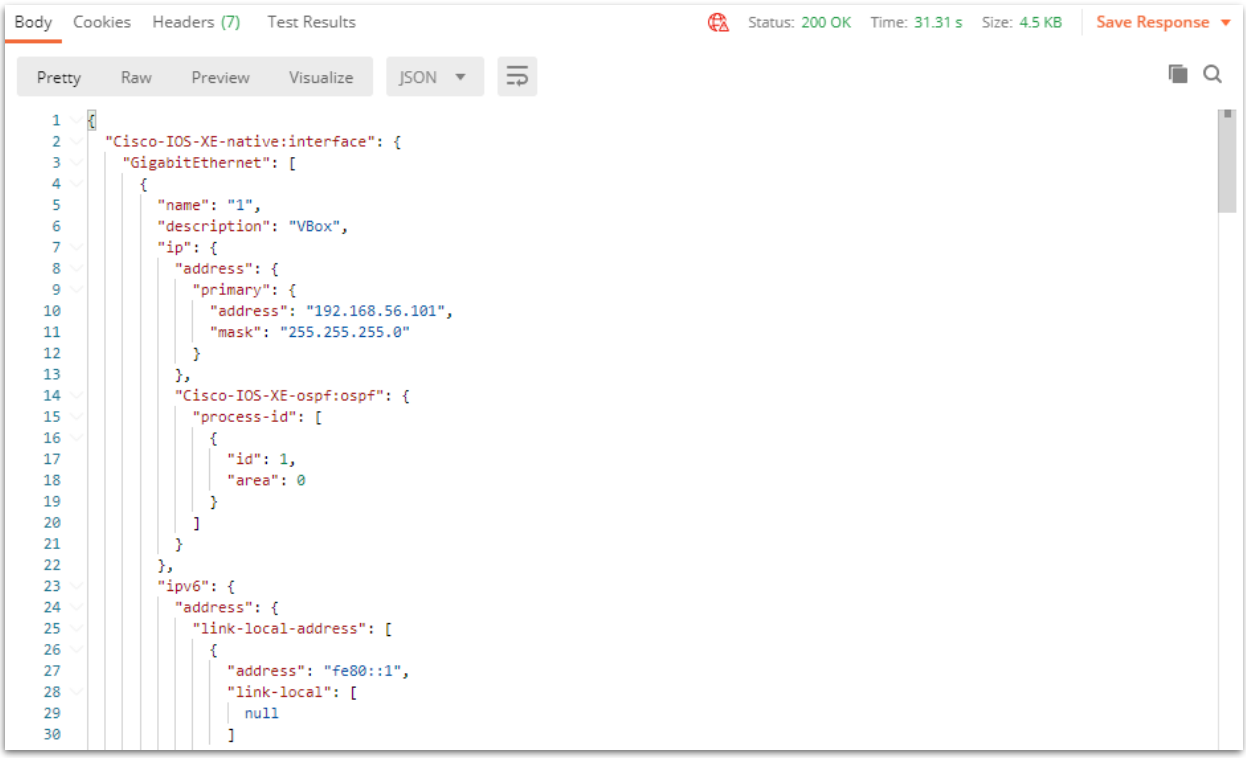
Výsledkom tohto filtra je vypísanie reprezentatívnej hodnoty stavu rozhrania. Takéto postupné vnáranie je možné vďaka vnáraným JSON objektom.

Teraz skúsme vyčítať informácie o bežiacej konfigurácii. Najskôr lokalizujme v akom module by sa mohla nachádzať. Jedná sa o modul *Cisco-IOS-XE-native*. Ak chceme ale vytvoriť URI pre RESTCONF, tak je potrebné zistiť aj názov kontajnera. Ten vyčítame cez nástroj PYANG, z ktorého získame odpoveď *native*.

URI pre získanie bežiacej konfigurácie je teda:

https://192.168.2.6/restconf/data/Cisco-IOS-XE-native:native

Bežiaca konfigurácia predstavuje rozsiahlu stromovú štruktúru, v ktorej je uložená celá konfigurácia. Skúsme zobraziť informácie o rozhraniach. Pre tento účel sa stačí pozrieť do výstupu nástroja PYANG, alebo do prijatej JSON štruktúry, kde je možné vidieť, že rozhrania sa nachádzajú v zozname s názvom interface. Do URI teda pridáme „/interface“. Výsledok zobrazuje obrázok 1.7:



Obrázok 1.7 Prostredie Postman – odpoveď GET požiadavky pre získanie všetkých rozhraní z bežiacej konfigurácie

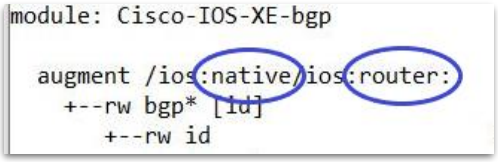
Podobne, ak by sme chceli zobraziť informáciu len o rozhraní GigabitEthernet, tak by URI vyzerala:

https://192.168.2.6/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet/

Tu nastáva problém, ako sa dostať ku konkrétnej hodnote IP adresy, ku ktorej sa potrebujeme dostať identifikovaním položky v poli cez index - nie cez kľúč. Postman takýto filter neumožňuje, no v prípade práce v programovacom jazyku sa dá k hodnotám dopracovať cez:

[0] [„ip“] [„address“] [„primary“] [„address“]

Teraz si ukážme prácu s YANG modelom, ktorý je definovaný cez rozšírenia. Takýmto modelom je napr. model pre protokol BGP. Najskôr vyhľadáme daný model v nástroji PYANG, ktorý nám vypíše takúto hodnotu zobrazenú na obrázku 1.8 (výpis je rozsiahly, pre nás je ale zaujímavý začiatok výpisu):

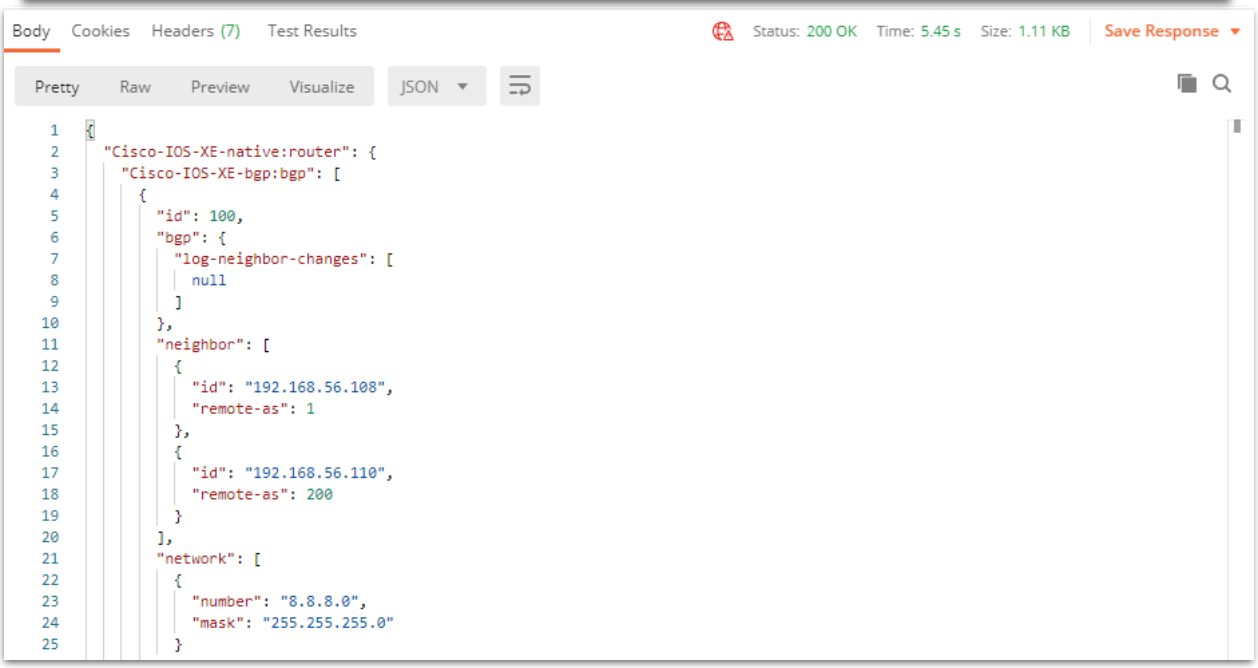


Obrázok 1.8 YANG modul definovaný rozšírením cez iný YANG modul

Na obrázku 1.8 vidíme, že modul nezačína žiadnym kontajnerom ale priamo sa odkazuje na iný YANG modul. Ak sa teda chceme dostať k požadovaným údajom, tak potrebujeme vytvoriť URI pre modul *native*, čo sme robili v predchádzajúcej časti. Následne cez „/“ uviesť sekciu „*router*“.

Výsledná URI vyzerá nasledovne, pričom očakávaný výstup zobrazuje obrázok 1.9:

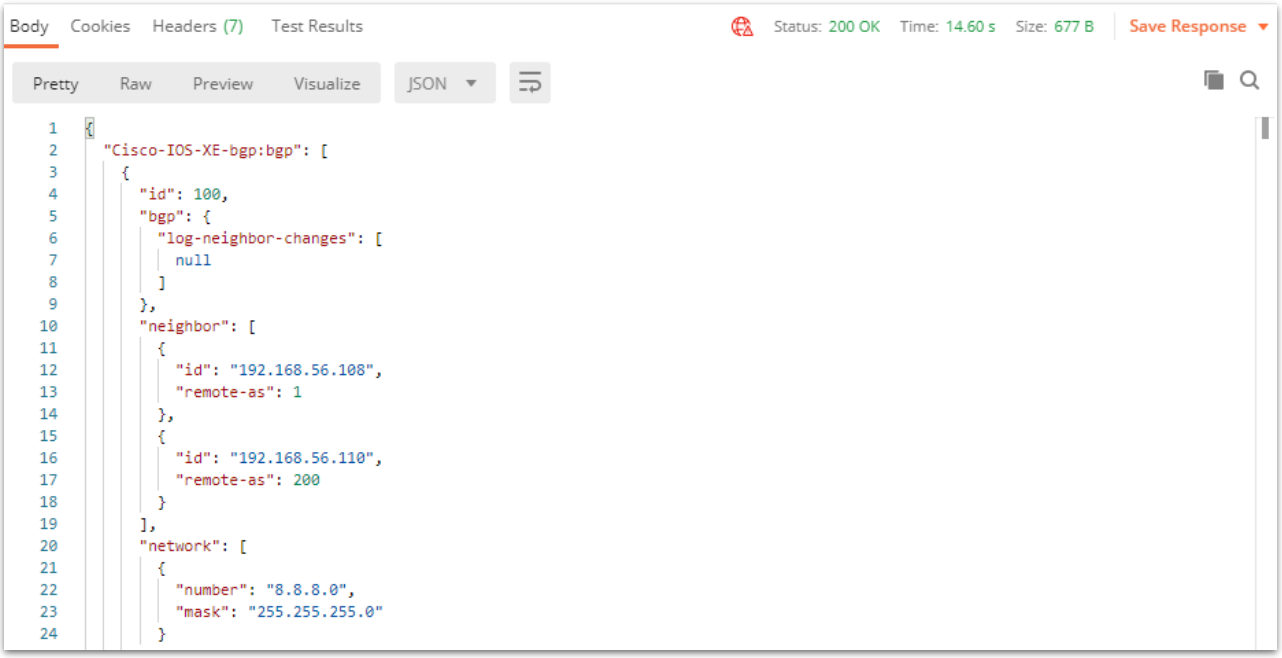
https://192.168.2.6/restconf/data/Cisco-IOS-XE-native:native/router



Obrázok 1.9 ***Prostredie Postman – odpoveď GET požiadavky pre modul router***

Rovnako ako v predchádzajúcich častiach sa vieme dostať ku konkrétnym údajom využitím filtrovania cez „**/**“ a využitím kľúčových slov, čo znázorňuje nasledujúci výpis a obrázok 1.10:

https://192.168.2.6/restconf/data/Cisco-IOS-XE-native:native/router/Cisco-IOS-XE-bgp:bgp



Obrázok 1.10 ***Prostredie Postman – odpoveď GET požiadavky pre protokol BGP***

Niekoľko ďalších vzorových RESTCONF URI:

https://192.168.2.5:443/restconf/data/ietf-routing:routing-state

https://192.168.2.5:443/restconf/data/Cisco-IOS-XE-acl-oper:access-lists

https://192.168.2.5:443/restconf/data/ietf-interfaces:interfaces-state

https://192.168.2.5:443/restconf/data/Cisco-IOS-XE-bgp-oper:bgp-state-data

Odporúčania na záver:

1. Využitím nástroja PYANG sa dá dostať k názvu modulu a kontajnera.
2. Ak YANG model neobsahuje koreňový kontajner ale rozšírenia, potom je potrebné podľa definície rozšírenia *„augment“* identifikovať YANG modul, cez ktorý je možné nájsť názov kontajnera.
3. Ak je známy modul a kontajner, potom je možné využitím filtrov pracovať s údajmi, ktoré sú pre nás zaujímavé.
4. Názov modulu a kontajnera je dôležitý pre vytvorenie URI. Následne je túto URI možné využiť v aplikácii Postman alebo akomkoľvek programovacom jazyku.

Upozornenie: Postman je vhodný len na základné testovanie a neumožňuje automatizáciu. Ako vhodný programovací jazyk sa pre tento účel dá považovať jazyk Python. Ukážka jednoduchého kódu vyčítania názvu zariadenia je znázornená nižšie:

```
#sekcia, ktorá vytvorí http správu na komunikáciu s RESTCONF API
url = “https://192.168.56.101:443/restconf/data/Cisco-IOS-XE-native:native“
payload = {}
headers = {
   ‘Accept‘: ‘application/yang-data+json‘,
   ‘Content-Type‘: ‘application/yang-data+json‘,
   ‘Authorization‘: ‘Basic Y21zY286Y21zY28xMjMh‘
}

#sekcia, ktorá pošle GET žiadosť a prevedie prijaté dáta do dict (JSON object) štruktúry
new_get=requests.request(„GET“, url, headers=headers, data=payload, verify=False)
data=new_get.json()

#sekcia, ktorá odošle názov zariadenia do Webexu
post_to_webex(data[„Cisco-IOS-XE-native:native“][„hostname“])
```

Ukážka kódu pre zmenu názvu zariadenia:

```
url = “https://192.168.56.101:443/restconf/data/native/hostname“
payload = {„Cisco-IOS-XE-native:hostname“: „novy_nazov“}
headers = {
   ‘Accept‘: ‘application/yang-data+json‘,
   ‘Content-Type‘: ‘application/yang-data+json‘,
   ‘Authorization‘: ‘Basic Y21zY286Y21zY28xMjMh‘
}

#odoslanie požiadavaky na smerovač, pre zmenu názvu zariadenia
new_set=requests.request(„PUT“,url,headers=headers, data=json.dumps(payload), verify=False)
```

Ako je možné vidieť, jediné čo je potrebné definovať je URI a hlavička, ktorá obsahuje autentifikačné údaje. V prípade zmeny je to JSON objekt, ktorý obsahuje potrebné informácie. Štruktúru spomínaného JSON objektu je veľmi jednoduché získať cez `GET` žiadosť, v ktorej `payload` nedefinujeme.

