Vendor Neutral Interface for Network Device Configuration

D. L. Hewage

(179464K)

Faculty of Information Technology

University of Moratuwa

June 2020

Vendor Neutral Interface for Network Device Configuration

D. L. Hewage

(179464K)

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirement of the Degree of Master of Science in Information Technology

June 2020

Declaration

I declare that this thesis is my own work and this does not incorporate without acknowledgement any material previously published submitted for a Degree or Diploma in any other university or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Student

D. L Hewage

_____ 14/ June/ 2020

Signature of Student Date

Supervised by

Dr. M. F. M. Firdhous

_____ 14/ June/ 2020

Signature of Supervisor Date

Abstract

Efficient network configuration of heterogeneous network devices from different vendors is a great challenge. Networking personnel have to adopt to different vendor specific CLIs when necessary, where they have to go through a learning curve to get adapted to a new product. Having to go through a learning process makes such personnel think twice before recommending a product of a different vendor, even though it offers more value for money. Even if they recommend a different vendor based product with hesitation, the implementation and troubleshooting of a networking solution with a new vendor based product increases the duration and cost of the project due to addition of training time and cost.

This research aims to introduce a simple vendor-neutral interface and command converter interface as a simple and efficient approach to overcome issues faced by networking personnel when managing heterogeneous network devices by different vendors. It also aims to make networking infrastructures vendor neutral in all aspects by contributing specially to network configuration management area.

Acknowledgement

I would like to convey my special thanks to Dr. M.F.M. Firdhous, Senior Lecturer, Faculty of Information Technology University of Moratuwa for the support, guidance and supervision given to me throughout the project, making it a success. My sincere gratitude to Mr. D.K. Withanage, Mr. C. Wijesiriwardana and Mr. S. Premarathne for their kind advices and support given to me.

I'm grateful to my family for their enormous support given to me at various points in time during the course of this research. Further I'm thankful to all my colleagues in both Msc. IT Batch 11 and work for the support rendered to me.

Finally, I thank all the staff members and Faculty members of Faculty of Information Technology University of Moratuwa.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

API – Application Programming Interface

CLI – Command Line Interface

CNRI – Corporation for National Research Initiatives

IANA – Internet Assigned Number Authority

IETF – Internet Engineering Task Force

NETCONF – Network Configuration Protocol

NETMOD – Network Modelling

YANG – Data Modelling Language for the Network Configuration Protocol

POSIX – Portable Operating System Interface

IEEE – Institute of Electronic and Electrical Engineers

SNMP – Simple Network Management Protocol

ISO – International Standard Organization

OSI – Open System Interconnection

LAN – Local Area Network

WLAN – Wireless Local Area Network

SSH – Secure Shell

LLDP – Link Layer Discovery Protocol

LACP – Link Aggregation Control Protocol

MSTP – Multiple Spanning Tree Protocol

VLAN – Virtual LAN

ACL – Access Control List

POC – Proof of Concept

Chapter 1

1.0 Introduction

1.1 Prolegomena

Since the initiative to introduce standards in networking that started in 1977, IETF pioneered to introduce a network configuration management protocol by the name SNMP in 1980s. The said protocol's name itself stands for its purpose Simple Network Management Protocol. Though there were certain issues with respect to security in initial release, they were able to address them in later releases as the protocol evolved over time. Even though SNMP was introduced to be used with both monitoring and configuration management, the industry kept on using device specific CLIs for the configuration management purpose. Further they embraced SNMP only as a monitoring protocol.

Network configuration management field suffers from inability to survive in a heterogeneous environment. One approach to solve this is to introduce a brand new protocol that the industry will embrace to be used with both monitoring and configuration purposes. This method would be effective in the long run, when industry accepts it and legacy equipment not having support for new protocol gets replaced with new ones. Another simple but effective approach that would be more productive in contemporary and future deployments of network configuration management, would be to introduce a simple vendor-neutral interface in conjunction with a command converter API. The said interface should be simple, user-friendly and acceptable to be used with any kind of vendor specific networking device. Further the said API should be capable of handling command conversions from a vendor-neutral CLI to vendor specific CLIs.

Introduction of a vendor-neutral interface and command converter API would be very much beneficial to the network configuration since it would be a solution to most of the hardships that networking personnel face when adopting to products manufactured by different manufacturers. Furthermore it would increase efficiency and reduce costs that has to be incurred in network configuration management of heterogeneous environments.

1.2 Problem Statement

Efficient network configuration of heterogeneous network devices from different vendors is a great challenge. This research aims to introduce a simple vendor-neutral interface and command converter API as a simple and efficient approach to overcome issues faced by networking personnel when managing heterogeneous network devices by different vendors. It also aims to make networking infrastructures vendor neutral in all aspects by contributing specially to the network configuration management area.

1.3 Aim and Objectives

1.3.1 Aim

Introduction of a simple vendor-neutral interface and a command converter API for simplified network configuration.

1.3.2 Objectives

1. Identification of existing CLIs of major network device vendors.
2. Analysis and comparison of different vendor specific CLIs with respect to their syntaxes and features.
3. Proposing a simple vendor-neutral interface to replace vendor specific CLIs.
4. Development and introduction of a command converter API that is capable of handling command conversions tasks mentioned below;
    i. Conversion between vendor specific CLIs.
    ii. Simple vendor-neutral interface to selected vendor specific CLIs.
5. Testing the command converter API with respect to configuration of products by selected major vendors.
6. Composing a user-friendly manual for vendor-neutral interface and command converter API.

1.4 Research Scope

1.Since the scope of considering all major vendors CLIs would be too large, consideration would only be done for three selected vendors namely, Cisco, HPE and Aruba.

2. During this research only switch CLI syntaxes would be considered.

3. Considering switch CLI syntaxes of selected three vendors, still it could be considered as verbose. Therefore following aspects of CLI would not be considered during this research;

i. IPv6 based configurations.

ii. Security configurations of both IPv6 and IPv4.

iii. Support for propitiatory protocols and features.

iv. Password recovery routines.

v. Monitoring aspects using SNMP.

vi. Quality of service configurations.

4. Since the research targets, proof of concept in terms of utilizing simple vendor-neutral interface for network configuration, priority will be given for support of only the key configuration aspects through the proposed CLI and API.

5. Since the development of API would be based on Python it could be considered as a platform independent API, but during deployment if any complications arise with certain OS platforms priority will be given for implementation in Linux based OS platform.

1.5 Structure of the Thesis

The thesis is structured as follows. Chapter 2 consists of literature review, chapter 3 explains the technologies adopted for the research, chapter 4 describes about the analysis and design process of research while chapter 5 demonstrates the implementation process of the research finally the chapter 6 consists of conclusion and future works of the thesis.

Chapter 2

2.0 Literature Review

2.1 Introduction

Chapter 2 review the concepts and techniques used in network configuration management field. Further it discusses the shortcomings in the same. The chapter tries to identify the solutions that can be used to overcome the issues in the discussed areas.

2.2 Literature Review

The networking in the new millennia has come a long way since the International Standard Organization (ISO) identified the urgent need for introduction of standards that would allow for interconnecting systems from different manufacturers, and established a new subcommittee (SC16) for "Open System Interconnection" in 1977 [1].Nowadays almost all networking devices support interoperability to a very high extent using standards and standardized protocols defined by the above mentioned initiatives. Thanks to standardization modern day networks have grown exponentially due to increased interoperability between multi-vendor devices.

Irrespective of the IT infrastructure model used within the organization whether it is cloud computing or local computing based, having a properly managed and well-designed LAN, WLAN infrastructure is very much important for the smooth functioning of its operations. LAN and WLAN infrastructure mainly consists of active networking components like firewalls, layer 3 and layer 2 switches, routers, wireless access points, WLAN controllers and etc. Though most of the above mentioned components support manageability through web based GUI or through a CLI, Network Engineers prefer the use of command line over GUI for configuration management of networking devices due to the reason of CLI being simple and highly efficient.

