Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Antti Puolakka

Muutostöiden automatisointi ohjelmallisesti verkkolaitteessa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

19.02.2022

Tekijä Otsikko Sivumäärä Aika	Antti Puolakka Muutostöiden automatisointi ohjelmallisesti verkkolaitteessa 34 sivua + 1 liite 19.02.2022
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine	Verkot ja pilvipalvelut
Ohjaajat	Lehtori Marko Uusitalo

Opinnäytetyön aiheena oli tutustua tietoliikennelaitteiden kuten palomuurien, kytkinten ja reitittimien automaatioon. Aluksi työssä käsiteltiin verkkolaitteidenautomaation teoriaa, jolla automatisointi on mahdollista toteuttaa. Teoriaa hyödyntämällä tehtiin pienoisohjelma Python-ohjelmointikielellä, jolla on mahdollista muuttaa tai lukea konfiguraatiotietoa verkkolaitteelta.

Tavoitteena oli luoda teorian pohjalta ohjelma ja tutustua eri vaihtoehtoihin, joilla verkkolaitteiden muutoksia on mahdollista automatisoida ja yhtenäistää. Työn aikana tutustuttiin useisiin eri vaihtoehtoihin, joita käytetään verkkolaiteympäristöä automatisoidessa. Työn aikana ohjelman laajuus kasvoi ja mahdollisti eri käytäntöjen käytön ohjelman päätarkoitukseen eli konfiguraation muokkaamiseen verkkolaitteesta.

Työn tuloksena syntyi ohjelma, joka mahdollistaa muutosten tekemisen verkkolaitteeseen ohjelmallisesti. Ohjelmalla on myös mahdollista pelkästään tutkia verkkolaitteen konfiguraatiotietoa. Ohjelman funktiopohjainen suunnittelu mahdollistaa ohjelman jatkokehityksen sekä uusien ominaisuuksien tuonnin ohjelmaan tulevaisuudessa.

Avainsanat	Automatisointi, tietoliikenne

Author	Antti Puolakka
Title	Programmatically editing network device configuration
Number of Pages	34 pages + 1 appendices
Date	19 February 2022
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	IoT and Cloud Computing
Instructors	Marko Uusitalo, Senior Lecturer

The goal of this final year project was to create a program that would help in automating changes to network devices such as switches, routers and firewalls. Thesis explores different options on how to automate changes in network devices and familiarizes with the technologies that are automating modern network environments.

First thesis focuses on the theory behind different technologies that automate network devices. Thesis has theory for multiple different angles in which automation is possible to implement into network environment.

As the result of the thesis a small program was written. Script enables automating small changes in different network devices. Program also allows reading configuration data from network devices.

This final year project gives a good starting point in how to begin automating network changes in network environments. Program has the basic functions to operate and command devices. Program code is also modifiable in the future.

Keywords	Network, automation

Sisällys

Lyhenteet

API	Application programming interface. Ohjelmointirajapinta, jolla ohjelmat voivat siirtää tai pyytää tietoa toiselta ohjelmalta.
HTTP	Hypertext Transfer Protocol. Tiedonsiirtoprotokolla selaimille.
IETF	The Internet Engineering Task Force. Kansainvälinen avoin yhteisö, joka vastaa internetin protokollien standardisoinnista ja kehityksestä.
SNMP	Simple Network Management Protocol. Protokolla tiedonsiirtoon ja -organisointiin verkkolaitteille.
MIB	Management information base. Tietokanta itsenäisten kokonaisuuksien hallitsemiseen tietoverkkolaitteilla.
TCP	Transmission Control Protocol. Tietoliikenneprotokolla luotettavaan tiedonsiirtoon laitteelta toiselle.
UDP	User Datagram Protocol. Tietoliikenneprotokolla tiedonsiirtämiseen laitteelta toiselle.
SOAP	Simple Object Access Protocol. Tietoliikenneprotokolla sovellusten viestimiseen.
BEEP	Blocks Extensible Exchange Protocol. Verkkosovellusten tiedonvälitykseen käytettävä protokolla.
TLS	Transport Layer Security. Salausprotokolla tiedon turvalliseen siirtämiseen.

1 Johdanto

Insinöörityön tavoitteena on tutustua tietoliikenneverkkoja ylläpitävien laitteiden automatisointiin ja luoda opittuun tietoon pohjautuva pieni ohjelma. Ohjelman tarkoituksena on hahmottaa, kuinka automatisoinnin lisääminen tietoliikenneverkoissa vaikuttaa verkon vikasietoisuuteen, helpottaa vianselvityksessä ja nopeuttaa laajoihin verkkokokonaisuuksiin suunnatuissa muutoksissa.

Automaation merkitys nopeasti reagoivissa tietoliikenneverkoissa kasvaa jatkuvasti. Tietoliikenneverkon automatisointi pienentää inhimillisten virheiden määrää huomattavasti ja luo vikasietoisemman pohjan verkkoliikenteelle. Tarve automaattiselle verkonhallinnalle minimoimalla työkuormaa näkyy jatkuvasti enemmän verkkolaitteissa.

Käyn opinnäytetyössäni läpi yleistyvät käytännöt, joilla voi helpottaa verkon ylläpitoa automatisoimalla yleisimmät pienet aikaa ja tarkkuutta vaativat tehtävät sekä automaatiotyökalut, joita verkkoympäristöjen hallinnoinnissa ja ylläpitämisessä käytetään. Sovellan esiteltyjä työkaluja insinöörityöhöni sopivaksi.

Luon työssäni ohjelman, joka hakee konfiguraatiotietoa verkkolaitteesta. Toteutan työn esittelemilläni työkaluilla ja hyödynnän automaatiossa suosittuja ja hyväksi todettuja käytäntöjä. Tavoitteena on myös konfiguraatiotiedon muokkaaminen verkkolaitteesta ohjelmallisesti.

2 Automaatio

Automaatiolla tarkoitetaan teknologian käyttämistä suorittamaan tehtäviä vähemmällä ihmisen avustuksella. Mikä tahansa tuotannonala, jolla työtehtävät muodostuvat yhtenäisistä ja toistuvista työtehtävistä, hyötyy prosessien ja tehtävien automatisoinnista. Automaatio on suuressa merkityksessä teollisuudessa, kuten valmistuksen, robotiikan ja informaatioteknologian aloilla. [1.]

Informaatioteknologian alalla automaatiolla viitataan yleensä toistuvien ja suoraviivaisten työtehtävien ohjeistamiseen tai prosessoimiseen koneellisesti. Automaatiolla voidaan siis vähentää tai kokonaan poistaa ihmisen tekemä manuaalinen työ. Digitaalisessa muutoksessa olevalle yritykselle automatisointi tarkoittaa prosessien virtaviivaistusta sekä eri automaatioteknologioiden käyttöönottoa. [2.]

Tietoliikenneverkon automaatiolla tarkoitetaan resurssien eli niin fyysisten kuin virtuaalisten laitteiden, konfiguroinnin, ylläpidon, hallinnan ja operoinnin automatisoimista. Erilaisten työtehtävien, kuten konfiguraationhallinnan, ympäristöjen integroimisen, laitteiden tai sovellusten käyttöönoton prosessien automatisointi tehostaa yrityksen tuottavuutta ja hyötysuhdetta. [2.]

Verkon automaatio jakaantuu kahteen ryhmään, joita ovat komentoliittymällä ajettavat ohjelmat ja graafisen käyttöliittymän omaavat ohjelmat. Komentoliittymällä ajettavat komentojono-ohjelmat vaativat aiempaa ohjelmointikielien tuntemusta, mutta kuka tahansa voi tehdä omiin tarpeisiinsa tarkoitettuja pieniä ohjelmia hyödyntäen komentoliittymällä ajettavia komentoja. Graafiseen käyttöliittymään perustuvat automatisointiohjelmat toimivat usein laitteiden ohjelmoitavilla rajapinnoilla, minkä takia vanhemman sukupolven laitteet, joihin ei ole enää saatavilla päivityksiä tai ohjelman ja laitteen yhdistävää rajapintaa, ovat vaikeampia sulauttaa automatisoituun ympäristöön. Vanhemman sukupolven laitteiden automatisointikyky jää siis komentoliittymällä ajettavien komentojen ja näiden päälle rakennettujen sovellusten varaan.

Yritysten digitalisoituessa, järjestelmien kehittyessä ja teknisten kokonaisuuksien kasvavassa muutoksessa tekniikkaa on automatisoitu vastaamaan kasvavaa tarvetta. Tietoverkkojen hallinta on ollut raskasta ja kankeaa, sillä useat muutokset ja päivitykset on

toteutettu manuaalisesti. Tämä on aiheuttanut viivettä palveluiden kehityksessä ja uusien palvelujen käyttöönotossa, koska useat muutokset ja uudistukset yritysten tietoverkoissa ovat kestäneet useita päiviä. Verkon automatisointi siis tukee prosesseja ja palveluiden kehitystä tekemällä muutoksista nopeampia toteuttaa, ja samalla myös vikasietoisempia vähentämällä inhimillisten virheiden määrää. [3.]

2.1 Automaation merkitys

Informaatioteknologian alalla prosesseja ja manuaalista työtä on automatisoitu eri menetelmillä jo useita vuosikymmeniä. Helppojen muutosten tai toistuvien pienten työtehtävien automatisointi on merkittävässä roolissa yrityksille, jotka haluavat panostaa palveluidensa kehittämiseen ylläpidon ohella. Automaatiolla voidaan nostaa merkittävästi palveluiden tasoa, laatua ja tehokkuutta. [4.]