Pozn.: Výhodou Postman aplikácie je hlavne jej jednoduchá obsluha a možnosť generovať zdrojový kód pre viacero rôznych programovacích jazykov, čo umožňuje integráciu a jednoduchú tvorbu kontrolérov aj v rôznych jazykoch.

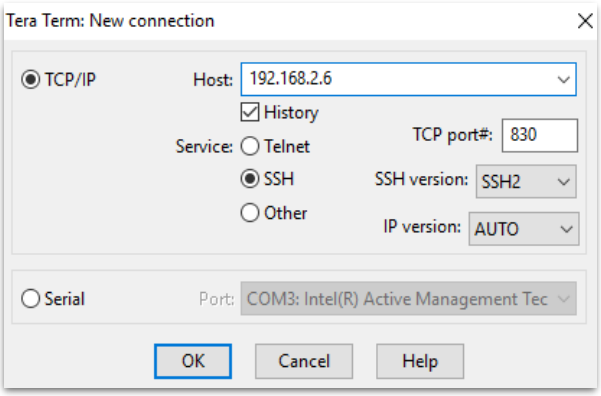
NETCONF a práca s YANG objektmi

Protokol NETCONF má narozdiel od protokolu RESTCONF zložitejšiu a robustnejšiu štruktúru. Tomu nasvedčuje aj to, že protokolom NETCONF je možné vykonávať zložitejšie operácie. Nevýhodou ale je, že jediným možným formátom dát je XML. Toto do istej miery komplikuje prácu s dátami.

Čo potrebujeme k práci s NETCONF protokolom? Tento krát nepotrebujeme vytvárať URI ale odosielať požiadavky (GET, GET-CONFIG, EDIT-CONFIG ...) v XML formáte. Tiež si ukážeme, ako je možné vytvárať filtre, na definovanie len tých dát, ktoré sú pre nás zaujímavé.

Keďže komunikácia NETCONF protokolu potrebuje byť taktiež zabezpečená, využívame šifrovaný dátový prenos využitím SSH protokolu na porte 830.

Skutočnosť, že na prenos dát využívame protokol SSH nám umožní s NETCONF protokolom komunikovať nástrojmi ako Putty či TeraTerm. Mimo toho, je to tiež možné využitím programovacieho jazyka – napr. Python. Pozor však, nástroj Postman v tomto prípade využívať nemôžeme. Obrázok 1.11 znázorňuje údaje potrebné pre pripojenie na NETCONF reláciu cez nástroj TeraTerm:



Obrázok 1.11 Používateľské rozhranie nástroja TeraTerm

Po otvorení spojenia, je potrebné potvrdiť SSH certifikát a následne zadať prihlasovacie meno a heslo pre prístup do zariadenia (autentifikačné údaje na prístup do ssh = username meno password heslo).Po úspešnom autentifikovaní nám NETCONF protokol pošle úvodnú správu, v ktorej nás informuje o jeho schopnostiach (angl. *capabilities*), čo znázorňuje nasledujúci výpis:

```
192.168.2.6 – Tera Term VT
<capability>urn:ietf:params:xml:ns:yang:smiv2:SNMPv2-MIB?module=SNMPv2-MIB&amp;revision=2002-10-16<
/capability>
<capability>urn:ietf:params:xml:ns:yang:smiv2:SNMPv2-TC?module=SNMPv2-TC</capability>
<capability>urn:ietf:params:xml:ns:yang:smiv2:SONET-MIB?module=SONET-MIB&amp;revision=2003-08-11</c
apability>
<capability>urn:ietf:params:xml:ns:yang:smiv2:TCP-MIB?module=TCP-MIB&amp;revision=2005-02-18</capab
ility>
<capability>urn:ietf:params:xml:ns:yang:smiv2:TOKEN-RING-RMON-MIB?module=TOKEN-RING-RMON-MIB</capab
ility>
<capability>urn:ietf:params:xml:ns:yang:smiv2:TOKENRING-MIB?module=TOKENRING-MIB&amp;revision=1994-
10-23</capability>
```

```
<capability>urn:ietf:params:xml:ns:yang:smiv2:TUNNEL-MIB?module=TUNEL-MIB&amp;revision=2005-05-16</
capability>
<capability>urn:ietf:params:xml:ns:yang:smiv2:UDP-MIB?module=UDP-MIB&amp;revision=2005-05-20</capab
ility>
<capability>urn:ietf:params:xml:ns:yang:smiv2:VPN-TC-STD-MIB?module=VPN-TC-STD-MIB&amp;revision=200
5-11-15</capability>
<capability>urn:ietf:params:xml:ns:netconf:base:1.0?module=ietf-netconf&amp;revision=2011-06-01</ca
pability>
<capability>urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults?module=ietf-netconf-with-default
s&amp;revision=2011-06-01</capability>
</capabilities>
<session-id>72004</session-id></hello>]]>]]>
```