Active manageable networking components have their own command line syntax depending on the manufacturer. Even though the networking device manufacturers adhere to ISO-OSI standards and standardized protocols, in order to support inter-operability, they maintain their own vendor specific CLIs for configuration of respective

devices. Irrespective of the device model a particular manufacturer's product range have somewhat similar command line syntaxes. When it comes to products from different manufacturers the CLI syntaxes are totally different from one another. Therefore Network Engineers face the challenge of having to get multiple product specific training and certifications to configure and manage products manufactured by different manufacturers. Though the networking products support interoperability, due to the above mentioned reason Network Engineers always prefer to stick to a specific manufacturer's products when it comes to design and implementation of network infrastructure.

Network Engineers being restricted to products manufactured by specific manufacturers increase the implementation costs of network infrastructure, due to the reason of losing the ability to make use of competitive advantage when purchasing products. Furthermore it degrades the quality of network infrastructure due to reduced capability to migrate to new products by other manufacturers having useful state of the art features. Being restricted to vendor specific products and CLI syntaxes increases possibility of using proprietary protocols, further reducing possibility of introducing products manufactured by different manufacturers into the existing infrastructure.

In late 1980s IETF introduced SNMP as a solution to overcome the issue of network monitoring and configuration management not being vendor independent [2]. Initial version of SNMP named SNMPv1 was widely deployed at the time and was superseded by SNMPv2 and the latest version SNMPv3 released in 2002 which addresses security requirements. Though SNMP was developed to implement vendor independent network monitoring and configuration management it is widely being used by the industry for the purpose of fault and performance monitoring. The industry kept on preferring the CLI for configuration management purposes.

In year 2002 during a workshop on Network Management hosted by CNRI, IETF's Architecture Board identified the issue of SNMP not being adopted by the industry for the purpose of network configuration management. Then IETF initiated their work on developing a vendor independent network configuration management protocol by the name NETCONF with the establishment of NETCONF work group [3]. The key reasons

for introduction of a totally new protocol was due to SNMP being good only for monitoring and CLI scripting being problematic, complex and time consuming in terms of utilization in large scale operator based network infrastructures. Later IETF established another work group by the name NETMOD for the development of YANG. In year 2011 IETF introduced NETCONF as a protocol that aids to install, manipulate, and delete the configurations of network devices [4] [5]. In year 2010 IETF introduced YANG as a data modelling language used to model configuration and state data manipulated by the Network Configuration Protocol [6].Since the introduction of NETCONF there has been a rush to embrace the new protocol by major vendors. Among the supporters were Brocade Communications Systems Inc. Cisco Systems Inc., Juniper Networks Inc., Hewlett Packard Enterprise, Ericsson-LG and Nokia Networks, but notably not Alcatel-Lucent [7].

IETF taking the initiative to implement a vendor independent CLI for network configuration management is admirable and very much beneficial to the industry. However there are certain limitations and practical issues pertaining to the utilization of NETCONF and YANG mostly in medium and small scale network infrastructures, namely; legacy network equipment not supporting NETCONF, NETCONF supporting network devices being more expensive, Network Equipment Manufacturers supporting NETCONF and YANG only with the firmware and products targeted for Data-centres and Operators, NETCONF and YANG implementation requiring extensive knowledge on XML, programming languages, scripting languages and System Engineering field, NETCONF and YANG being more suitable for large scale automated deployments and configuration requirements rather than for medium to small scale infrastructure deployments with human intervention. Furthermore it should be understood that migration to NETCONF and YANG supported network infrastructure cannot be achieved by enforcing a flag day for everyone to start using it. This was the same dilemma that IANA faced when introducing IPv6 to replace IPv4.

Rather than introducing a totally new set of protocols like NETCONF and YANG to make network configuration management vendor neutral, a simpler approach could be utilized to overcome the above issue; that is to introduce a simple vendor-neutral interface and a

command converter API capable of converting commands between different vendor specific CLIs and between proposed simple vendor-neutral interface and vendor specific CLIs. Introduction of such API would be advantages to the networking industry, since it would facilitate Network Engineers who are restricted to certain vendor platforms to work on other platforms with zero learning curve, further decreasing certification and training costs. It would also improve the competitive advantage when purchasing network devices by reducing tendency to use preferred brand specific specifications. It indirectly aids in reduction of using propitiatory protocols in network infrastructure.

Even-though the application is totally different IEEE's POSIX stands for Portable Operating System Interface, which was introduced to facilitate application portability, application interoperability, data portability, and user portability at the source code level mostly in Unix/Linux platform could be considered as a similar attempt by a consortium of vendors to create a single standard version of UNIX/Linux based applications [8]. POSIX compliance makes it simpler to port applications between OS Platforms, which results in time savings. So developers got acquainted with the fundamentals of this widely used standard when developing applications for UNIX/Linux based OS platforms. Examples of some POSIX-compliant systems are AIX, HP-UX, Solaris, and MacOS (since 10.5 Leopard). On the other hand, Android, FreeBSD, Linux Distributions, OpenBSD and VMWare follow most of the POSIX standard, but they are not certified [9].

Paramiko is a community maintained, python based API that has been developed under GNU Lesser General Public License (LGPL) to facilitate SSH connectivity to other SSH supporting devices [10]. Though this API could be considered as a good platform to develop a command converter API, there are certain known issues with it that makes it incapable of connecting to most of the network product by major vendors like Cisco and HPE.

Netmiko is another community maintained, Python based API build on top of Paramiko, introduced by Kirk Byers. It contains a multi-vendor library to simplify Paramiko SSH connections to network devices [11]. This particular GitHub fork has regularly tested set

of libraries that makes it capable of connecting to network products by major vendors like Cisco, Juniper and HPE. It also has some more experimental libraries with limited testing that support connectivity to other vendor's products as well. Netmiko could be considered as a suitable Platform for development of a command converter API that would facilitate command conversion between different vendor specific CLIs and between proposed simple vendor-neutral interface and vendor specific CLIs. (See Figure 2.1)

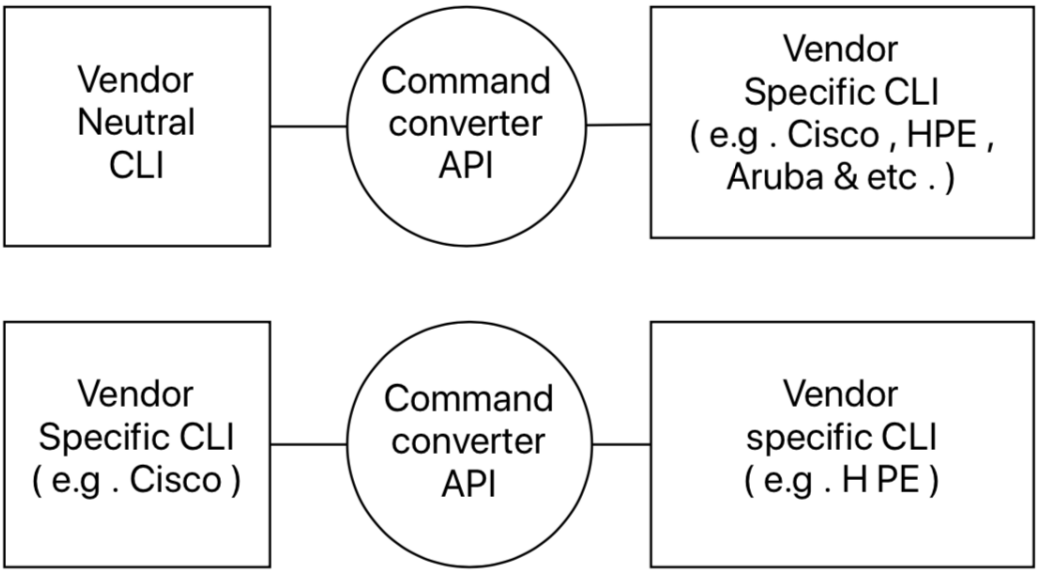

Figure 2.1 High-level Architecture of the APIs

Apart from facilitative APIs for SSH connection establishment, currently there aren't similar APIs that are capable of conducting command conversion between different vendor specific CLIs. The only available aid that helps Network Engineers to configure products from different vendors are some CLI Reference Guides having comparisons of multi-vendor CLI syntaxes [12]. Therefore it could be considered that this type of approach to build a simple vendor-neutral CLI and API would be worth the venture.