Ohjelmoinnilla saavutettu automaatio tietoliikenneverkkojen ylläpitämisessä ei ennen ole tullut tarpeen ohjelmointiin käytetyn ajan ja siitä saatavan hyödyn näkökulmasta [5]. Nykyään saatavilla on valtava määrä eri moduuleita, kirjastoja ja valmiiksi tehtyjä erilaisia resursseja. Myös laitevalmistajat ovat kehittäneet eri menetelmiä laitteisiinsa, joilla automaatiota voi kehittää ja lisätä tietoliikenneverkkoihin. [6.]

Tietoliikenneverkkojen ylläpidon automatisoinnin on pelätty vievän monia työpaikkoja alalta, jossa tietyn laitevalmistajan laitteen komentoliittymän osaaminen on ollut työntekijälle hyödyksi. Työpaikkojen väheneminen vaikuttaa kuitenkin hyvin epätodennäköiseltä, koska alalla työskentelevillä on tarvittava osaaminen tietoliikenneverkon segmentointiin, erilaisten reititysprotokollien käyttöönottoon ja pääsylistan luomiseen haitallisen liikenteen eliminoimiseen tuotantokriittisestä verkosta. Työntekijöiden asiantuntemusta tarvitaan siis tulevaisuudessa haastavimpiin tehtäviin, kun taas toistuvat ja suoraviivaiset tehtävät saadaan automatisoitua uusilla työkaluilla ja ohjelmilla. Automatisointityökalujen ja -ohjelmien ylläpitoon tarvitaan samat henkilöt, jotka tekivät samoja tehtäviä manuaalisesti aikaisemmin. [7.]

Laitevalmistajat parantavat jatkuvasti ominaisuuksia, joilla he pyrkivät parantamaan kilpailukykyänsä tarjoamalla erilaisia työkaluja laitteiden ylläpitämiseen, konfigurointiin ja vianselvitykseen. Uusimpana ominaisuutena on esimerkiksi rajapinta koneiden välillä

(API). Rajapintaa hyödyntämällä voidaan verkkolaitteita konfiguroida laajoissakin ympäristöissä hyvin ketterästi poistamalla yhtälöstä inhimilliset virheet ja tekemällä toistuvat tehtävät automaattisesti. [8.]

Ohjelmointikielien nykyaikaistuessa mahdollisuudet uusien teknologioiden käyttöönotolle ovat paremmat kuin koskaan aikaisemmin. Verkkoympäristöissä yksi suosituimmista automatisoinnissa käytetyistä ohjelmointikielistä on Python. Ohjelmointikieliä kuten JavaScript ja Perl käytetään myös verkkolaitteiden automatisoinnissa, mutta Pythonin helpon syntaksin ja valtavan suosion takia on Pythonille löydettävissä monia erilaisia kirjastoja, joita verkonsuunnittelussa ja automaation integroimisessa tietoliikennelaitteille voidaan käyttää [9; 10].

Automaatio nopeuttaa tietoliikenneverkkoa ylläpitävien tahojen reagointikykyä mahdollisiin verkon ongelmatilanteisiin ja auttaa tahoja optimoimaan verkossa tapahtuvia muutoksia.

Tietoliikenneverkkoon integroitua automatisointijärjestelmää on vaivatonta huoltaa ja päivittää. Tietoliikenneverkon automatisointijärjestelmät mahdollistavat paremmat työkalut verkon valvontaan. Tietoliikenneverkkoa valvovat automatisointijärjestelmät tarjottavat paremman näkyvyyden verkkoliikenteen kulkuun ja automatisoidussa ympäristössä verkkoliikenteen ongelmatilanteet on helpompi ratkaista. [11.]

3 Automaatiotyökalut

3.1 Python

Python on yksi suosituimpia tietoliikenneverkoissa käytettäviä ohjelmointikieliä [12]. Verkkoja optimoi yhä useampi verkkoinsinööri ohjelmilla ja koodeilla, jotka on kirjoitettu Pythonilla. Verkkoinsinöörinä toimivan henkilön, jonka ei ole aikaisemmin tarvinnut ohjelmoida, on vaivatonta käyttää Pythonia ohjelmointikielenä sen helpon syntaksin ja nopean toteuttamisen vuoksi. Muilla ohjelmointikielillä, kuten C:llä tai Javalla tarvitsee kirjoittaa useita satoja rivejä koodia, jotta saadaan haluttu lopputulos, mutta Pythonilla ohjelmoidessa tarvitaan vain murto-osa tuosta koodista. [13.]

Pythonin käyttötarkoitusten ja skaalautuvien ohjelmien myötä yhä useammissa työpaikkalistauksissa alkaa näkymään vaatimus tai tarve Pythonin osaamiselle. Pythonin suosion ja käytettävyyden takia on sille luotu tuhansia esimerkkipienohjelmia, joita voi oman käyttötarkoituksen mukaan ladata eri lähteistä, kuten esimerkiksi GitHub:sta. [14.]

Automatisoinnin aloittaminen ohjelmoimalla omia pienohjelmia ja sovelluksia on helppoa ja mutkatonta. Ohjelmointiin tarvitsee vain tekstinmuokkausohjelman, kuten Windows-koneista löytyvän muistio-ohjelman ja Python-ohjelman. [15.]

```
$
>>> print(“Hello world!”)
Hello world!
```

Esimerkkikoodi 1. Ensimmäinen koodi Pythonilla, jonka lähes jokainen ohjelmoija kirjoittaa aloittaessaan koodaamisen.

3.1.1 Python-kirjastot

Python-kirjastot ovat ohjelmaan tuotuja erilaisia ohjelmakoodin osia tai kokonaisia kirjastoja ohjelmista. Kirjastot tarjoavat ominaisuuksia tai tietoa, jota ohjelma tarvitsee toimiakseen. Ohjelmankehitys Python-kirjastoilla on nopeampaa kuin itse ohjelmoidut toiminnot kirjastoiden helpon saatavuuden vuoksi. Python-kirjaston liittäminen omaan ohjelmaan tapahtuu käyttämällä Pythoniin sisältyviä tuontisanoja ”import” ja ”from – import”. [16.]

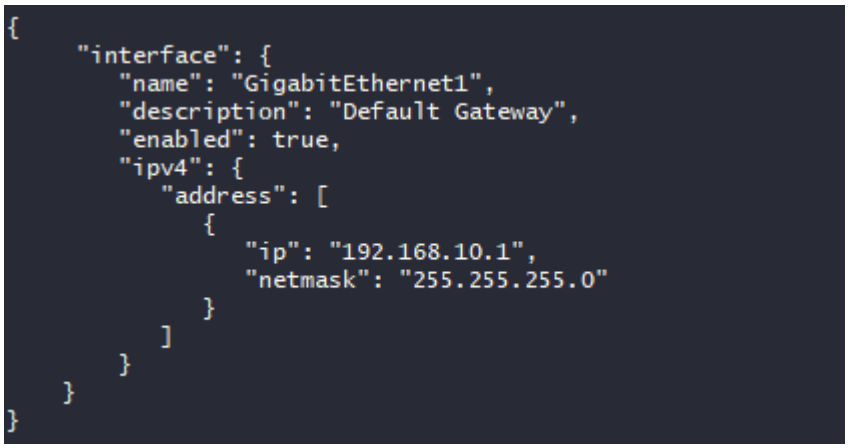
Käytännöllisimmät kirjastot verkkolaitteiden dataa käsitteleville ohjelmille ovat xmltodict, json, yaml ja csv. Ohjelmoitavaa rajapintaa varten kirjastot requests, ncclient ja netmiko tuovat monia tärkeitä ominaisuuksia ohjelmaan. [17.]

3.2 Tietorakenteet

Tietorakenteet on suunniteltu liikuttamaan ja säilömään tietoa. Useat tietorakenteet, joita käytetään verkkolaitteiden automatisoinnissa, on suunniteltu siten, että niitä pystyy lukemaan niin ihminen kuin ohjelma tai koodi. [18.]

3.2.1 JavaScript Object Notation eli JSON

JSON on yksinkertainen ihmisluettavassa muodossa oleva tietorakenne, jota käytetään tiedon tallentamiseen ja siirtämiseen. JSON-tietorakenne perustuu avain-arvo pareihin, mikä tekee rakenteesta helposti tulkittavan. Verkkolaitteita ohjelmoitaessa JSON-tietorakenne on hyvin keskeisessä asemassa.



Kuva 1. JSON-tietorakenne verkkolaitteen GigabitEthernet1-kytkennästä.

JSON:lle löytyy oma kirjasto Pythonista. Kirjastoa hyödyntämällä saa luotua helposti ohjelman, joka muuntaa JSON-tietoa Python-sanastoksi ja takaisin JSON-muotoon. Tiedon muuntamiseen on neljä eri funktiota, mikäli JSON-kirjasto on liitetty Python-ohjelmaan.

- load()
- loads()
- dump()
- dumps().

Funktioiden perässä oleva *s* viittaa string-datatyyppiin Pythonissa. [19.]

3.2.2 Extensible Markup Language eli XML

XML on yksinkertainen tietorakenne, jota käytetään verkkolaitteiden automatisoinnissa tiedon siirtämiseen. Samankaltaisesti kuten HTML-kielessä käytetään XML:ssä myös jäsentimiä tiedonhallintaan ja tiedonsiirtoon. XML-konfigurointitiedostoa pystyy liikuttamaan helposti käyttäen standardeja internetprotokollia, kuten http:tä ja https:ää. [20.]

```
<?xml version="1.0" encoding="UTF-8" ?>
<interface xmlns="ietf-interfaces">
 <name>GigabitEthernet2</name>
 <description>Office LAN</description>
 <enabled>true</enabled>
 <ipv4>
    <address>
        <ip>192.168.10.1</ip>
        <netmask>255.255.255.0</netmask>
        </address>
 </ipv4>
</interface>
```

Kuva 2. Esimerkki XML:n tiedostomuotoilusta.