Od užívateľa sa v tejto chvíli očakáva, že odpovie tzv. „Hello“ správou, v ktorej odošle svoje schopnosti. Hello správa zadaná do terminálu vyzerá nasledovne:

192.168.2.6 – **Tera Term VT**

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <capabilities>
      <capability>urn:ietf:params:netconf:base:1.0</capability>
   </capabilities>
</hello>
]]>]]>
```

Ak nedôjde k vypísaniu chybovej hlášky, tak sa úspešne podarilo otvoriť NETCONF spojenie. Na ukončenie spojenia sa používa špeciálna správa, ktorej tvar je nasledovný:

```
<rpc message-id="9999999" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <close-session/>
</rpc>
]]>]]>
```

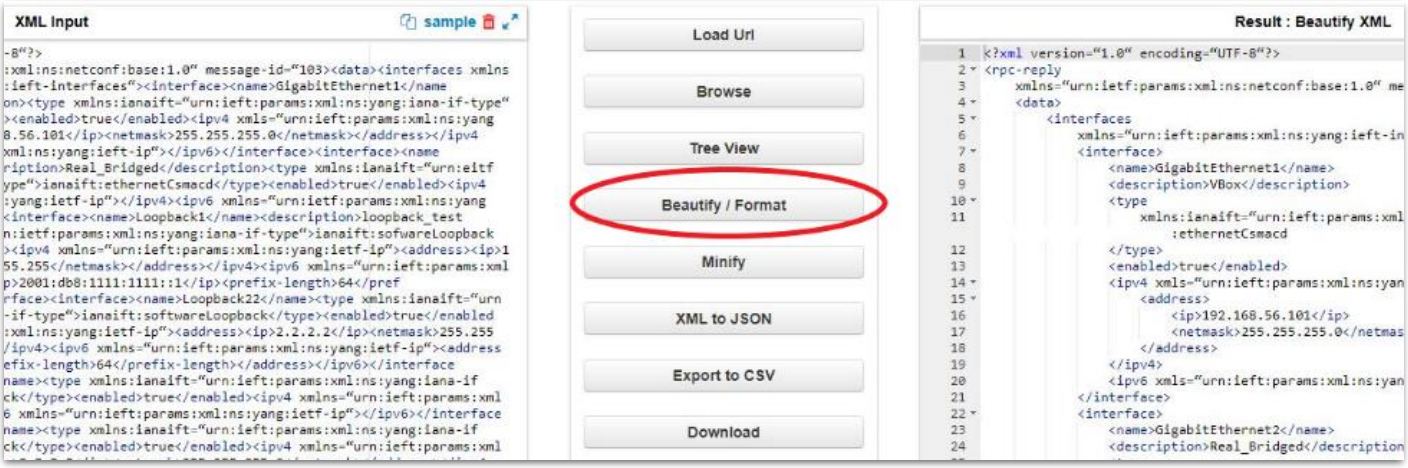
Keďže výpisy NETCONF protokolu sú často rozsiahle, tak je pre nás dôležité vytváranie NETCONF filtrov. Ak by sme chceli vypísať všetko, čo dané zariadenie poskytuje, tak hrubým odhadom sa jedná o dátovú štruktúru o rozsahu 25 000 riadkov XML stromovej štruktúry.

Správa pre získanie všetkých informácií vyzerá nasledovne:

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
   </get>
</rpc>
]]>]]>
```

Ide o vykonanie GET požiadavky bez filtra z base1.0 *capabilities*. S touto správou sa pre jej rozsiahlosť neodporúča pracovať, odporúčané je využívať filtre.

Odporúčanie: po prijatí XML dát je vhodné využívať online nástroje na vypísanie stromovej štruktúry - využitie tzv. *beautify* funkcionality. Jedným z online nástrojov je napr. https://codebeautify.org/xmlviewer, ktorý je možné vidieť na obrázku 1.12:



Obrázok 1.12 Online nástroj pre prehľadnejší výpis rozsiahlych XML vstupov

Začnime jednoduchým filtrom:

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
      <filter>
         <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"/>
      </filter>
   </get>
</rpc>
]]>]]>
```

Začiatok a koniec filtra je označený tag-om „`filter`“, ktorý sa nachádza medzi tag-mi operácie, ktorú chceme vykonať. V tomto prípade sa jedná o operáciu GET. Na vytvorenie filtra potrebujeme poznať dve informácie → názov kontajnera YANG modelu a tzv. namespace, ktorý je možné získať z textového súboru pre daný YANG model. Výstup uvedeného filtra je zobrazený na nasledujúcom výpise:

192.168.2.6 – **Tera Term VT**

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"/>
    </filter>
  </get>
</rpc>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103"><data><interfaces xmlns
="urn:ieft:params:xml:ns:yang:ieft-interfaces"><interface><name>GigabitEthernet1</name><description
>VBox</description><type xmlns:ianaift="urn:ieft:params:xml:ns:yang:iana-if-type">ianaift:ethernetC
smacd</type><enabled>true</enabled><ipv4 xmls="urn:ieft:params:xml:ns:yang:ieft-ip"><address><ip>19
2.168.56.101</ip><netmask>255.255.255.0</netmask></address></ipv4><ipv6 xmls="urn:ieft:params:xml:n
s:yang:ieft-ip"></ipv6></interface><interface><name>GigabitEthernet2</name><description>Real_Bridge
d</description><type xmlns:ianaift="urn:eitf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmac
d</type><enabled>true</enabled><ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"></ipv4><ipv6 xmlns
="urn:ietf:params:xml:ns:yang:ietf-ip"></ipv6></interface><interface><name>Loopback1</name><descrip
tion>loopback_test</description><type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">iana
ift:sofwareLoopback</type><enabled>true</enabled><ipv4 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip">
<address><ip>1.1.1.1</ip><netmask>255.255.255.255</netmask></address></ipv4><ipv6 xmlns="urn:ieft:p
arams:xml:ns:yang:ietf-ip"><address><ip>2001:db8:1111:1111::1</ip><prefix-length>64</prefix-length>
</address></ipv6></interface><interface><name>Loopback22</name><type xmlns:ianaift="urn:ieft:params
:xml:ns:yang:iana-if-type">ianaift:softwareLoopback</type><enabled>true</enabled><ipv4 xmlns="urn:i
eft:params:xml:ns:yang:ietf-ip"><address><ip>2.2.2.2</ip><netmask>255.255.255.255</netmask></addres
s></ipv4><ipv6 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip"><address><ip>2001:db8:2222::1</ip><prefi
```

```
x-length>64</prefix-length></address></ipv6></interface><interface><name>Loopback3</name><type xmln
s:ianaift="urn:ieft:params:xml:ns:yang:iana-if-type">ianaift:softwareLoopback</type><enabled>true</
enabled><ipv4 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip"></ipv4><ipv6 xmlns="urn:ieft:params:xml:n
s:yang:ietf-ip"></ipv6></interface><interface><name>Loopback8</name><type xmlns:ianaift="urn:ieft:p
arams:xml:ns:yang:iana-if-type">ianaift:softwareLoopback</type><enabled>true</enabled><ipv4 xmlns="
urn:ieft:params:xml:ns:yang:ietf-ip"><address><ip>8.8.8.8</ip><netmask>255.255.255.0</netmask></add
ress></ipv4><ipv6 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip"></ipv6></interface><interface><name>L
oopback10</name><type xmlns:ianaift="urn:ieft:params:xml:ns:yang:iana-if-type">ianaift:softwareLoop
back</type><enabled>true</enabled><ipv4 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip"><ipv4><ipv6 xml
ns="urn:ieft:params:xml:ns:yang:ietf-ip"></ipv6></interface><interface><name>Loopback13</name><type
xmlns:ianaift="urn:ieft:params:xml:ns:yang:iana-if-type">ianaift:softwareLoopback</type><enabled>tr
ue</enabled><ipv4 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip"><address><ip>1.2.3.5</ip><netmask>255
.255.255.0</netmask></address></ipv4><ipv6 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip"><address><ip
>2001:db8:0:13::13</ip><prefix-length>64</prefix-length></address></ipv6></interface><interface><na
me>Loopback20</name><type xmlns:ianaift="urn:ieft:params:xml:ns:yang:iana-if-type">ianaift:software
Loopback</type><enabled>true</enabled><ipv4 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip"><address><i
p>20.20.20.20</ip><netmask>255.0.0.0</netmask></address></ipv4><ipv6 xmlns="urn:ieft:params:xml:ns:
yang:ietf-ip"></ipv6></interface><interface><name>Loopback22</name><type xmlns:ianaift="urn:ieft:pa
rams:xml:ns:yang:iana-if-type">ianaift:softwareLoopback</type><enabled>true</enabled><ipv4 xmlns="u
rn:ieft:params:xml:ns:yang:ietf-ip"><address><ip>22.22.22.22</ip><netmask>255.255.255.0</netmask></
address></ipv4><ipv6 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip"><address><ip>2001:db8:0:22::22</ip
><prefix-length>64</prefix-length></address></ipv6></interface><interface><name>Loopback23</name><t
ype xmlns:ianaift="urn:ieft:params:xml:ns:yang:iana-if-type">ianaift:softwareLoopback</type><enable
d>true</enabled><ipv4 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip"><address><ip>23.22.22.22</ip><net
mask>255.255.255.0</netmask></address></ipv4><ipv6 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip"><add
ress><ip>2001:db8:0:23::23</ip><prefix-length>64</prefix-length></address></ipv6></interface></inte
rfaces></data></rpc-reply>]]>]]>
```