2.3 Problem Definition

Difficulty in managing multi-vendor network devices, due to the reason of network configuration CLIs being vendor specific has to be overcome either by using a command converter API or through a vendor neutral CLI suitable for all vendor platforms.

2.4 Summary

This chapter comprehensively review the shortcomings and issues in the network configuration management field. It also identifies the positive solutions that can be used to overcome the discussed issue.

Chapter 3

3.0 Technology Adopted

3.1 Introduction

Chapter 3 describes the process of selecting technologies that were used for the implementation of the solutions mentioned in chapter 2. Further it discusses the reasons for selection of the specific technologies that were adopted.

3.2 Selection of CLI Brands

There are hundreds of networking product vendors available in the world. Some of them are famous and being used all around the world and some of them are famous in certain regions. However method of selecting number of brands to be utilized for the project had to be worked out. Therefore Gardner Magic Quadrant for the Wired and Wireless LAN Access Infrastructure for past three years 2017, 2018 and 2019 were taken into consideration. (See figure 3.1, 3.2 and 3.3 given below)

*Fig*ure 3.1 Gartner Magic Quadrant for the Wired and Wireless LAN Access Infrastructure 2019

Figure 3.2 Gartner Magic Quadrant for the Wired and Wireless LAN Access Infrastructure 2018

Figure 3.3 Gartner Magic Quadrant for the Wired and Wireless LAN Access Infrastructure 2017

After analysing the Gartner Magic Quadrant it was identified that HPE, Aruba and Cisco had been in the Magic Quadrant for past three consecutive years. Therefore HPE, Aruba and Cisco were selected as candidates for the API development.

3.3 CLI Comparison Metric

After selection of three brands for development of command converter APIs, it was decided that the selected configuration commands have to be compared using a comparison metric to identify the CLI similarities. Therefore following formula was developed and applied to compare the CLI commands.

$$C = \sum A_0 + A_1 + A_2 + A_3 + B_0 + B_1 + B_2$$

C = Comparison Value
A0 = 3 if command is exactly the same else A0 = 0
A1 = similar argument count * 2
A2 = count of synonymous or arguments with minor character differences * 1
A3 = 1 if argument count is same else A3 = 0
B0 = -3 if command is different else B0 = 0
B1 = different argument count * -2
B2 = -1 if argument count is different else B2 = 0

C = -15 if the commands are incomparable

Since the HPE Comware 5 and Comware 7 were very much similar the formula was not applied to compare Comware 5 and 7. Instead of that Cisco and HPE Comware7, Cisco and Aruba, HPE Comware7 and Aruba were compared using the above mentioned formula. When compare the cumulative Comparison value for Aruba and HPE was -460 while comparison value between Aruba and Cisco was -141 and comparison value between Cisco and HPE was -246. Therefore it was understood that all three (3) CLIs are have more dissimilar than being similar. Aruba and HPE turned out to be most dissimilar

while Aruba and Cisco turned out to be similar compared with others. The comparison values mentioned above clearly show the importance of developing a command converter API.

3.4 Paramiko

In order to develop a connection sub-system it was decided to utilize an already available open source ssh client API. Therefore Paramiko turned out to be a perfect candidate since it was supporting both Python 2.7 and 3.4+ implementation of the SSHv2 protocol, providing both client and server functionality. It leverages a Python C extension for low level cryptography (Cryptography). [10] However upon testing Paramiko functions for the development purpose it was identified that there were some connectivity problems in terms of connecting to different models of network devices using Paramiko.

3.5 Netmiko

After unsuccessful attempt to develop the vendor neutral API on top of Paramiko it was identified that there was another library specialized in connecting to Network devices. This Library was built on top of Paramiko Library and supported many networking brands available. Further Investigation of the Netmiko documentation identified that the Netmiko Library already supports successful connectivity to the brands selected for the project [11]. Netmiko support many number of networking brands and successful testing of connectivity to selected networking brands made it the most suitable library to be used to make this project a success. Further Netmiko being able to install on Windows, Linux and OSX Operating System (OS) platforms aided the decision to select Netmiko for the API development.

3.6 Python 3

Since it was already decided to make use of Python based Netmiko library for the development of the connection sub-system, priority was given to a programing language that is compatible with Netmiko library. Further it was identified that even though Netmiko supported both Python 2 and 3, Python 2 was being discontinued with effect from 1st Jan 2020 [13]. Therefore automatically Python 3 turned out to be most suitable

candidate for development. Furthermore Python being a loosely typed language having lots of text manipulation functions made it even more suitable for the purpose developing a CLI based command converter subsystem.

Development of the API was done making use of free community edition of Sublimetext IDE. The prototype testing was done on a Centos 7 based VM having installed with Python 3, Paramiko and Netmiko libraries. Since Centos 7 native installation of python was Python 2, Python 3 had to be installed manually.

3.5 Summary

This chapter presented justifications for selection process of specific technologies that were used to in chapter 5. Namely the Python 3, Paramiko based Netmiko libraries were used for the implementation as per the justifications given here.

Chapter 4

4.0 Analysis and Design

4.1 Introduction

Chapter 4 describe the analysis and design process conducted prior to the implementation process discussed in chapter 5. The chapter describes the functional requirements, Non-functional requirements, and development methodology used in chapter 5.

4.2 Functional Requirements

1. API should be capable of connecting to selected set networking devices from selected brands using ssh protocol.
2. API should be capable of converting selected set of CLI commands from a particular brand to another brand's CLI commands.
3. API should have an ability to show the outputs generated by networking device as a response to the converted command sent to it.
4. API should have an ability identify and display error messages if the input command is not supported by the networking device being managed.
5. API should support for seamless conversion of different modes or states of configurations between the CLIs of different brands of networking devices.
6. API should be capable of gracefully terminating the ssh connection with the networking device being configured.

4.3 Non Functional Requirements

1. API should have capable of being scalable to support command conversion to other brands of networking devices
2. API should be maintainable in terms of making modifications to commands if a specific vendor makes a modification to an existing command with a new release of firmware. Further it should support adding new commands as required.

3. API should be user-friendly whereas the end user should feel like doing configuration changes to the networking device user is familiar with, though configuration would be done on a different device having a different CLI.

4.4 User Characteristics

1. User should be familiar with networking concepts.
2. User should have a basic knowledge and familiarity with configuration of networking devices using a specific brands CLI.
3. User should have basic computer skills with respect to using a terminal program.