3.2.3 YAML

YAML on suosittu ja selkeäkielinen tallennusmuoto konfiguraatiotiedostojen luomiseen ja tiedontallennukseen. YAML kehitettiin samoihin käyttötarkoituksiin kuin XML, mutta sillä on helpompi syntaksi ja luettavampi muoto kuin XML:llä.

```
pip install pyyaml
```

Esimerkkikoodi 2. Muuntaakseen dataa YAML-tiedostomuodosta Pythonin datatyypeiksi pitää ohjelmaan lisätä pyyaml-kirjasto.

```
import yaml
```

Esimerkkikoodi 3. Lisätään YAML-kirjasto koodiin.

Tiedon muuntamiseksi YAML-muodosta Pythonin eri tietomuotoihin tuodaan PyYaml-kirjastosta funktio yaml.load. Tämä muuttaa YAML-tiedostosta saadun tiedon Python-ohjelman tulkittaviksi objekteiksi. Pythonista saa tiedon takaisin YAML-tiedostomuotoon käyttämällä yaml.dump-funktiota, joka löytyy myös PyYaml-kirjastosta. [21.]

3.3 Ohjelmointirajapinta API

Ohjelmointirajapinta eli API on ohjelmointiliitäntä, jota sovellukset käyttävät kommunikoidakseen muiden laitteiden, kuten palvelimien tai muiden sovellusten ja ohjelmien kanssa. Monien modernien sovellusten toiminta perustuu ohjelmointirajapinnan käyttöön. API:t ovat olemassa, jotta integrointi uusiin palveluihin, sovelluksiin, ohjelmiin tai laitteisiin olisi sujuvampaa. [22.]

3.3.1 API verkko-ohjelmoinnissa

Verkko-ohjelmoinnissa ohjelmointirajapintaa käytetään verkkolaitteiden, kuten kytkinten, reitittimien ja palomuurien väliseen kommunikoimiseen, tiedonvälitykseen ja konfiguroimiseen. Samaa rajapintaa voidaan hyödyntää myös tuotantokriittisten verkkoyhteyksien tai verkkolaitteiden valvonnassa. [23.]

3.3.2 REST API

REST (Representational State Transfer) tarkoittaa tietyllä arkkitehtuurityylillä luotua ohjelmointirajapintaa. Yleisimmät modernit sovellukset ja uusimmat verkkolaitteet käyttävät REST-ohjelmointirajapintaa. REST on kevyt, skaalautuva ja joustava työkalu automaation integroimiseen verkkoympäristössä. REST-ohjelmointirajapinta käyttää http-protokollaa kerätäkseen ja siirtääkseen tietoa sovelluksesta tai laitteesta toiseen. Ennalta määritellyn rakenteen takia http-protokollassa ovat monet laitevalmistajat kehittäneet omia laitteitaan tukemaan API-rakennetta laitteiden hallitsemiseksi. [24.]

- Post
- Get
- Put
- Patch
- Delete.

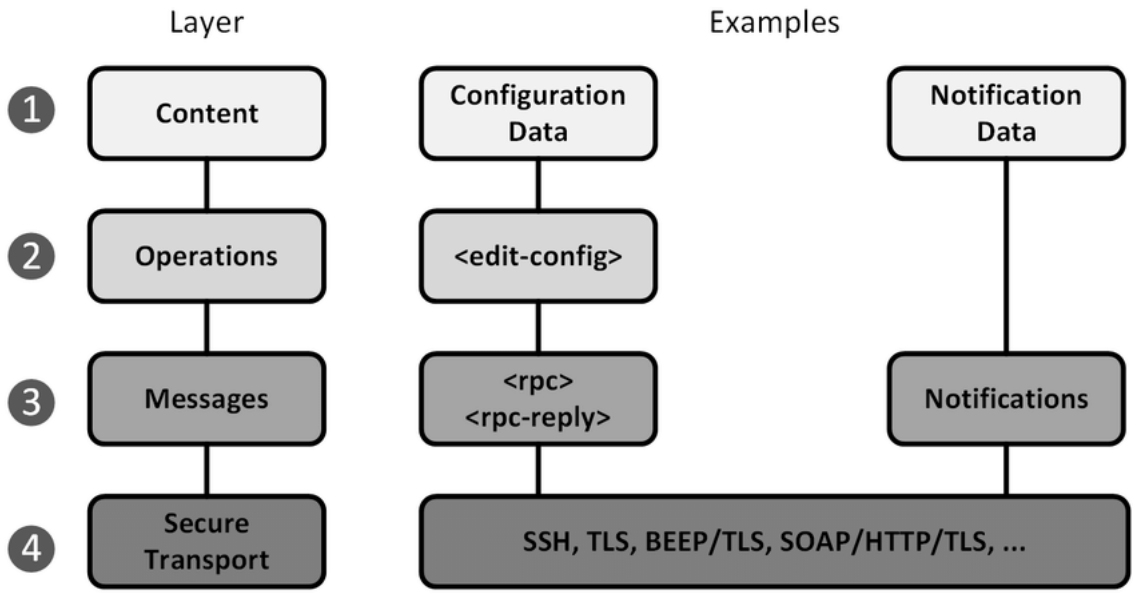
REST-ohjelmointirajapinta käyttää http-protokollan viittä eri funktiota lukeakseen, muuntaakseen tai poistaakseen tietoa kohdepalvelimelta tai sovelluksesta. [25.]

3.4 NETCONF

NETCONF on tietoverkkojen konfigurointiin käytettävä standardisoitu protokolla. Joulukuussa 2006 IETF:n julkaiseman protokollan oli tarkoitus syrjäyttää laitekohtaiset komentoliittymät. NETCONF tarjoaa ohjelmoitavan rajapinnan verkkolaitteelle ja sen konfiguraatioon, poistaa tarpeen laitekohtaisten komentoliittymien syntaksin ymmärtämiselle. [26.]

NETCONF-protokolla mahdollistaa myös konfiguraatiotiedon erittelemisen laitestatistiikasta käyttämällä eri funktioita. SNMP-protokolla, jolla myös haetaan dataa ja statistiikkatietoja verkkolaitteilta, toteuttaa saman yhdellä funktiolla ilman tiedonerittelyä. SNMP-protokollalla haettua tietoa saa verkkolaitteesta valtavan määrän, joten datan tulkitseminen ja tärkeän tiedon etsiminen MIB-muuttujista jää käyttäjän tai sovelluksen tulkittavaksi. [27.]

Tiedonkuljetukseen NETCONF käyttää vikasietoista TCP-tietoliikenneprotokollaa. SNMP käyttää tiedonsiirtoon UDP-tietoliikenneprotokollaa. Ainoastaan SNMP-protokollan uusin versio SNMPv3 tukee tiedonsiirron suojausta. SNMP-protokolla on siksi jäänyt yleisesti laitteiden suorituskyvyn sekä sovelluksien valvontaan. [27.]



Kuva 3. Tasot, jolla NETCONF-protokolla toimii. [28].

Sisältötasoon NETCONF-protokollassa tulee tarvittava data verkkolaitteen hallintaan. Taso on myös ainoa, mitä ei ole standardisoitu, eikä tasolla ole standardia mallinnuskieltä tai tietorakennetta. [29.]

Operaatiotaso määrittelee protokollakohtaiset toiminnot, joilla voidaan hakea, lisätä, kopioida tai poistaa tietoa verkkolaitteen tietokannasta. [30.]

Operation	Description
<get>	Retrieve running configuration and device state information
<get-config>	Retrieve all or part of specified configuration data store
<edit-config>	Loads all or part of a configuration to the specified data store
<copy-config>	Replace an entire configuration data store with another
<delete-config>	Delete a configuration data store
<commit>	Copy candidate data store to running data store
<lock> / <unlock>	Lock or unlock the entire configuration data store system
<close-session>	Graceful termination of NETCONF session
<kill-session>	Forced termination of NETCONF session

Kuva 4. Lista funktioista, joita operaatiotasolla voidaan antaa. [30.]

NETCONF käyttää <rpc>kommunikaatiomallia kehystääkseen pyynnöt ja vastaukset palvelimen ja käyttäjän välillä. [31.]

Tiedon turvallisen siirron NETCONF-protokollassa hoitaa SSH, mutta tarvittaessa tukee myös SOAP-, BEEP- ja TLS-yhteyksiä. [32.]

Taulukko 1. NETCONF-siirtotason protokollat ja oletusportit. [32.]

Protokolla	**Portti**
NETCONF-SSH	TCP / 830
NETCONF-BEEP	TCP / 831
NETCONF-SOAPHTTP	TCP / 832
NETCONF-SOAPBEEP	TCP / 833
NETCONF-TLS	TCP / 6513

NETCONF on istuntopohjainen ja sen on tarjottava tietoa luotettavasti ja järjestelmällisesti. NETCONF ei vahvista palvelimelta saamansa tiedon yhteneväisyyttä tai tarkista saamansa tiedon koskemattomuutta. Tämän takia NETCONF on tiedonsiirtoon käytettävän protokollan varassa. [33.]

3.4.1 YANG

YANG (Yet Another Next Generation) on tiedonmallinnuskieli. YANG tarjoaa standardisoidun mallin verkkolaitteiden konfiguraatiolle ja operatiiviselle tiedolle. YANG on käännettävissä eri tietomuotoihin, kuten XML tai JSON. [34.]

YANG kehitettiin taklaamaan automaatiohaasteet, jotka ilmenevät tietoliikenneverkkojen automatisoinnissa. Laitevalmistajien luomat komentoliittymät ovat kaikki erilaisia sekä vaativat tietämystä laitekohtaisista komennoista, tämän vuoksi usean laitevalmistajan laitteista koostuvan verkon automatisointi ohjelmallisesti on ollut hyvin hankalaa ja hidasta. YANG, jota käytetään tiedonmallintamiseen NETCONF-protokollasta, tuo standardisoidun työkalun konfiguraatiotiedon hallitsemiseen kaikkiin NETCONF-protokollaa tukeviin verkkolaitteisiin. [35.]