Ako je možné vidieť, využitím filtra nám boli odoslané len informácie o rozhraniach. Výpis je pre bežného človeka značne nečitateľný, preto pre lepšiu orientáciu je vhodné prijaté dáta skopírovať a vložiť do online *Viewer* nástroja. Tento nástroj dáta prevedie do čitateľnejšej podoby. Kopírovať treba časť od <rpc-reply> ... </rpc-reply>

Výsledok zobrazený v Beautify XML je nasledovný:

```
<rpc-reply
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103">
   <data>
      <interfaces
         xmlns="urn:ieft:params:xml:ns:yang:ieft-interfaces">
         <interface>
            <name>GigabitEthernet1</name>
            <description>VBox</description>
            <type
               xmlns:ianaift="urn:ieft:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd
            </type>
            <enabled>true</enabled>
            <ipv4 xmls="urn:ieft:params:xml:ns:yang:ieft-ip">
               <address>
                  <ip>192.168.56.101</ip>
                  <netmask>255.255.255.0</netmask>
               </address>
            </ipv4>
            <ipv6 xmls="urn:ieft:params:xml:ns:yang:ieft-ip"></ipv6>
         </interface>
         <interface>
            <name>GigabitEthernet2</name>
            <description>Real_Bridged</description>
            <type
               xmlns:ianaift="urn:eitf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd
            </type>
            <enabled>true</enabled>
            <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
            </ipv4>
            <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
            </ipv6>
         </interface>
         <interface>
            <name>Loopback1</name>
            <description>loopback_test</description>
```

```
            <type
               xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:sofwareLoopback
            </type>
            <enabled>true</enabled>
            <ipv4 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip">
               <address>
                  <ip>1.1.1.1</ip>
                  <netmask>255.255.255.255</netmask>
               </address>
            </ipv4>
            <ipv6 xmlns="urn:ieft:params:xml:ns:yang:ietf-ip">
               <address>
                  <ip>2001:db8:1111:1111::1</ip>
                  <prefix-length>64</prefix-length>
               </address>
            </ipv6>
         </interface>
         <interface>
            <name>Loopback22</name>
            <type
               xmlns:ianaift="urn:ieft:params:xml:ns:yang:iana-if-type">ianaift:softwareLoopback
            </type>
            <enabled>true</enabled>
```

V tomto objekte je zobrazený kompletný výpis z nástroja TeraTerm, pre jeho rozsiahlosť v dokumente je však objekt zmenšený.

Ak by sme chceli zobraziť informácie o nejakom inom YANG objekte - postup je úplne rovnaký, stačí vyhľadať názov kontajnera a namespace a tieto hodnoty uviesť do filtra.

Niekoľko príkladov:

```
<acl xmlns="http://openconfig.net/yang/acl"/>
<interfaces xmlns="http://openconfig.net/yang/interfaces"/>
<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"/>
<interfaces-state xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"/>
<routing-state xmlns="urn:ietf:params:xml:ns:yang:ietf-routing"/>
<access-lists xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-acl-oper"/>
```

Všetky ukážky predstavujú základný filter, ktorý zobrazí všetky údaje z daného YANG modelu. Podobne ako pri protokole RESTCONF vieme aj v protokole NETCONF robiť granulárnejšie filtre, ktoré nám vedia zobraziť napr. niektoré konkrétne hodnoty o danom objekte.

Napr. ukážka filtra pre zobrazenie názvu zariadenia z Native modelu vyzerá nasledovne:

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
      <filter>
         <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
            <hostname>
            </hostname>
         </native>
      </filter>
   </get>
</rpc>
```

Využitím tag-ov je možné definovať požadované údaje, ktoré chceme prijať. V tomto prípade je ale potrebné aj samotný kontajner, v ktorom sa tieto údaje nachádzajú, ukončiť využitím dvojice tag-ov.

Pre úplnosť, si uveďme filter, ktorý zobrazí pre rozhranie GigabitEthernet1 informáciu o jeho popise a IPv6 nastavenia. Pre Loopback22 naopak zobrazí len informáciu o IPv4 konfigurácii.

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
      <filter>
         <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
            <interface>
               <GigabitEthernet>
                  <name>1</name>
                  <description></description>
                  <ipv6></ipv6>
               </GigabitEthernet>
               <Loopback>
                  <name>22</name>
                  <ip></ip>
               </Loopback>
            </interface>
         </native>
      </filter>
   </get>
</rpc>
```

Ako je možné vidieť, tvorba filtrov je do veľkej miery škálovateľná. Jediným dôležitým krokom je zistiť ako sa nazývajú jednotlivé tag-y. Túto informáciu získame vykonaním GET žiadosti bez akýchkoľvek špecifických filtrov.

V prípade, ak chceme vykonávať konfiguračné úpravy na zariadení, tak je potrebné spraviť malé zmeny v posielaných XML správach. Metódu „`get`“ zmeníme na metódu „`edit-config`“. Ďalej pridáme časť `<target>`, kde ako argument uvedieme časť zariadenia, v ktorom chceme vykonať zmeny (`running|startup|candidate`). Poslednou časťou je prepísanie tag-u „`filter`“ na tag „`config`“, kde v tele musí byť definovaná nová hodnota, ktorú chceme upraviť.