4.5 Architectural Design

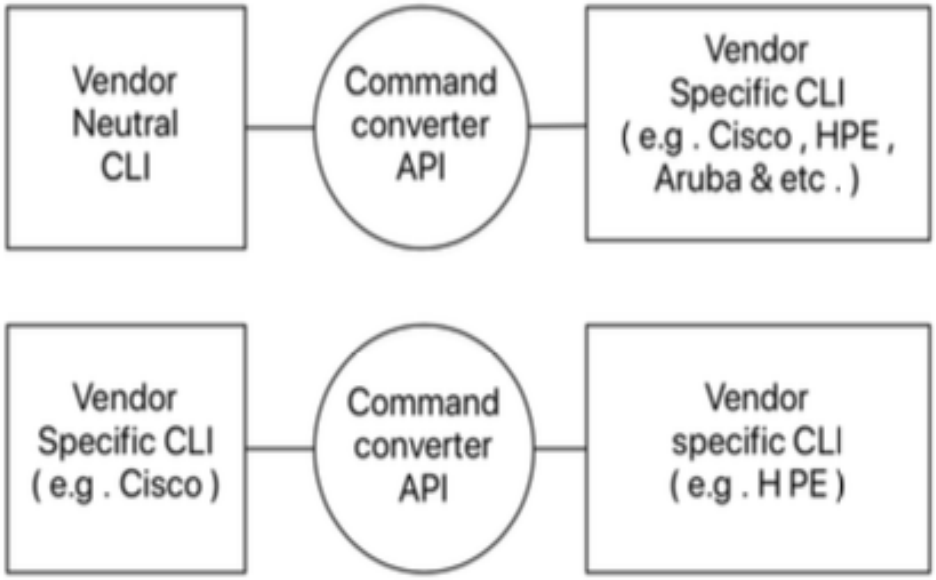

Figure 4.1 Architectural Design

4.6 Use case Diagram

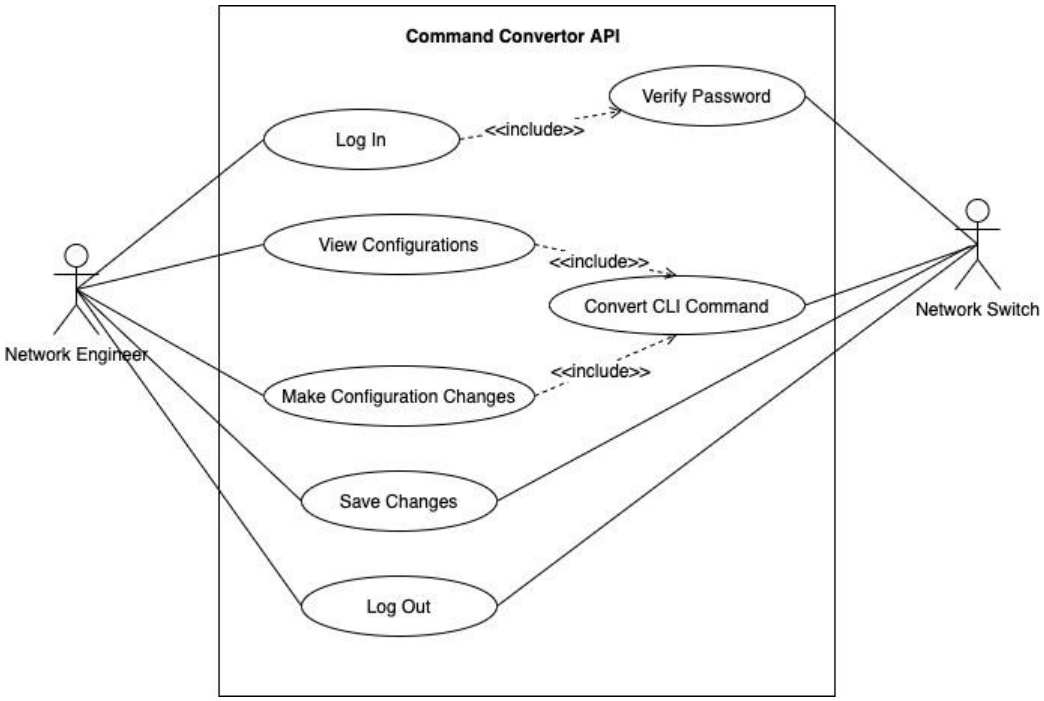

Figure 4.2 Use Case Diagram

4.7 Software Development Life Cycle

4.7.1 Software Prototyping

Software prototyping can be considered as a working model of software with some limited functionality. The prototype does not always hold the exact logic used in the actual software application prototyping could also be used to allow the users evaluate developer proposals and try them out before implementation. It also helps to understand the requirements which are user specific and may not have been considered by the developer during product design stage [14].

Since the Software Prototyping tuned out to be the most suitable candidate for the design requirement of vendor neutral and vendor specific API for CLI based network configuration it was decided to make use of the Software Prototyping as the development of the project. Therefore Following stepwise approach was used in the development process;

1. Basic Requirement Identification
2. Developing the initial Prototype
3. Review of the Prototype
4. Revise and Enhance the Prototype

There are different types of software prototyping techniques used in the industry namely;

1. Throwaway Prototyping
2. Evolutionary Prototyping
3. Incremental Prototyping
4. Extreme Prototyping

After analysing different types of prototyping it was identified that most suitable option for this specific requirement would be Incremental Prototyping.

4.7.2 Incremental Prototyping

Incremental prototyping method was used where multiple functional prototypes were built for the connection sub-system and command converter sub-system. Then the final prototypes were integrated to form a complete API.

4.8 Summary

This chapter presented design process involved prior to the initiation of the implementation stage. It descriptively describes software development lifecycle used in the design process. It also identifies functional and non-functional requirements that were identified documented beforehand.

Chapter 5

5.0 Implementation

5.1 Introduction

Chapter 5 descriptively explains the implementation process of the solutions identified in chapter 1. The chapter consists of 4 stages of development that were done according to the SDLC method identified in Chapter 4. As per the selected SDLC method incremental prototyping, the development process was conducted incrementally by initial development of connection sub-system, command conversion sub-system and finally the combining of the two sub systems. Most importantly this chapter tabulates selected vendor based CLIs in comparison, which aided the design and testing and implementation process directly.

5.2 Implementation of Netmiko connection sub-system

Netmiko API supports ssh connectivity to following brands of networking products with full compatibility Arista EOS, Cisco ASA, Cisco IOS/IOS-XE, Cisco IOS-XR, Cisco NX-OS, Cisco SG300, HP Comware7, HP, ProCurve, Juniper Junos and are being regularly tested [15].

Further the same API supports following brands of networking products Alcatel AOS6/AOS8, Apresia, Systems AEOS, Calix B6, Cisco AireOS (Wireless LAN Controllers), Dell OS9 (Force10), Dell OS10, Dell PowerConnect, Extreme ERS (Avaya), Extreme VSP (Avaya), Extreme VDX (Brocade), Extreme MLX/NetIron (Brocade/Foundry), Huawei, IP Infusion OcNOS, Mellanox, NetApp cDOT, OneAccess, Palo Alto PAN-OS, Pluribus, Ruckus ICX/FastIron, Ubiquiti, EdgeSwitch, Vyatta VyOS with limited testing [15].

Netmiko also supports connectivity with the following brands A10, Accedian, Aruba, Ciena SAOS, Citrix Netscaler, Cisco Telepresence, Check Point GAiA, Coriant, Dell OS6, Dell EMC Isilon, Eltex, Enterasys, Extreme EXOS, Extreme Wing, Extreme SLX (Brocade), F5 TMSH, F5 Linux, Fortinet, MRV Communications OptiSwitch, Nokia/Alcatel SR-OS, QuantaMesh, Rad ETX experimentally [15].

Since the Netmiko API supports the connectivity to HP, Cisco, Aruba and a very large number of networking brands it was selected as the most suitable candidate to be used as a platform to build the required connection sub-system. Therefore during the testing stage of initial prototype to connect with HP, Aruba and Cisco devices it was identified that the netmiko.hp.hp_comware module residing in /opt/rh/rh-python36/root/usr/lib/python3.6/site-packages/netmiko/hp path did not support configuration of HPE switches having the following models numbers namely, HPE 1910, 1920, 1950 but the module supported configuration of HPE 5500 and 5510 successfully. After investigation of above issue it was identified that configuration problem occurred due to certain HPE switches by default only supporting management through the web interface and full range of commands being disabled by default. In order to enable CLI in those switches certain command had to be entered in to the terminal followed by a password specific to the switch model.

Therefore initial prototype of Netmiko based connection sub-system was developed using send_command_timing() method. See Appendix A for code. But it turned out to be partially successful due to the reason of connection sub-system not being able to disable paging in the switch CLI. The problem with not being able to disable paging would be when a show command would require display of multiple pages only the first page would be visible to the end user. This was due to netmiko.hp.hp_comware module not having privileges to disable paging during connection initiation.

In order to overcome above issues, three (3) new modules were developed with slight modifications to original netmiko.hp.hp_comware module. See Appendix B for code. The newly modified modules were placed inside the same path as original module. Further the __init__.py and ssh_dispatcher.py were also modified to support the newly added modules. See Appendix C for code. Newly developed Netmiko based connection sub-system turned out to be successful in terms of connecting and configuring all HPE Comware based switches having both Comware7 and Comware5 firmware.