4 Verkkolaitteiden muutostöiden automatisoinnin toteutus

Luon insinöörityössä ohjelman, jolla on mahdollista lukea verkkolaitteen porteissa olevaa konfiguraatiodataa. Sovelluksella on mahdollista myös muokata porteissa olevaa konfiguraatiota, sulkea tai avata portteja sekä luoda silmukkakytkentöjä verkkolaitteelle.

Sovelluksen ohjelmointikieleksi valikoitui Python sen helpon syntaksin ja käytettävyyden takia. Atom-sovellus toimii työssä tekstinkäsittelyohjelmana ja Git-versionhallinan sovelluksena. Verkkolaitteiden saatavuuden ja arvokkuuden vuoksi käytin Ciscon tarjoamaa pilvipohjaista Sandbox-ympäristöä ohjelman testaamiseen ja työn toteutukseen.

Insinöörityö vaati valmistelemaan tietokoneen siten, että ohjelmointi onnistuisi vaivatta ja nopeasti. Aloitin lataamalla Git- ja Atom-sovellukset verkosta ja tutustuin näiden käyttöön. Työ tehtiin Python-ohjelmointikielellä ohjelmointikieleen liitettävien kirjastojen takia, millä pystyttiin välttämään ylimääräinen ohjelmointi. Käytin insinöörityössäni Python 3.8.6 -versiota.

4.1 Python-kirjastot virtuaaliympäristössä

Ennen ohjelmoinnin aloittamista valmistelin virtuaaliympäristön käyttäen Pythonia ja Git-bash-komentopäätettä. Virtuaaliympäristö mahdollistaa kirjastojen tuomisen ainoastaan ennalta määritettyyn kohteeseen [36]. Nimeksi annoin tuolle ympäristölle virtualEnv, jotta pystyn työssäni tarkentamaan ja muokkaamaan sovelluksen ympäristöä. Tämän jälkeen aktivoin ympäristön ja varmistin, että luomani virtuaaliympäristö käyttää Pythonin versiota 3.8.6.

```
User@DESKTOP-X MINGW64 ~
$ py -3 -m venv virtualEnv
```

Esimerkkikoodi 4. Syöttämällä tämän komennon Git-komentopäätteeseen luotiin virtuaaliympäristö nimeltä virtualEnv.

```
User@DESKTOP-X MINGW64 ~
$ source VirtualEnv/Scripts/Activate
(virtualEnv)
User@DESKTOP-X MINGW64 ~
$
```

Esimerkkikoodi 5. Virtuaaliympäristö aktivoidaan ja varmistetaan toimivuus (virtualEnv) ennen käyttäjää.

```
User@DESKTOP-X MINGW64 ~
$ python -V
Python 3.8.6
```

Esimerkkikoodi 6. Varmistetaan Pythonin versioksi 3.8.6.

Sovellus käyttää yhdistäessään verkkolaitteisiin ncclient Python-kirjastossa olevaa manager-toiminnallisuutta, sekä netmiko Python-kirjastossa olevaa connecthandler-toiminnallisuutta.

```
(virtualEnv)
User@DESKTOP-X MINGW64 ~
$ pip install ncclient
Collecting ncclient
  Using cached ncclient-0.6.9.tar.gz (118 kB)
Requirement already satisfied: setuptools>0.6 in c:\users\antti\virtu-
alenv\lib\site-packages (from ncclient) (49.2.1)
Collecting paramiko>=1.15.0
  Using cached paramiko-2.7.2-py2.py3-none-any.whl (206 kB)
Collecting lxml>=3.3.0
  Downloading lxml-4.6.2-cp38-cp38-win_amd64.whl (3.5 MB)
Collecting six
  Using cached six-1.15.0-py2.py3-none-any.whl (10 kB)
```

```
Collecting cryptography>=2.5
  Downloading cryptography-3.2.1-cp38-cp38-win_amd64.whl (1.5 MB)
Collecting bcrypt>=3.1.3
  Using cached bcrypt-3.2.0-cp36-abi3-win_amd64.whl (28 kB)
Collecting pynacl>=1.0.1
  Using cached PyNaCl-1.4.0-cp38-cp38-win_amd64.whl (206 kB)
Collecting cffi!=1.11.3,>=1.8
  Downloading cffi-1.14.4-cp38-cp38-win_amd64.whl (179 kB)
Collecting pycparser
  Using cached pycparser-2.20-py2.py3-none-any.whl (112 kB)
Using legacy 'setup.py install' for ncclient, since package 'wheel' is not in-
stalled.
Installing collected packages: six, pycparser, cffi, cryptography, bcrypt,
pynacl, paramiko, lxml, ncclient
    Running setup.py install for ncclient: started
    Running setup.py install for ncclient: finished with status 'done'
Successfully installed bcrypt-3.2.0 cffi-1.14.4 cryptography-3.2.1 lxml-4.6.2
ncclient-0.6.9 paramiko-2.7.2 pycparser-2.20 pynacl-1.4.0 six-1.15.0
(virtualEnv)
User@DESKTOP-X MINGW64 ~
$ pip install netmiko
Collecting netmiko
  Downloading netmiko-3.3.3-py3-none-any.whl (168 kB)
Collecting ntc-templates
  Downloading ntc_templates-2.0.0-py3-none-any.whl (280 kB)
Collecting pyserial
  Downloading pyserial-3.5-py2.py3-none-any.whl (90 kB)
Collecting scp>=0.13.2
  Downloading scp-0.13.3-py2.py3-none-any.whl (8.2 kB)
Collecting tenacity
  Downloading tenacity-7.0.0-py2.py3-none-any.whl (23 kB)
Requirement already satisfied: setuptools>=38.4.0 in c:\users\antti\virtu-
alenv\lib\site-packages (from netmiko) (49.2.1)
Requirement already satisfied: paramiko>=2.6.0 in c:\users\antti\virtu-
alenv\lib\site-packages (from netmiko) (2.7.2)
Collecting textfsm<2.0.0,>=1.1.0
  Downloading textfsm-1.1.0-py2.py3-none-any.whl (37 kB)
Requirement already satisfied: six>=1.9.0 in c:\users\antti\virtu-
alenv\lib\site-packages (from tenacity->netmiko) (1.15.0)
Requirement already satisfied: cryptography>=2.5 in c:\users\antti\virtu-
alenv\lib\site-packages (from paramiko>=2.6.0->netmiko) (3.2.1)
Requirement already satisfied: pynacl>=1.0.1 in c:\users\antti\virtu-
alenv\lib\site-packages (from paramiko>=2.6.0->netmiko) (1.4.0)
Requirement already satisfied: bcrypt>=3.1.3 in c:\users\antti\virtu-
alenv\lib\site-packages (from paramiko>=2.6.0->netmiko) (3.2.0)
Collecting future
  Downloading future-0.18.2.tar.gz (829 kB)
Requirement already satisfied: cffi!=1.11.3,>=1.8 in c:\users\antti\virtu-
alenv\lib\site-packages (from cryptography>=2.5->paramiko>=2.6.0->netmiko)
(1.14.4)
Requirement already satisfied: pycparser in c:\users\antti\virtu-
alenv\lib\site-packages (from cffi!=1.11.3,>=1.8->cryptography>=2.5->para-
miko>=2.6.0->netmiko) (2.20)
Using legacy 'setup.py install' for future, since package 'wheel' is not in-
stalled.
Installing collected packages: future, textfsm, ntc-templates, pyserial, scp,
tenacity, netmiko
    Running setup.py install for future: started
    Running setup.py install for future: finished with status 'done'
Successfully installed future-0.18.2 netmiko-3.3.3 ntc-templates-2.0.0 pyse-
rial-3.5 scp-0.13.3 tenacity-7.0.0 textfsm-1.1.0
```

Esimerkkikoodi 7. Käyttämällä komentoja pip install ncclient ja pip install netmiko käyttäjän ollessa virtuaaliympäristössä saa asennettua ncclient-kirjaston virtuaaliympäristöön.

Ohjelmalla on toiminto, joka hakee konfiguraatiotietoa verkkolaitteelta ja palauttaa sen käyttäjälle XML-tietomuodossa. XML-tietorakenne on suunnattu koneiden ja ohjelmien käsiteltäväksi. Jotta tuo tietomuoto saadaan ihmisläheisemmäksi, tuodaan virtuaaliympäristöön myös kirjastot xmltodict ja xml.dom.minidom. Nämä kirjastot helpottavat XML-muodossa olevan tiedon käsittelyä ja nopeuttavat ohjelman luontia ja testausta huomattavasti.

```
(virtualEnv)
User@DESKTOP-X MINGW64 ~/virtualEnv
$ pip install xmltodict
Collecting xmltodict
  Using cached xmltodict-0.12.0-py2.py3-none-any.whl (9.2 kB)
Installing collected packages: xmltodict
Successfully installed xmltodict-0.12.0
```

Esimerkkikoodi 8. Asennetaan xmltodict-kirjasto virtuaaliympäristöön.

4.2 Ohjelman kirjoittaminen

Kirjastojen asennusten jälkeen voidaan aloittaa ohjelman kirjoittaminen. Lisätään aluksi kirjastot ohjelmakoodin alkuun. Kirjastojen lisääminen ohjelmakoodin alkuun selkeyttää ohjelmakoodia. Muiden tarkastellessa ohjelmakoodia se helpottaa näkemään, mitä Python-kirjastoja ohjelma käyttää.

```
from ncclient import manager
import xmltodict
import xml.dom.minidom
import csv
from netmiko import ConnectHandler
import json
import sys
```

Esimerkkikoodi 9. Lisätään Python-kirjastot ohjelmakoodin alkuun.