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config>
         <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
            <hostname> TEST_Hostname_NETCONF </hostname>
         </native>
      </config>
   </edit-config>
</rpc>
```

V prípade úspešnej operácie bude v konzole SSH pripojenia zobrazená správa:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103"><ok/></rpc-reply>
]]>]]>[]
```

Podobne sa na spravovanom zariadení vypíše na konzolu správa o vykonaní transakcie:

```
*Jul 8 02:58:48.550: %DMI-D-CONFIG_I: F0: nesd:  Configured from NETCONF/RESTCONF by cisco,
transaction-id 210032_
```

Z príkladov je možné odvodiť, že práca s YANG modelom, využitím protokolu NETCONF, je zo začiatku náročnejšia, no po znalosti niekoľkých pravidiel sa postupne stane jednoduchšou na pochopenie a následné praktizovanie.

Uvedený prístup komunikácie cez TeraTerm je síce možný, no podobne ako v prípade nástroja Postman, tento prístup neumožňuje automatizáciu a priame prepojenie na ďalšie systémy. Pre účely automatizácie je možné využiť programovací jazyk Python a knižnicu *ncclient*, ktorá umožní komunikáciu cez NETCONF protokol.

Nasledujúci zdrojový kód znázorňuje prácu s NETCONF protokolom pre čítanie názvu zariadenia:

```
#definovanie údajov na pripojenie na NETCONF
HOST = '192.168.56.101'
USER = 'cisco'
PASS = 'cisco123!'
PORT = 830

#filter pre prácu s názvom zariadenia
netconf_filter = """
<filter>
   <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
      <hostname>
      </hostname>
   </native>
</filter>
"""
#vystvorenie NETCONF spojenia a odoslanie GET_Config požiadavky
m = manager.connect(host=HOST, port=PORT, username=USER, password=PASS,
hostkey_verify=False, device_params={'name': 'csr'})
reply = m.get_config(source='running', filter=netconf_filter)

#parsovanie XML a odoslanie získaného názvu zariadenia do Webex Teams
result = xmltodict.parse(reply.xml)
post_to_webex(result["rpc-reply"]["data"]["native"]["hostname"])

#kontrolný výpis xml prijatých dát
#print(xml.dom.minidom.parseString(reply.xml).toprettyxml())

#uzatvorenie spojenia
m.close_session()
```

Na vytvorenie spojenia je potrebné definovať autentifikačné údaje, IP adresu cieľa a port pre NETCONF. Následne sa už len využívajú funkcie `manager.connect()` na vytvorenie spojenia a `get_config()` na odoslanie správy. Ako si môžeme všimnúť, tak dôležitou časťou je aj v tomto prípade využitie filtrov, ktorým sme sa venovali v predchádzajúcich častiach.

Podobne sa postupuje aj v prípade, ak chceme na zariadení nastavenia meniť. Jedinou zmenou je, že nepoužijeme filter ale `config` tag a funkcia, ktorú využijeme sa volá `edit_config()`. Zdrojový kód pre vytvorenie/zmenu rozhrania je nasledovný:

```
#definovanie údajov na pripojenie na NETCONF
HOST = '192.168.56.101'
USER = 'cisco'
PASS = 'cisco123!'
PORT = 830

#konfiguračný filter
netconf_data = netconf_data = """
<config>
   <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
      <interface>
         <"""+mylist[1]+""">
            <name>"""+mylist[2]+"""</name>
            <ip>
               <address>
                  <primary>
                     <address>"""+mylist[3]+"""</address>
```

```
                    <mask>“““+mylist[4]+“““</mask>
                  </primary>
               </address>
            </ip>
            <ipv6>
               <address>
                  <prefix-list>
                     <prefix>“““+mylist[5]+“/“+mylist[6]+“““</prefix>
                  </prefix-list>
               </address>
            </ipv6>
         </“““+mylist[1]+“““>
      </interface>
   </native>
</config>
“““
#vystvorenie NETCONF spojenia a odoslanie EDIT_Config požiadavky
m = manager.connect(host=HOST, port=PORT, username=USER, password=PASS,
hostkey_verify=False, device_params={‘name‘: ‘csr‘})
reply = m.edit_config(target=‘running‘, config=netconf_data)
#uzatvorenie spojenia
m.close_session()
```

Zhrnutie na záver:

Protokoly NETCONF a RESTCONF sú primárne vytvorené na prácu s konfiguráciou zariadení. Isté i keď obmedzené možnosti, sú ponúkané aj z pohľadu prístupu k stavovým informáciám. Medzi tie patrí napríklad samotný stav rozhrania, vyťaženosť procesora a pamäte, informácie o smerovacej tabuľke a pod. Podstatnou informáciou však ostáva, že NETCONF a RESTCONF sú komunikačné protokoly, ktoré nám prostredníctvom API umožňujú pristupovať k zariadeniam automatizovaným spôsobom. Samotnú reprezentáciu vlastností zariadení zabezpečujú YANG modely, ktoré definujú štruktúru dát. Z pohľadu formátovania a manipulácie dát využívame dáta vo forme JSON a XML.