Though HPE connection and configuration issues were solved by creating new Netmiko modules, similarly it was identified that Netmiko. Aruba module residing in /opt/rh/rh-python36/root/usr/lib/python3.6/site-packages/netmiko/arub path also had problems connecting to specific Aruba switches namely Aruba 2530 switch. After a quite a lot of efforts to correct the problem, it was identified that netmiko.hp.hp_procurve module residing in /opt/rh/rh-python36/root/usr/lib/python3.6/site-packages/netmiko/hp path works successfully in terms of connecting and configuring Aruba switches. The reason for the Netmiko. Aruba module not working was due to the reason of the module being developed for connectivity to Aruba WLAN controller firmware rather than for switch firmware.

However during the development of Netmiko connection sub-system there were no issue in connecting to Cisco IOS based devices using the netmiko.cisco.cisco_ios module residing in /opt/rh/rh-python36/root/usr/lib/python3.6/site-packages/netmiko/cisco path.

The above mentioned modified hp_comware, hp_comware512900, hp_comwarejinhua, hp_comwarefoes, hp_procurve and cisco_ios modules were used to develop initial connection sub-system. The developed sub-system was capable of connecting and configuring selected vendor specific switch brands through vendor specific CLIs using ssh protocol.

5.3 Implementation of vendor specific command converter sub-system

During the process of designing scope of the project it was identified that development of an API to support command conversion for higher number of brands would be infeasible. Therefore it was decided that an API should be designed to only support for three brands. Even with the selection of three brands due to the reason of selected brands having very large number of networking devices ranging from layer 1 to layer 7 in ISO-OSI model, it was decided to provide support for only Switched that operate in Layer 2 and 3. In order to further reduce the scope to comply with the timeline of the project, it was decided that the command conversion should support only IPv4 based most important configuration routines that are frequently used by Network Engineers.

Since the scope of the project specifies that the API development would only consider command conversion for selected IPv4 configurations of three selected brands of switches, the development of command converter sub-system only considered the following configuration routines for the development.

1. Transitioning between modes
2. Save configurations
3. View configurations
4. SSH configurations
5. Link Layer Discovery Protocol (LLDP) configurations
6. Port management
7. VLAN management
    i. Data
    ii. Voice
8. VLAN assignment
    i. Trunk
    ii. Access
    iii. Voice
9. VLAN interface IP Address assignment
10. Default route configuration
11. Link Aggregation Control Protocol (LACP) configurations
12. Multiple Spanning Tree Protocol (MSTP) configurations
    i. MSTP configurations
    ii. MSTP hardening
13. Access Control Lists (ACLs)
    i. ACL creation
        a) Standard numbered ACLs
        b) Standard named ACLs
        c) Extended numbered ACLs
        d) Extended named ACLs
    ii. Routed ACL assignment
        a) Standard numbered ACLs

b) Standard named ACLs

c) Extended numbered ACLs

d) Extended named ACLs

iii. Port ACL assignment

a) Standard numbered ACLs

b) Standard named ACLs

c) Extended numbered ACLs

d) Extended named ACLs

Separate command conversion sub-systems were developed to handle conversions between Cisco and other brands, Aruba and other brands, HPE Comware5 and other Brands and HPE Comware7 and other brands. Even though the project scope specifies command converter sub-systems for only three selected brands development process had to consider development of four sub-systems due to HPE having three versions of firmware namely Comware5, Comware7 and Procurve. Procurve which not considered in order to reduce development effort. It was decided that considering the above mentioned sub-systems would be enough for Proof of Concept (PoC).

During development process of above mentioned four (4) subsystems, command sequences of above mentioned thirteen (13) configuration routines were compared in tabular format to identify similarities and most suitable conversion process. Further during development process following tables were used to develop incremental prototypes of command conversion sub-systems for selected brands.

Initial vendor specific command converter subsystem was developed using Python 3 and it was decided that commands that needs conversion to be stored in Lists inside multidimensional Dictionaries. The reason for storage of commands within the script itself was due to the face that there would not be any simultaneous access of commands by end users. The initial prototypes were developed to get the command that needs to be converted, through standard input (stdin) and to provide the converted command as an output to standard output (stdout). See Appendix D for code.

5.3.1 Transitioning Between Modes

As indicated in following table both Aruba and Cisco switches have similar initial modes once logged using ssh (User-Mode). But HPE switches have a different mode having similar capability (User-View).

Table 5.1 Activating Enable Mode or System View

Brand	**Mode**	**CLI Command**
Aruba	User-Mode	enable
HPE Comware5	User-View	system-view
HPE Comware7	User-View	system-view
Cisco	User-Mode	enable

Aruba and Cisco have another mode called Privileged-Mode/Enable-Mode having higher level of privileges, HPE has a different mode called System-View again having similar features.

The said modes can be reached using the commands given in the table below. There is a significant difference between Aruba, Cisco and HPE where HPE only has two modes of operation User-View and System-View but Aruba and Cisco have three modes namely User-Mode, Privilege-Mode and Config-Mode.

Table 5.2 Activating Configure Mode or System View

Brand	**Mode**	**CLI Command**
Aruba	Privileged-Mode	configure
HPE Comware5	System-View	system-view

HPE Comware7	System-View	system-view
Cisco	Privileged-Mode/ Enable-Mode	configure terminal

In Cisco and Aruba the user can exit a certain mode by using exit command whereas the same can be done in HPE using quit command.

Table 5.3 Exit or Quit a Mode of operation

Brand	**Mode**	**CLI Command**
Aruba	Any	exit
HPE Comware5	Any	quit
HPE Comware7	Any	quit
Cisco	Any	exit

Virtual Terminal Configuration Mode

Aruba switches have no support for this mode of operation. Cisco and HPE Comware 5 and Comware7 supports this mode with different numbers of vty connection ranges.

Table 5.4 Switch to Virtual Terminal Interface Configuration

Brand	**Mode**	**CLI Command**
Aruba	N. A.	Command not supported
HPE Comware5	System-View	user-interface vty 0 15
HPE Comware7	System-View	user-interface vty 0 15
Cisco	Config-Mode	line vty 0 15

Interface Configuration Mode

All three brands have support for this mode but they have their own distinct way of representing individual physical ports of the switch.

Table 5.5 Switch to Interface Configuration mode

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	interface 1
HPE Comware5	System-View	interface g1/0/1
HPE Comware7	System-View	interface g1/0/1
Cisco	Config-Mode	interface g0/1

Interface Range Configuration Mode

Only HPE Comware7 and Cisco brands have support for this quite useful mode of operation where multiple physical ports can be managed simultaneously.

Table 5.6 Switch to Interface Range Configuration mode

Brand	Mode	CLI Command
Aruba	Config-Mode	Command not supported
HPE Comware5	System-View	Command not supported
HPE Comware7	System-View	interface range g1/0/1 to g1/0/4
Cisco	Config-Mode	interface range g0/1 - 4

VLAN Configuration Mode

All three brands of switches support this mode of operation having the exact same.

Table 5.7 Switch to VLAN Configuration Mode

Brand	Mode	CLI Command
Aruba	Config-Mode	vlan 200
HPE Comware5	System-View	vlan 200
HPE Comware7	System-View	vlan 200
Cisco	Config-Mode	vlan 200

VLAN Interface Configuration Mode

Though HPE and Cisco consist of this mode of operation Aruba has a different mode to do the same configuration.

Table 5.8 Switch to VLAN Interface Configuration mode

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	vlan 300
HPE Comware5	System-View	interface Vlan-interface 300
HPE Comware7	System-View	interface Vlan-interface 300
Cisco	Config-Mode	interface vlan 300

Link Aggregation Configuration Mode

Though LACP configuration is supported in Aruba there is no specific mode to do the configuration instead it's done in Config mode. Therefore the following table indicates that command nut supported under Aruba.

Table 5.9 Switch to Link Aggregation Configuration Mode

Brand	Mode	CLI Command
Aruba	Config-Mode	Command not supported
HPE Comware5	System-View	interface Bridge-Aggregation 1
HPE Comware7	System-View	interface Bridge-Aggregation 1
Cisco	Config-Mode	interface port-channel 1

MSTP Region Configuration Mode

Though MSTP configuration is supported in Aruba there is no particular mode to do the configuration instead it's done in Config mode. Therefore following table indicates that command nut supported under Aruba.