4.2.1 Sovelluksen toiminnan valinta

Ohjelma on luotu näyttämään konfiguraatiotietoja, muokkaamaan porteissa olevia konfiguraatioita sekä luomaan uusia silmukkakytkentöjä verkkolaitteelle. Ohjelman käynnistyessä käyttäjältä kysytään, mitä ohjelmantoimintoa käyttäjä haluaa käyttää. Tämän valinnan perusteella ohjelma kysyy eri ohjelmakoodiin kirjoitettujen toimintojen avulla käyttäjältä, mitä tietoa halutaan muuttaa tai tulostaa.

4.2.2 Verkkolaitteen osoitteen antaminen

Ohjelmalla on tarkoitus pystyä hakemaan tietoa ja konfiguroimaan useita eri laitteita. Tähän tarkoitukseen Python-kirjasto csv on oiva työkalu. Sovellukseen tehdään funktio, joka kysyy käyttäjältä, minkä verkkolaitteen kanssa käyttäjä haluaa työskennellä. Tähän annetaan kaksi vaihtoehtoa: manuaalisesti syötetty verkkolaitteen osoite ja csv-tiedostosta haettu verkkolaitteen osoite.

Ohjelma kysyy ensin käyttäjältä, miten käyttäjä haluaa antaa verkkolaitteen osoitteen, manuaalisesti vai csv-tiedoston kautta. Valinnan jälkeen, riippuen kumman tyylin käyttäjä valitsee, voi sovelluksen käyttäjä kirjoittaa verkkolaitteen osoitteen itse tai csv-tiedoston nimen tai tiedostopolun csv-tiedostoon. Valittaessa csv-tiedostosta haettu verkkolaitteenosoite luodaan lista objekti, jonka jälkeen hyödynnetään csv Python -kirjaston opencsv-toimintoa, jolla avataan csv-tiedosto ja tulostetaan käyttäjälle valittavissa olevat verkkolaitteiden osoitteet. Verkkolaitteen osoite palautetaan tämän jälkeen paikkaan ohjelmassa, jossa toimintoa on kutsuttu.

```
def device_select():
    choice = input("How do you want to import device information?\n(1)
CSV\n(2) Manual input\n")
    if choice == '1':
        csvFilename = input("Write CSV filename\n")
        data = []
        opencsv(csvFilename, data)

        for row, k in enumerate(data):
            print(row, data[row]['Device'])
        x = len(data)-1
        print("Select the device: 0-"+ str(x))
        deviceName = int(input())
        device = data[deviceName]['Device']
        return device
```

```
    elif choice == '2':
        device = input("Give hostname/address:\n")
        return device
    else:
        print("Exiting.")

def opencsv(csvFilename, data):
    #data = []
    with open(csvFilename, encoding="utf-8", newline='\n') as f:
        reader2 = csv.DictReader(f, delimiter=';')
        try:
            for row in reader2:
                data.append(row)
                #print(row)
        except csv.Error as e:
            sys.exit('file {}, line {}: {}'.format(csvFilename,
reader2.line_num, e))
        return data
```

Esimerkkikoodi 10. Toiminnot, jolla verkkolaitteen osoite saadaan pyydettyä käyttäjältä manuaalisesti tai haettua csv muodossa olevasta tiedostosta.

4.2.3 Silmukkakytkennän luominen ohjelmalla

Käyttäjän valitessa silmukkakytkennän luomisen ohjelmalla ohjelma pyytää ensimmäisenä käyttäjältä verkkolaiteen osoitetta, johon silmukkakytkentä luodaan. Tämän jälkeen ohjelma kysyy silmukkakytkennän tietoja, kuten kytkentäportin numeroa, kuvausta, IP-osoitetta ja verkkomaskia, jotka portille annetaan. Ohjelma käy tekemässä silmukkakytkennän verkkolaitteelle ja palauttaa tulosteena luodun silmukkakytkennän konfiguraatiotiedot.

```
def loopback_info():

    loopback_number = "Loopback"+input("Give loopback number: ")
    loopback_description = input("Give loopback description: ")
    loopback_ip = input("Give loopback IP: ")
    loopback_subnet = input("Give loopback subnet: ")

    loopback_config = [
    "interface {}".format(loopback_number),
    "description {}".format(loopback_description),
    "ip address {} {}".format(loopback_ip, loopback_subnet),
    "no shut"
    ]
    return loopback_config
```

Esimerkkikoodi 11. Funktio, jolla kysytään silmukkakytkennän tiedot ja tallennetaan ne Pythonin lista objektiin.

```
def loopback_creator():
    with ConnectHandler(ip = device_select(),
    port=22,
```

```
    username="developer",
    password="C1sco12345",
    device_type="cisco_ios") as ch:
        loopback_config = loopback_info()
        ch.send_config_set(loopback_config)
        complete_loopback = ch.send_command('show run {}'.format(loopback_con-
fig[0]), use_textfsm=True)
        print("Created interface: \n" + complete_loopback)
```

Esimerkkikoodi 12. Toiminto yhdistää ohjelman verkkolaitteeseen ja kirjoittaa verkkolaitteelle käyttäjän antamat silmukkakytkennän tiedot. Tämän jälkeen toiminto palauttaa luodun silmukkakytkennän tiedot tulosteena laitteelta.

```
$ py Script.py
What would you like to do?
(1)Create loopback interface with netmiko
(2)Edit existing interface
(3)Show interface configuration
1
How do you want to import device information?
(1) CSV
(2) Manual input
1
Write CSV filename
testi.csv
0 ios-xe-mgmt.cisco.com
1 192.168.255.1
2 192.168.255.2
3 192.168.255.3
4 192.168.255.4
5 192.168.255.5
6 10.10.20.48
Select the device: 0-6
6
Give loopback number: 170
Give loopback description: Test loopback interface 170
Give loopback IP: 10.170.0.170
Give loopback subnet: 255.255.255.255
Created interface:
Building configuration...

Current configuration : 111 bytes
!
interface Loopback170
 description Test loopback interface 170
 ip address 10.170.0.170 255.255.255.255
end
```

Kuva 5. Ohjelman tulosteet ja käyttäjän syötteet, kun ohjelmalla luodaan silmukkakytkentä.

4.2.4 Verkkolaitteen portin muokkaaminen ohjelmalla

Verkkolaitteen porttien muokkaus ohjelmalla hyödyntää molempia Python-kirjastoja, ncclientia ja netmikoa. Näiden kirjastoiden avulla ohjelma ja verkkolaite kykenevät jaka-

maan tietoa. Käyttäjän valitessa portin konfiguraatiotiedon muokkaamisen ohjelmalla kysyy ohjelma, mitä tietoa portista halutaan muokata. Vaihtoehdot portin muokkaamiselle ohjelmalla on portin tila, portinkuvaus tai portin IP-osoitetieto. Riippuen käyttäjän valinnasta käyttää ohjelma netmiko-kirjastosta tuotua connecthandler-toimintoa tai ncclient-kirjastosta tuotua manager-toimintoa.

4.2.5 Verkkolaittee portin tilan muokkaaminen ohjelmalla

Verkkolaitteen portin tilaa muokattaessa ohjelma kysyy alustavat tiedot käyttäjältä, mitä käyttäjä haluaa tehdä, mitä tietoa käyttäjä haluaa muokata portista ja mistä laitteesta tietoa halutaan muokata. Ohjelma hakee verkkolaitteen valinnan jälkeen kaikki mahdolliset portit verkkolaitteesta, joita ohjelmalla on mahdollista muokata. Tämän jälkeen ohjelma tulostaa käyttäjälle uudelleen valinnan, josta käyttäjän on valittava muokattava portti. Verkkolaitteen portin tilasta huolimatta ohjelma tulostaa käyttäjälle valinnan, halutaanko portti asettaa päälle vai pois päältä. Valinnan jälkeen ohjelma varmistaa vielä, mihin laitteeseen muutos tehdään kysymällä laitteen osoitetta uudelleen. Ohjelma tulostaa lopuksi varmistusviestin, onnistuuko portin tilan muokkaus vai ei.

Verkkolaitteen portin tilaa muokattaessa ohjelma käyttää manager-toimintoa, jolla yhdistetään ohjelma verkkolaitteeseen. Ohjelma asettaa käyttäjän valitseman portin tilan laitteelle pelkistämällä xml-konfiguraatiotietoa pelkästään portintilaan vaikuttavaksi.

```
def change_interface_status(interface_choice, status):
    m = manager.connect(host=device_select(),port=830,username="devel-
oper",password="C1sco12345",hostkey_verify=False)


    config = '''
        <config>
            <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
                <interface>
                    <name>{}</name>
                    <enabled>{}</enabled>
                </interface>
            </interfaces>
        </config>
        '''.format(interface_choice, status)

    config_dict = xmltodict.parse(config)

    if status == "true":
        config_dict["config"]["interfaces"]["interface"]["enabled"] = "true"
        config = xmltodict.unparse(config_dict)
    elif status == "false":
```

```
        config_dict["config"]["interfaces"]["interface"]["enabled"] = "false"
        config = xmltodict.unparse(config_dict)

config_dict = xmltodict.parse(config)
netconf_reply = m.edit_config(target='running', config=config)
print("Task completed, {}".format(netconf_reply.ok))
```

Esimerkkikoodi 13. Funktio, joka yhdistää verkko laitteeseen ja muokkaa verkkolaitteen portin tilaa.

```
$ py Script.py
What would you like to do?
(1)Create loopback interface with netmiko
(2)Edit existing interface
(3)Show interface configuration
2
What do want to edit?
(1)Status
(2)Description
(3)IP address
1
How do you want to import device information?
(1) CSV
(2) Manual input
1
Write CSV filename
testi.csv
0 ios-xe-mgmt.cisco.com
1 192.168.255.1
2 192.168.255.2
3 192.168.255.3
4 192.168.255.4
5 192.168.255.5
6 10.10.20.48
Select the device: 0-6
6
1 GigabitEthernet1
2 GigabitEthernet2
3 GigabitEthernet3
Which interface you want to work with? 1-3
3
Select interface new state
(1) Up
(2) Down
1
How do you want to import device information?
(1) CSV
(2) Manual input
2
Give hostname/address:
10.10.20.48
Task completed, True
```

Kuva 6. Ohjelman tulosteet ja käyttäjän syötteet, kun ohjelmaa käytetään verkkolaitteenportin tilan muokkaamiseen.

4.2.6 Verkkolaitteen portin kuvauksen muokkaaminen ohjelmalla

Verkkolaitteen portin kuvausta muokatessa ohjelma toimii hyvin samankaltaisesti kuin muokatessa portin tilaa. Ohjelma käyttää samaa manager toimintoa yhdistääkseen ohjelman käyttäjän valitsemaan verkkolaitteeseen. Myös verkkolaitteen konfiguraatiotiedon syöttäminen laitteelle tapahtuu suodattamalla oleellisin osa konfiguraatiosta, eli portinkuvaus XML-pohjaisesta konfiguraatiotiedosta.

```
def change_interface_description(interface_choice):
    m = manager.connect(host=device_select(),port=830,username="devel-
oper",password="C1sco12345",hostkey_verify=False)
    new_desc = input("Please write what you want as a description:\n")

    config = '''
        <config>
            <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
                <interface>
                    <name>{}</name>
                    <description></description>
                </interface>
                </interfaces>
        </config>
      '''.format(interface_choice)
    config_dict = xmltodict.parse(config)

    config_dict["config"]["interfaces"]["interface"]["description"] = new_desc
    config = xmltodict.unparse(config_dict)

    netconf_reply = m.edit_config(target='running', config=config)
    print("Task completed, {}".format(netconf_reply.ok))
```

Esimerkkikoodi 14. Toiminto, jolla muokataan konfiguraatiotietoa verkkolaitteelta, mikä vie käyttäjän syöttämän uuden portinkuvauksen verkkolaitteelle.

4.2.7 Verkkolaitteen portin IP-osoitetiedon muokkaaminen ohjelmalla

Verkkolaitteen portin IP-osoitetietoa muokatessa ohjelmalla käyttää ohjelma netmiko Python -kirjastosta tuotua connecthandler-toimintoa. Tämän toiminnon avulla pystytään lähettämään komentoja verkkolaitteelle ja tulostamaan osia verkkolaitteen konfiguraatiosta.

Ohjelma kysyy, mistä portista IP-osoitetietoa halutaan vaihtaa, ja pyytää käyttäjää antamaan uuden IP-osoitetiedon ja verkkomaskin syötteenä. Ohjelma varmistaa tämän jälkeen, mihin laitteeseen tiedot annetaan, kysymällä verkkolaitteen osoitetta uudelleen, jonka jälkeen ohjelma tulostaa verkkolaitteenportin vanhan IP-osoitetiedon ja uuden IP-osoitetiedon.

```
$ py Script.py
What would you like to do?
(1)Create loopback interface with netmiko
(2)Edit existing interface
(3)Show interface configuration
2
What do want to edit?
(1)Status
(2)Description
(3)IP address
3
How do you want to import device information?
(1) CSV
(2) Manual input
2
Give hostname/address:
10.10.20.48
1 GigabitEthernet1
2 GigabitEthernet2
3 GigabitEthernet3
4 Loopback150
Which interface you want to work with? 1-4
4
New IP address:
10.200.200.150
New subnet:
255.255.255.255
How do you want to import device information?
(1) CSV
(2) Manual input
2
Give hostname/address:
10.10.20.48
Old interface configuration:

Building configuration...

Current configuration : 99 bytes
!
interface Loopback150
 description Test loopback
 ip address 10.149.149.100 255.255.255.255
end

New interface configuration:
Building configuration...

Current configuration : 99 bytes
!
interface Loopback150
 description Test loopback
 ip address 10.200.200.150 255.255.255.255
end
```

Kuva 7. Ohjelman tulosteet ja käyttäjän syötteet, kun ohjelmaa käytetään verkkolaitteen portin IP-osoitetietojen muokkaamiseen.

Toiminto, joka ohjelmaan on kirjoitettu muokkaamaan verkkolaitteen portin IP-osoitetietoja, kysyy käyttäjältä uudet IP-osoitetiedot, tallentaa nuo käyttäjän syöttämät tiedot listaobjektiksi Pythonissa, jossa listan arvoja ovat merkkijonot, jotka puolestaan toimivat komentoina verkkolaitteelle. Tämä lista ajetaan netmiko-kirjaston connecthandler-toiminnolla verkkolaitteelle. Samalla muokaten verkkolaitteen konfiguraatiotietoa ja toimintaa. Toiminto myös tulostaa portin vanhan ja uuden IP-osoitetiedon käyttäjälle nähtäväksi.

```
def edit_interface_IPaddr(interface_choice):
    new_IPaddr = input("New IP address: \n")
    new_Subnet = input("New subnet: \n")

    try:
        with ConnectHandler(ip = device_select(),
        port=22,username="developer",
        password="C1sco12345",
        device_type="cisco_ios") as ch:

            interfaces = ch.send_command('show run int {}'.format(inter-
face_choice), use_textfsm=True)
            print("Old interface configuration: \n" + interfaces)

            interface_ip_config = [
            "interface {}".format(interface_choice),
            "ip address {} {}".format(new_IPaddr, new_Subnet)
            ]

            ch.send_config_set(interface_ip_config)

            interfaces = ch.send_command('show run int {}'.format(inter-
face_choice), use_textfsm=True)
            print("New interface configuration: \n" + interfaces)
            return

    except Exception as e:
        print("Unable to connect. Exiting.\n"+e)
```

Esimerkkikoodi 15. Käyttäjän valitseman portin IP-osoitetietoa muokkaava toiminto.

4.2.8 Verkkolaitteen konfiguraatiotiedon tulostaminen ohjelmalla

Ohjelmalla on myös mahdollista tulostaa kaikkien verkkolaitteenporttien tietoja käyttäjälle luettavaksi. Tämä ominaisuus helpottaa verkkolaitteen vianselvityksessä ja verkkoympäristön kokonaisuuden hahmottamisessa. Käyttäjän valitessa porttitietojen tarkastuksen ohjelmalla, ohjelma kysyy käyttäjältä, missä muodossa näitä tietoja halutaan tarkastella. Vaihtoehtoina porttitietojen tulostamiseen ovat json- ja xml-muoto.

Käyttäjän valitessa porttitietojen tarkastelun XML-muodossa ohjelma käyttää porttitietojen tulostamiseen NETCONF-ominaisuutta ja suodattaa haetun konfiguraatiotiedon pelkästään verkkolaitteen porttitietoon. XML on hyvin vaikealukuista, mikäli tuota tietoa ei suodateta tai tulosteta porrastetusti käyttäjän nähtäväksi. Tästä syystä ohjelmaan on tuotu xml.dom.minidom-Python-kirjasto, jonka avulla tuota haettua XML-konfiguraatiotietoa on helppo jäsentää käyttäjälle helpommin luettavaan muotoon.

```
def get_interfaces(m):
    netconf_filter = '''
    <filter>
        <interfaces>
            <interface></interface>
        </interfaces>
    </filter>
    '''

    netconf_reply = m.get_config(source='running',filter=netconf_filter)
    netconf_data = xmltodict.parse(netconf_reply.xml)

    netconf_data = xml.dom.minidom.parseString(netconf_reply.xml).topret-
tyxml()
    print(netconf_data)
    return
```

Esimerkkikoodi 16. Ohjelman toiminto, jolla haetaan verkkolaitteelta kaikkien porttientiedot XML-muotoon ja tulostetaan käyttäjälle.

Kuva 8. Osa ohjelman tulosteista käyttäjän valitessa verkkolaitteen porttitietojen tulostamisen XML-muodossa.

Ohjelmalla on myös mahdollista tulostaa porttitietoja JSON-tietomuodossa. Käyttäjän valitessa JSON-tiedostomuodossa porttitiedon tulostamiseen ohjelma käy syöttämässä verkkolaitteelle komennon, jonka tuloste palautetaan ja syötetään json-Python-kirjastossa olevalle json.dumps-toiminnolle. Tuo toiminto muuntaa saadun JSON-muodossa olevan tiedon Pythonin listaobjektiksi.

```
print(json.dumps(json_intbrief(), indent=4))
```

Esimerkkikoodi 17. Rivi koodissa, jossa kutsutaan toimintoa hakemaan tietoa verkkolaitteenporteista ja muutetaan toiminnosta saatu tieto Python-listaobjektiksi.

```
def json_intbrief():
    try:
        with ConnectHandler(ip = device_select(),port=22,username="devel-
oper",password="C1sco12345",device_type="cisco_ios") as ch:
            interfaces = ch.send_command('show ip int brief',
use_textfsm=True)
            return interfaces
    except Exception as e:
        print("Unable to connect.\n"+e)
```

Esimerkkikoodi 18. Toiminto, jolla haetaan verkkolaitteelta porttitieto.

```
$ py Script.py
What would you like to do?
(1)Create loopback interface with netmiko
(2)Edit existing interface
(3)Show interface configuration
3
How would you like to see interface data?
(1)XML format
(2)JSON format
2
How do you want to import device information?
(1) CSV
(2) Manual input
2
Give hostname/address:
10.10.20.48
[
    {
        "intf": "GigabitEthernet1",
        "ipaddr": "10.10.20.48",
        "status": "up",
        "proto": "up"
    },
    {
        "intf": "GigabitEthernet2",
        "ipaddr": "unassigned",
        "status": "administratively down",
        "proto": "down"
    },
    {
        "intf": "GigabitEthernet3",
        "ipaddr": "unassigned",
        "status": "down",
        "proto": "down"
    },
    {
        "intf": "Loopback150",
        "ipaddr": "10.200.200.150",
        "status": "up",
        "proto": "up"
    },
    {
        "intf": "Loopback500",
        "ipaddr": "10.201.200.150",
        "status": "administratively down",
        "proto": "down"
    },
    {
        "intf": "Loopback999",
        "ipaddr": "10.202.200.150",
        "status": "up",
        "proto": "up"
    }
]
```

Kuva 9. Ohjelman tuloste, kun ohjelmaa käytetään verkkolaitteenporttien tiedon hakemiseen JSON-muodossa.