Table 5.10 Switch to MSTP Configuration mode

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	Command not supported
HPE Comware5	System-View	stp region-configuration
HPE Comware7	System-View	stp region-configuration
Cisco	Config-Mode	spanning-tree mst configuration

Standard Numbered ACL Configuration Mode

Aruba and Cisco have exact same command for this operation whilst HPE consist of different command having totally different ACL number ranges.

Table 5.11 Switch to Standard Numbered ACL Configuration Interface

Brand	Mode	CLI Command
Aruba	Config-Mode	ip access-list standard 50
HPE Comware5	System-View	acl number 2500
HPE Comware7	System-View	acl number 2500
Cisco	Config-Mode	ip access-list standard 50

Standard Named ACL Configuration Mode

Aruba and Cisco have exact same command for this operation whilst HPE consist of different command having totally different ACL number ranges.

Table 5.12 Switch to Standard Named ACL Configuration Interface

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	ip access-list standard std_acl
HPE Comware5	System-View	acl number 2600 name std_acl
HPE Comware7	System-View	acl number 2600 name std_acl
Cisco	Config-Mode	ip access-list standard std_acl

Extended Numbered ACL Configuration Mode

Aruba and Cisco have exact same command for this operation whilst HPE consist of different command having totally different ACL number ranges.

Table 5.13 Switch to Extended Numbered ACL Configuration Interface

Brand	Mode	CLI Command
Aruba	Config-Mode	ip access-list extended 150
HPE Comware5	System-View	acl number 3500
HPE Comware7	System-View	acl number 3500
Cisco	Config-Mode	ip access-list extended 150

Extended Named ACL Configuration Mode

Aruba and Cisco have exact same command for this operation whilst HPE consist of different command having totally different ACL number ranges.

Table 5.14 Switch to Extended Named ACL Configuration Interface

Brand	Mode	CLI Command
Aruba	Config-Mode	ip access-list extended ext_acl
HPE Comware5	System-View	acl number 3600 name ext_acl
HPE Comware7	System-View	acl number 3600 name ext_acl
Cisco	Config-Mode	ip access-list extended ext_acl

5.3.2 Save Configurations

Configuration changes made can be stored in the HPE, Cisco and Aruba switches using distinct commands given below.

Table 5.15 Save the Configurations

Brand	**Mode**	**CLI Command**
Aruba	Privileged-Mode	save
HPE Comware5	System-View	save force
HPE Comware7	System-View	save force
Cisco	Privileged-Mode	copy run start

5.3.3 View Configurations

Following table represents the commands of selected brands that can be used to view configurations of respective switches.

Table 5.16 View configuration details

Aruba	**HPE Comware5**	**HPE Comware7**	**Cisco**
Privileged-Mode	System-View	System-View	Privileged-Mode
View general configuration details			
show flash	dir	dir	show flash
show version	display version	display version	show version
show system information	display device manuinfo	display device manuinfo	show inventory
show modules	display device verbose	display device verbose	show version
show run	display current-configuration	display current-configuration	show run

show config	display saved-configuration	display saved-configuration	show start
show history	display history	display history	show history
show logging	display info-center	display info-center	show logging
show ip route	display ip routing-table	display ip routing-table	show ip route
show ip	display ip interface brief	display ip interface brief	show ip interface brief
show tech	display diagnostic-information	display diagnostic-information	show tech-support
show system fans	display fan	display fan	show env fan
show system power-supply	display power	display power	show env power
show system temperature	display enviorenment	display enviorenment	show env temperature
View ssh configuration details			
show telnet	display users	display users	show users
show ip ssh	Command not supported	display ssh server status	show ip ssh
Command not supported	Command not supported	display ssh server session	show ssh
show crypto host-public-key	Command not supported	display public-key local rsa public	show crypto key mypubkey rsa
View LLDP details			
show lldp info remote-device	display lldp neighbor-information list	display lldp neighbor-information list	show lldp neighbors

Command not supported	display lldp neighbor-information brief	Command not supported	Command not supported
show lldp info remote-device 1	display lldp neighbor-information interface g1/0/1	display lldp neighbor-information interface g1/0/1	show lldp neighbors g0/1 detail
View interface details			
show interfaces brief	display interface brief	display interface brief	show interfaces status
show interfaces brief 1	display interface g1/0/1 brief	display interface g1/0/1 brief	show interfaces g0/1 status
show interfaces 1	display interface g1/0/1	display interface g1/0/1	show interfaces g0/1
View VLAN details			
show vlans	display vlan	display vlan	show vlan brief
Command not supported	display vlan all	display vlan all	Command not supported
show vlans 200	display vlan 200	display vlan 200	Command not supported
View LACP details			
show lacp	display link-aggregation summary	display link-aggregation summary	show etherchannel summary
show lacp peer	display link-aggregation member-port	display link-aggregation member-port	show interface etherchannel
View STP details			
show spanning-tree	display stp	display stp	show spanning-tree

show spanning-tree mst-config	display stp region-configuration	display stp region-configuration	show spanning-tree mst configuration
show spanning-tree instance ist	display stp instance 0	display stp instance 0	show spanning-tree mst 0
show spanning-tree instance 1	display stp instance 1	display stp instance 1	show spanning-tree mst 1
View STP hardening details			
show link-keepalive	display dldp	display dldp	show udld g0/1
show link-keepalive statistics	display dldp statistics	display dldp statistics	Command not supported
View Command details			
show	display	display	show

5.3.4 SSH Configurations

Though all three brands support for ssh configuration Aruba configuration routine has limited set of commands having limited set of features. HPE and Cisco has somewhat similar routines having their own distinct commands to do the same set of operations.

Table 5.17 SSH Configurations

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	hostname ArubaSW
	Config-Mode	Command not supported
	Config-Mode	crypto key generate ssh
	Config-Mode	ip ssh
	N. A.	Command not supported
	N. A.	Command not supported
	N. A.	Command not supported

HPE Comware5	System-View	sysname Hpe5SW
	System-View	domain domain.com
	System-View	public-key local create rsa
	System-View	ssh server enable
	System-View	user-interface vty 0 15
	VTYS	authentication-mode [none | password | scheme]
	VTYS	protocol inbound [all | ssh | telnet]
HPE Comware7	System-View	sysname Hpe7SW
	System-View	domain domain.com
	System-View	public-key local create rsa
	System-View	ssh server enable
	System-View	user-interface vty 0 15
	VTYS	authentication-mode [none | password | scheme]
	VTYS	protocol inbound [all | ssh | telnet]
Cisco	Config-Mode	hostname CiscoSW
	Config-Mode	ip domain-name domain.com
	Config-Mode	crypto key generate
	Config-Mode	ip ssh version 2
	Config-Mode	line vty 0 15
	VTYC	login [local | tacacs]
	VTYC	transport input [all | ssh | telnet]

5.3.5 Link Layer Discovery Protocol (LLDP) Configurations

Aruba and Cisco has similar commands to enable LLDP globally while HPE Comware 5 and 7 has the ability to enable LLDP both globally and locally to individual ports. Since the project scope specifies that the proprietary protocols would not be considered the Cisco Discovery Protocol (CDP) commands were not considered.

Table 5.18 Activate LLDP

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	lldp run
	N. A.	Command not supported
	N. A.	Command not supported
HPE Comware5	System-View	lldp enable
	System-View	interface g1/0/1
	IS	lldp enable
HPE Comware7	System-View	lldp global enable
	System-View	interface g1/0/1
	IS	lldp enable
Cisco	Config-Mode	lldp run
	N. A.	Command not supported
	N. A.	Command not supported

5.3.6 Port Management

Other than the fact that Aruba switches have a single command to configure both duplex and speed there seems to be similar routines between other brands except for slight changes in the syntaxes.

Table 5.19 Port Management Configurations

Brand	Mode	CLI Command
Aruba	Config-Mode	interface 1
	IC	name link-to-core
	IC	speed-duplex [10-half | 100-half | 10-full | 100-full |1000-full | auto]
	IC	speed-duplex [10-half | 100-half | 10-full | 100-full |1000-full | auto]
	IC	disable
	IC	enable
HPE Comware5	System-View	interface g1/0/1
	IS	description link-to-core
	IS	duplex [auto | full | half]
	IS	speed [auto | 10 | 100 | 1000]
	IS	shutdown
	IS	undo shutdown
HPE Comware7	System-View	interface g1/0/1
	IS	description link-to-core
	IS	duplex [auto | full | half]
	IS	speed [auto | 10 | 100 | 1000]
	IS	shutdown
	IS	undo shutdown
Cisco	Config-Mode	interface g0/1
	IC	description link-to-core

	IC	duplex [auto | full | half]
	IC	speed [auto | 10 | 100 | 1000]
	IC	shutdown
	IC	no shutdown

5.3.7 VLAN Management

Data

VLAN management configuration in terms of creating data vlans seems to be having the exact same syntax with respect to all selected brands.

Table 5.20 Creating and Naming a Data VLAN

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	vlan 800
	VIC	name test
HPE Comware5	System-View	vlan 800
	VIS	name test
HPE Comware7	System-View	vlan 800
	VIS	name test
Cisco	Config-Mode	vlan 800
	VIC	name test

Voice

VLAN management configuration in terms of creating voice vlans seems to be having the exact same syntax with respect to all selected brands except for Aruba.

Table 5.21 Creating and Naming a Voice VLAN

Brand	Mode	CLI Command
Aruba	Config-Mode	vlan 500
	VIC	voice
HPE Comware5	System-View	vlan 500
	VIS	name voice
HPE Comware7	System-View	vlan 500
	VIS	name voice
Cisco	Config-Mode	vlan 500
	VIC	name voice

5.3.8 VLAN Assignment

Trunks

How trunk ports configured in Aruba switches have a distinct difference from how other brands perform the operation. Therefore conversion of this routine from Aruba commands to other brands turned out to be infeasible.

Table 5.22 Configuration of a Trunk Port

Brand	Mode	CLI Command
Aruba	Config-Mode	vlan 200
	VIC	tagged 2
	Config-Mode	vlan 100
	VIC	tagged 2
	Config-Mode	vlan 240
	VIC	tagged 2

HPE Comware5	System-View	interface g1/0/2
	IS	port link-type trunk
	IS	port trunk permit vlan 200 100 240
	System-View	interface g1/0/2
	IS	port link-type trunk
	IS	port trunk permit vlan all
HPE Comware7	System-View	interface g1/0/2
	IS	port link-type trunk
	IS	port trunk permit vlan 200 100 240
	System-View	interface g1/0/2
	IS	port link-type trunk
	IS	port trunk permit vlan all
	System-View	interface range g1/0/1 to g1/0/2
	IRS	port link-type trunk
	IRS	port trunk permit vlan all
Cisco	Config-Mode	interface g0/2
	IC	switchport trunk encapsulation [negotiate | isl | dot1q]
	IC	switchport trunk allowed vlan 200,100,240
	Config-Mode	interface g0/2
	IC	switchport trunk encapsulation [negotiate | isl | dot1q]
	IC	switchport trunk allowed vlan all
	Config-Mode	interface range g0/1 - 2

	IRC	switchport trunk encapsulation [negotiate | isl | dot1q]
	IRC	switchport trunk allowed vlan all

Access

Again how access ports configured in Aruba switches have a distinct difference from how other brands perform the operation. Therefore conversion of this routine from Aruba commands to other brands turned out to be infeasible.

Table 5.23 Configuration of an Access Port

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	vlan 400
	VIC	untagged 4
HPE Comware5	System-View	interface g1/0/4
	IS	port link-type access
	IS	port access vlan 400
HPE Comware7	System-View	interface g1/0/4
	IS	port link-type access
	IS	port access vlan 400
	System-View	interface range g1/0/4 to g1/0/8
	IRS	port link-type access
	IRS	port access vlan 400
Cisco	Config-Mode	interface g0/4
	IC	switchport mode access
	IC	switchport access vlan 400

	Config-Mode	interface range g0/4 - 8
	IRC	switchport mode access
	IRC	switchport access vlan 400

Voice

Again how hybrid ports configured in Aruba switches have a distinct difference from how other brands perform the operation. Therefore conversion of this routine from Aruba commands to other brands turned out to be infeasible.

Table 5.24 Configuration of a Voice Port

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	vlan 500
	VIC	tagged 10
	Config-Mode	vlan 800
	VIC	untagged 10
HPE Comware5	System-View	interface g1/0/10
	IS	port link-type hybrid
	IS	port hybrid vlan 500 tagged
	IS	port hybrid vlan 800 untagged
HPE Comware7	System-View	interface g1/0/10
	IS	port link-type hybrid
	IS	port hybrid vlan 500 tagged
	IS	port hybrid vlan 800 untagged
	System-View	interface range g1/0/10 to g1/0/15
	IRS	port link-type hybrid

	IRS	port hybrid vlan 500 tagged
	IRS	port hybrid vlan 800 untagged
Cisco	Config-Mode	interface g0/10
	IC	switchport mode access
	IC	switchport voice vlan 500
	IC	switchport access vlan 800
	Config-Mode	interface range g0/10 - 15
	IRC	switchport mode access
	IRC	switchport voice vlan 500
	IRC	switchport access vlan 800

5.3.9 VLAN Interface IP Address Assignment

Assignment of ip addresses to VLAN interfaces can be performed using the exact same commands in all three (3) brands selected.

Table 5.25 IP Address Assignment for a VLAN Interface

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	vlan 100
	VIC	ip address 192.168.1.1 255.255.255.0
HPE Comware5	System-View	vlan 100
	VVIS	ip address 192.168.1.1 255.255.255.0
HPE Comware7	System-View	vlan 100
	VVIS	ip address 192.168.1.1 255.255.255.0
Cisco	Config-Mode	vlan 100

	VVIC	ip address 192.168.1.1 255.255.255.0

5.3.10 Default Route Configuration

Assignment of ip addresses to VLAN interfaces can be performed in a similar manner in all three (3) brands selected with only a slight change of syntax in HPE.

Table 5.26 Default Route Configuration

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	ip route 0.0.0.0 0.0.0.0 192.168.1.1
HPE Comware5	System-View	ip route-static 0.0.0.0 0.0.0.0 192.168.1.1
HPE Comware7	System-View	ip route-static 0.0.0.0 0.0.0.0 192.168.1.1
Cisco	Config-Mode	ip route 0.0.0.0 0.0.0.0 192.168.1.1

5.3.11 Link Aggregation Control Protocol (LACP) Configurations

How LACP configured in Aruba switches have a distinct difference from how other brands perform the operation. Therefore conversion of this routine from Aruba commands to other brands turned out to be infeasible. However same operation routine in other brands could be done using set of commands unique to respective brands. Assignment of ports to LACP groups could be performed quite easily in Cisco and HPE Comware 7 since they support for simultaneous configuration of multiple ports.

Table 5.27 LACP Configurations

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	trunk 20-23 trk1 lacp
	Config-Mode	vlan 200 tagged trk1
	Config-Mode	vlan 100 tagged trk1

	Config-Mode	vlan 240 tagged trk1
HPE Comware5	System-View	interface Bridge-Aggregation 1
	BAIS	link-aggregation mode [dynamic | static]
	BAIS	port link-type trunk
	BAIS	port trunk permit vlan 200 100 240
	System-View	interface g1/0/23
	IS	port link-aggregation group 1
	System-View	interface g1/0/24
	IS	port link-aggregation group 1
HPE Comware7	System-View	interface Bridge-Aggregation 1
	BAIS	link-aggregation mode [dynamic | static]
	BAIS	port link-type trunk
	BAIS	port trunk permit vlan 200 100 240
	System-View	interface range g1/0/23 to g1/0/24
	IRS	port link-aggregation group 1
Cisco	Config-Mode	interface port-channel 1
	N.A.	Command not supported
	BAIC	switchport trunk encapsulation [negotiate | isl | dot1q]
	Config-Mode	interface range g0/23 - 24
	IRC	channel-group 1 mode [active | passive | on | desirable]

5.3.12 Multiple Spanning Tree Protocol (MSTP) Configurations

MSTP Configurations

MSTP configuration was included in this project scope despite availability of other STP protocols like RSTP, PVSTP and RPVSTP due to the reason of almost all features available under said protocols being available in MSTP with reduced hassle in configuration.