5 Yhteenveto

Tutkin insinöörityössäni eri menetelmiä, joilla verkkolaitteen automatisointia parantaa ohjelmallisesti, ja pyrin luomaan opittujen menetelmien pohjalta ohjelman, millä on mahdollista tehdä muutoksia verkkolaitteiden konfiguraatioon.

Insinöörityön tavoitteena oli luoda ohjelma, jonka avulla pystyy muokkaamaan verkkolaitteen konfiguraatiotietoa. Ohjelman toteutus onnistui, ja tuloksena ohjelmalla pystyy muokkaamaan käyttäjän haluamaa konfiguraatiotietoa verkkolaitteessa. Ohjelman toteutusympäristöihin tutustuminen työn aikana mahdollistaa jatkokehityksen ohjelmalle ja sen testaukselle.

Jatkokehittäessäni ohjelmaa tulee ottaa huomioon ohjelman sopeutuva skaalautuvuus suurempaan verkkolaiteympäristöön sekä eri laitevalmistajien verkkolaitteisiin mukautuva konfiguraatiotiedon lähetys. Nykyistä ohjelmaa voidaan käyttää pohjana jatkossa kehitettäville ohjelmille, joiden tarkoituksena on viedä konfiguraatiotietoa usealle verkkolaitteelle samanaikaisesti.

Ohjelmaa kehittäessä huomattiin, kuinka monimutkaista uuden teknologian jalkautus jo hyvin monimuotoiseen ympäristöön todellisuudessa tulee olemaan. Teorian oppiminen johti helpompaan ohjelman tuottamiseen ja luontevampaan kehittämiseen.

Lähteet

1 Understanding automation, Redhat. Verkkoaineisto. https://www.redhat.com/en/topics/automation. Luettu 20.8.2021.

2 What's IT automation?, Redhat. Verkkoaineisto. https://www.redhat.com/en/topics/automation/whats-it-automation. Luettu 20.8.2021.

3 What is Network Automation, AppviewX. Verkkoaineisto. https://www.appviewx.com/education-center/education-network-automation/what-is-network-automation/. Luettu 21.8.2021

4 Juniper Networks. Verkkoaineisto. Juniper Networks. https://www.juniper.net/uk/en/products-services/what-is/network-automation/. Luettu 22.8.2021.

5 Daniel Tencer, 14.7.2017 Verkkoaineisto. Huffingtonpost. https://www.huffingtonpost.ca/2017/07/14/85-of-jobs-that-will-exist-in-2030-haven-t-been-invented-yet-d_a_23030098/?guccounter=1&guce_referrer=aHR0cHM6Ly9. Luettu 22.8.2021.

6 The Future of Network Engineers, 6.9.2018 Verkkoaineisto. Thinqtank Learning. https://thinqtanklearning.com/blog/the-future-of-network-engineers/. Luettu 22.8.2021.

7 Lori MacVittie, 2.3.2015 Network Engineers: Don't Fear The Code Verkkoaineisto. Networkcomputing. https://www.networkcomputing.com/careers-and-certifications/network-engineers-dont-fear-code. Luettu 29.8.2021.

8 Andrew Foehlich, 11.12.2017 APIs and Networking: 3 Use Cases Verkkoaineisto. Networkcomputing. https://www.networkcomputing.com/networking/apis-and-networking-3-use-cases. Luettu 29.8.2021.

9 Radford, Adam; Maccioni, Fabrizio; Zapodeanu, Gabriel; Koren, Itai; McLaughlin, Jeff; Cohoe, Jeremy; Yang, Jonathan; Kotha, Krishna; Michraf, Nabil; Grasby, Robert IOS XE Programmability – Automating Device Lifecycle Management. Toukokuu 2018. s 31. Luettu 5.9.2021.

10 Jesus Vigo 23.7.2020 Top 5 programming languages for network admins to learn Verkkoaineisto. Techrepublic. https://www.techrepublic.com/article/top-5-programming-languages-for-network-admins-to-learn/. Luettu 5.9.2021.

11 Radford, Adam; Maccioni, Fabrizio; Zapodeanu, Gabriel; Koren, Itai; McLaughlin, Jeff; Cohoe, Jeremy; Yang, Jonathan; Kotha, Krishna; Michraf, Nabil; Grasby, Robert Toukokuu 2018. IOS XE Programmability – Automating Device Lifecycle Management. s 13-15. Luettu 5.9.2021.

12 Tiobe Index for September 2020. Verkkoaineisto. Tiobe https://www.tiobe.com/tiobe-index/ Luettu 5.9.2021.

13 Why Network Engineers Need to Learn Python? 8.4.2018 Verkkoaineisto. Testclue. https://www.testclue.com/network-engineers-need-learn-python/ Luettu 6.9.2021.

14 Jackson, Chris; Gooley, Jason; Iliesiu, Adrian; Malegaonkar, Ashutosh Lokakuu 2020. Cisco Certified DevNet Associate DEVASC 200-901 Official Cert Guide s 58-61. Luettu 5.9.2021.

15 https://developer.cisco.com/video/net-prog-basics/01-programming_fundamentals/python_part_2 14.12.2017. Video. Python Part 2: Working with Libraries and Virtual Environments. Luettu 6.9.2021.

16 https://developer.cisco.com/video/net-prog-basics/01-programming_fundamentals/python_part_3 14.12.2017. Video. Python Part 3: Useful Python Libraries for Network Engineers. Luettu 6.9.2021.

17 https://developer.cisco.com/video/net-prog-basics/01-programming_fundamentals/data_formats 14.12.2017. Video. Data Formats: Understanding and using JSON, XML and YAML. Luettu 6.9.2021.

18 Jackson, Chris; Gooley, Jason; Iliesiu, Adrian; Malegaonkar, Ashutosh Lokakuu 2020. Cisco Certified DevNet Associate DEVASC 200-901 Official Cert Guide s 113-115. Luettu 11.9.2021.

19 Marraskuu 2010. Cisco Sysytems. Cisco 4700 Series Application Control Engine Appliance Administration Guide https://www.cisco.com/c/en/us/td/docs/app_ntwk_services/data_center_app_services/ace_appliances/vA4_1_0/configuration/administration/guide/admgd.pdf. Luettu 19.9.2021.

20 Jackson, Chris; Gooley, Jason; Iliesiu, Adrian; Malegaonkar, Ashutosh Lokakuu 2020. Cisco Certified DevNet Associate DEVASC 200-901 Official Cert Guide s 117-119. Luettu 11.9.2021.

21 Kin Lane, 5.11.2020. Intro to APIs: What Is an API? Verkkoaineisto. Postman Blog. https://blog.postman.com/intro-to-apis-what-is-an-api/. Luettu 29.8.2021.

22 Jackson, Chris; Gooley, Jason; Iliesiu, Adrian; Malegaonkar, Ashutosh Lokakuu 2020. Cisco Certified DevNet Associate DEVASC 200-901 Official Cert Guide s 128-130. Luettu 11.9.2021.

23 Jackson, Chris; Gooley, Jason; Iliesiu, Adrian; Malegaonkar, Ashutosh Lokakuu 2020. Cisco Certified DevNet Associate DEVASC 200-901 Official Cert Guide s 132-134. Luettu 11.9.2021.

24 Networkop.co.uk 1.1.2016. Verkkoaineisto. REST API for Network Engineers https://networkop.co.uk/blog/2016/01/01/rest-for-neteng/. Luettu 10.10.2021.

25 Timms, Natalie 20.6.2014. Verkkoaineisto. NETCONF: Introduction To An Emerging Network Standard. Networkcomputing. https://www.networkcomputing.com/networking/netconf-introduction-emerging-networking-standard. Luettu 24.10.2021.

26 NETCONF OVERVIEW. Verkkoaineisto. Tail-f. https://www.tail-f.com/what-is-netconf/. Luettu 25.10.2021.

27 Mendiola, Alaitz; Astorga, Jasone; Jacob, Eduardo; Higuero, Marivi Toukokuu 2017. A survey on the contributions of Software-Defined Networking to Traffic Engineering. https://www.researchgate.net/publication/311338103_A_Survey_on_the_Contributions_of_Software-Defined_Networking_to_Traffic_Engineering. Luettu 25.10.2021.

28 Huawei Techologies. 26.4.2019. Verkkoaineisto. Configuration Guide – Network Management and Monitoring https://support.huawei.com/enterprise/en/doc/EDOC1100004357/13482586/netconf-content-layer. Luettu 31.10.2021.

29 Huawei Techologies. 26.4.2019. Verkkoaineisto. Configuration Guide – Network Management and Monitoring https://support.huawei.com/enterprise/en/doc/EDOC1000166639/83f73817/netconf-operation-layer. Luettu 31.10.2021.

30 Cisco. Verkkoaineisto. https://yang-prog-lab.ciscolive.com/pod/0/mdp_foundations/introduction_to_netconf. Luettu 16.10.2021.

31 Juniper Networks. Verkkoaineisto. Muokattu 6.7.2017. https://www.juniper.net/documentation/en_US/bti-series/bti78004.2/topics/concept/c-7800-swicg-about-netconf.html. Luettu 16.10.2020.

32 Netconf central. Verkkoaineisto. http://www.netconfcentral.org/netconf_docs. Luettu 17.10.2021.

33 Huawei Techologies. 26.4.2019. Verkkoaineisto. Configuration Guide – Network Management and Monitoring https://support.huawei.com/enterprise/en/doc/EDOC1000166639/6df9593b/netconf-transport-layer. Luettu 22.10.2021.

34 Rick Donato 30.11.2017. Verkkoaineisto. An Introduction to NETCONF/YANG Fir3net. https://www.fir3net.com/Networking/Protocols/an-introduction-to-netconf-yang.html. Luettu 7.10.2021.

35 Jackson, Chris; Gooley, Jason; Iliesiu, Adrian; Malegaonkar, Ashutosh Lokakuu 2020. Cisco Certified DevNet Associate DEVASC 200-901 Official Cert Guide s 347-348. Luettu 11.9.2021.

36 Python. Verkkoaineisto. https://docs.python.org/3/tutorial/venv.html. Luettu 23.10.2021.

Ohjelmakoodi

Ohjelmakoodi, jonka työssäni loin. Liite sisältää pelkästään ohjelmakoodin osuuden, joten virtuaaliympäristön ja muu valmistelu ohjelman toimintaa varten on tehtävä erikseen.

```
from ncclient import manager
import xmltodict
import xml.dom.minidom
import sys
import csv
from netmiko import ConnectHandler
import json

def program_function():
    function = int(input("What would you like to do?\n(1)Create loopback in-
terface with netmiko\n(2)Edit existing interface\n(3)Show interface configura-
tion\n"))
    return function

def hostname():
    netconf_reply = get_running_config()
    netconf_data = xmltodict.parse(netconf_reply.xml)
    print("IOS Version: {}".format(netconf_data["rpc-reply"]["data"]["na-
tive"]["version"]))
    print("Hostname: {}".format(netconf_data["rpc-reply"]["data"]["na-
tive"]["hostname"]))

def device_select():
    choice = input("How do you want to import device information?\n(1)
CSV\n(2) Manual input\n")
    if choice == '1':
        csvFilename = input("Write CSV filename\n")
        data = []
        opencsv(csvFilename, data)

        for row, k in enumerate(data):
            print(row, data[row]['Device'])
        x = len(data)-1
        print("Select the device: 0-"+ str(x))
        deviceName = int(input())
        device = data[deviceName]['Device']
        return device

    elif choice == '2':
        device = input("Give hostname/address:\n")
        return device
    else:
        print("Exiting.")

def opencsv(csvFilename, data):
    with open(csvFilename, encoding="utf-8", newline='\n') as f:
        reader2 = csv.DictReader(f, delimiter=';')
        try:
            for row in reader2:
                data.append(row)
        except csv.Error as e:
```

```
            sys.exit('file {}, line {}: {}'.format(csvFilename,
reader2.line_num, e))
        return data


def get_running_config(m):
    netconf_reply = m.get_config(source='running')


def get_interfaces(m):
    netconf_filter = '''
    <filter>
        <interfaces>
            <interface></interface>
        </interfaces>
    </filter>
    '''
    netconf_reply = m.get_config(source='running',filter=netconf_filter)
    netconf_data = xmltodict.parse(netconf_reply.xml)

    netconf_data = xml.dom.minidom.parseString(netconf_reply.xml).topret-
tyxml()
    print(netconf_data)
    return

def loopback_info():
    loopback_number = "Loopback"+input("Give loopback number: ")
    loopback_description = input("Give loopback description: ")
    loopback_ip = input("Give loopback IP: ")
    loopback_subnet = input("Give loopback subnet: ")
    loopback_config = [
    "interface {}".format(loopback_number),
    "description {}".format(loopback_description),
    "ip address {} {}".format(loopback_ip, loopback_subnet),
    "no shut"
    ]
    return loopback_config


def loopback_creator():
    with ConnectHandler(ip = device_select(),
    port=22,
    username="developer",
    password="C1sco12345",
    device_type="cisco_ios") as ch:
        loopback_config = loopback_info()
        ch.send_config_set(loopback_config)
        complete_loopback = ch.send_command('show run {}'.format(loopback_con-
fig[0]), use_textfsm=True)
        print("Created interface: \n" + complete_loopback)
        return

def netconf_edit_choice():
    edit = int(input("What do want to edit?\n(1)Status\n(2)Description\n(3)IP
address\n"))
    return edit

def show_interface_json():
    try:
        with ConnectHandler(ip = device_select(),port=22,username="devel-
oper",password="C1sco12345",device_type="cisco_ios") as ch:
```

```
            interfaces = ch.send_command('show ip int brief',
use_textfsm=True)
            return interfaces

    except Exception as e:
        print("Unable to connect. Exiting.\n"+str(e))

def interface(interfaces):
    i = 0
    for key in interfaces:
        i = i + 1
        print("{} {}".format(i, key['intf']))

    selectInterface = int(input("Which interface you want to work with? 1-
"+str(len(interfaces))+"\n"))
    selectInterface = selectInterface-1
    selectedInterface = interfaces[selectInterface]['intf']
    return selectedInterface

def choose_interface():
    interfaces = show_interface_json()
    selectedInterface = interface(interfaces)
    return selectedInterface

def change_interface_status(interface_choice, status):
    m = manager.connect(host=device_select(),port=830,username="devel-
oper",password="C1sco12345",hostkey_verify=False)
    config = '''
        <config>
            <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
                <interface>
                    <name>{}</name>
                    <enabled>{}</enabled>
                </interface>
            </interfaces>
        </config>
        '''.format(interface_choice, status)

    config_dict = xmltodict.parse(config)

    if status == "true":
        config_dict["config"]["interfaces"]["interface"]["enabled"] = "true"
        config = xmltodict.unparse(config_dict)
    elif status == "false":
        config_dict["config"]["interfaces"]["interface"]["enabled"] = "false"
        config = xmltodict.unparse(config_dict)

    config_dict = xmltodict.parse(config)
    netconf_reply = m.edit_config(target='running', config=config)
    print("Task completed, {}".format(netconf_reply.ok))

def change_interface_description(interface_choice):
    m = manager.connect(host=device_select(),port=830,username="devel-
oper",password="C1sco12345",hostkey_verify=False)
    new_desc = input("Please write what you want as a description:\n")
    config = '''
        <config>
            <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
                <interface>
                    <name>{}</name>
                    <description></description>
                </interface>
```

```
                </interfaces>
            </config>
          '''.format(interface_choice)
    config_dict = xmltodict.parse(config)
    config_dict["config"]["interfaces"]["interface"]["description"] = new_desc
    config = xmltodict.unparse(config_dict)
    netconf_reply = m.edit_config(target='running', config=config)
    print("Task completed, {}".format(netconf_reply.ok))

def json_intbrief():
    try:
        with ConnectHandler(ip = device_select(),port=22,username="devel-
oper",password="C1sco12345",device_type="cisco_ios") as ch:
            interfaces = ch.send_command('show ip int brief',
use_textfsm=True)
            return interfaces

    except Exception as e:
        print("Unable to connect.\n"+e)

def edit_interface_status():
    status = int(input("Select interface new state\n(1) Up\n(2) Down\n"))
    if status == 1:
        status = "true"
        return status
    elif status == 2:
        status = "false"
        return status
    else:
        print("Error")
        sys.exit(main())

def edit_interface_IPaddr(interface_choice):
    new_IPaddr = input("New IP address: \n")
    new_Subnet = input("New subnet: \n")
    try:
        with ConnectHandler(ip = device_select(),
        port=22,username="developer",
        password="C1sco12345",
        device_type="cisco_ios") as ch:

            interfaces = ch.send_command('show run int {}'.format(inter-
face_choice), use_textfsm=True)
            print("Old interface configuration: \n" + interfaces)

            interface_ip_config = [
            "interface {}".format(interface_choice),
            "ip address {} {}".format(new_IPaddr, new_Subnet)
            ]

            ch.send_config_set(interface_ip_config)

            interfaces = ch.send_command('show run int {}'.format(inter-
face_choice), use_textfsm=True)
            print("New interface configuration: \n" + interfaces)
            return

    except Exception as e:
        print("Unable to connect. Exiting.\n"+e)

def main():
    function = program_function()
```

```
    if function == 1:
        loopback_creator()
    elif function == 2:
        #User sub-options // (1)Status (2)Description (3)IP address
        edit = netconf_edit_choice()
        interface_choice = choose_interface()
        #Interface status changer
        if edit == 1:
            status = edit_interface_status()
            change_interface_status(interface_choice, status)
        #Interface description changer
        elif edit == 2:
            change_interface_description(interface_choice)
        #Interface IP address changer
        elif edit == 3:
            edit_interface_IPaddr(interface_choice)
        else:
            print("Unable to connect. Exiting.")
            sys.exit(main())
#Option3
    elif function == 3:
        data_struc = int(input("How would you like to see interface
data?\n(1)XML format\n(2)JSON format\n"))
        if data_struc == 1:
            try:
                with manager.connect(host=device_se-
lect(),port=830,username="developer",password="C1sco12345",hostkey_ver-
ify=False) as m:
                    get_interfaces(m)
            except Exception as e:
                print(e)
        elif data_struc == 2:
            print(json.dumps(json_intbrief(), indent=4))
    else:
        print("Invalid character, try again.")
        sys.exit(main())
if __name__ == '__main__':
    sys.exit(main())
```