MSTP configuration routines of all brands seems to be quite similar except for the syntaxes. However the mode of operation with respect to regional/mst configurations in terms of Aruba could be considered different from other brands.

Table 5.28 MSTP Configurations

Brand	Mode	CLI Command
Aruba	Config-Mode	spanning-tree
	N.A.	Command not supported
	Config-Mode	spanning-tree config-name SWMSTP
	Config-Mode	spanning-tree config-revision 1
	Config-Mode	spanning-tree instance 1 vlan 200 100 240
	Config-Mode	spanning-tree instance 2 vlan 500 800
	N.A.	Command not supported
	Config-Mode	spanning-tree priority 1
	Config-Mode	spanning-tree instance 1 priority 2
	Config-Mode	spanning-tree instance 2 priority 3
	Config-Mode	spanning-tree pathcost mstp [8021d | 8021t | proprietary]
	N.A.	Command not supported

	Config-Mode	spanning-tree 20 path-cost 20000
	Config-Mode	spanning-tree 20 priority 4
	Config-Mode	spanning-tree instance 1 20 path-cost 20000
	Config-Mode	spanning-tree instance 1 20 priority 4
HPE Comware5	System-View	stp enable
	System-View	stp region-configuration
	STPRIS	region-name SWMSTP
	STPRIS	revision-level 1
	STPRIS	instance 1 vlan 200 100 240
	STPRIS	instance 2 vlan 500 800
	STPRIS	active region-configuration
	System-View	stp priority 4096
	System-View	stp instance 1 priority 8192
	System-View	stp instance 2 priority 12288
	System-View	stp pathcost-standard [dot1d-1998 | dot1t | legacy]
	System-View	interface g1/0/20
	IS	stp cost 20000
	IS	stp port priority 16394
	IS	stp instance 1 cost 20000
	IS	stp instance 1 port priority 16384
HPE Comware7	System-View	stp enable
	System-View	stp region-configuration
	STPRIS	region-name SWMSTP

	STPRIS	revision-level 1
	STPRIS	instance 1 vlan 200 100 240
	STPRIS	instance 2 vlan 500 800
	STPRIS	active region-configuration
	System-View	stp priority 4096
	System-View	stp instance 1 priority 8192
	System-View	stp instance 2 priority 12288
	System-View	stp pathcost-standard [dot1d-1998 | dot1t | legacy]
	System-View	interface g1/0/20
	IS	stp cost 20000
	IS	stp port priority 16394
	IS	stp instance 1 cost 20000
	IS	stp instance 1 port priority 16384
Cisco	Config-Mode	spanning-tree mode mst
	Config-Mode	spanning-tree mst configuration
	STPRIC	name SWMSTP
	STPRIC	revision 1
	STPRIC	instance 1 vlan 200,100,240
	STPRIC	instance 2 vlan 500,800
	N.A.	Command not supported
	Config-Mode	spanning-tree mst 0 priority 4096
	Config-Mode	spanning-tree mst 1 priority 8192
	Config-Mode	spanning-tree mst 2 priority 12288
	Config-Mode	spanning-tree pathcost method [long | short]

	IC	interface g0/20
	IC	spanning-tree cost 20000
	IC	spanning-tree port-priority 16394
	IC	spanning-tree mst 1 cost 20000
	IC	spanning-tree mst 1 port-priority 16384

MSTP Hardening

Data Link Discovery Protocol (DLDP)

This hardening option when enabled monitors a link between two switches and blocks the port on both ends of the link if the link fails at any point between the two devices. This option is quite useful for detecting failures in fibre links. Except for syntactic differences, how this option is enabled in switches that belongs to all three brands are similar.

Table 5.29 DLDP Configurations

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	interface 1
	IC	link-keepalive
HPE Comware5	System-View	interface g1/0/1
	IS	dldp enable
HPE Comware7	System-View	interface g1/0/1
	IS	dldp enable
Cisco	Config-Mode	interface g0/1
	IC	udld port

Edge-port

This option ones enabled allow for immediate transition from blocking to forwarding, normally enabled on access ports that are not connected to switches. Except for syntactic differences, this option can be enabled in switches that belongs to all three brands in a similar manner. However there is a slight change in mode of doing the configuration in Aruba unlike other switch brands where the configuration would be done in Config-Mode rather than in respective interface configuration mode.

Table 5.30 Edge Port Configurations

Brand	Mode	CLI Command
Aruba	N.A.	Command not supported
	C	spanning-tree 2 admin-edge-port
HPE Comware5	System-View	interface g1/0/2
	IS	stp edge-port enable
HPE Comware7	System-View	interface g1/0/2
	IS	stp edge-port
Cisco	Config-Mode	interface g0/2
	IC	spanning-tree portfast

Bridge Protocol Data Unit (BPDU) Guard

BPDU guard is a security feature designed to protect the active MSTP topology by preventing spoofed BPDU packets from entering the MSTP domain. In a typical implementation, it would be advised to apply BPDU guard to edge ports connected to end devices that do not run MSTP. If BPDU packets are received on a BPDU guard enabled port, this feature would disable that port.

Except for syntactic differences, this option can be enabled in switches that belongs to all three brands in a similar manner. However there is a slight change in mode of doing the

configuration in Aruba unlike other switch brands where the configuration would be done in Config-Mode rather than in respective interface configuration mode.

Table 5.31 BPDU Configurations

Brand	Mode	CLI Command
Aruba	N.A.	Command not supported
	C	spanning-tree 3 bpdu-protection
HPE Comware5	System-View	interface g1/0/3
	IS	stp bpdu-protection
HPE Comware7	System-View	interface g1/0/3
	IS	stp bpdu-protection
Cisco	Config-Mode	interface g0/3
	IC	spanning-tree bpduguard enable

Root Guard

This is normally enabled on the designated ports of root a switch once enabled superior BPDUs received by a port enabled root guard receives are ignored. All other BPDUs are accepted and external devices may belong to the spanning tree as long as they do not claim to be root device. Except for syntactic differences, this option can be enabled in switches that belongs to all three brands in a similar manner. However there is a slight change in mode of doing the configuration in Aruba unlike other switch brands where the configuration would be done in Config-Mode rather than in respective interface configuration mode.

Table 5.32 Root Guard Configurations

Brand	Mode	CLI Command
Aruba	N.A.	Command not supported
	C	spanning-tree 4 root-guard
	System-View	interface g1/0/4

HPE Comware5	IS	stp root-protection
HPE Comware7	System-View	interface g1/0/4
	IS	stp root-protection
Cisco	Config-Mode	interface g0/4
	IC	spanning-tree guard root

Loop Guard

Unidirectional link failures may cause a root port or alternate port to become designated as root if BPDUs are absent. Some software failures may introduce temporary loops in the network. The loop guard feature checks if a root port or an alternate root port receives BPDUs. If the port is receiving BPDUs, the loop guard feature puts the port into an inconsistent state until it starts receiving BPDUs again. Except for syntactic differences, this option can be enabled in switches that belongs to all three brands in a similar manner. However there is a slight change in mode of doing the configuration in Aruba unlike other switch brands where the configuration would be done in Config-Mode rather than in respective interface configuration mode.

Table 5.33 Loop Guard Configurations

Brand	**Mode**	**CLI Command**
Aruba	N.A.	Command not supported
	C	loop-protect 5 receiver-action send-disable
HPE Comware5	System-View	interface g1/0/5
	IS	loopback-detection enable
HPE Comware7	System-View	interface g1/0/5
	IS	loopback-detection enable vlan all
Cisco	Config-Mode	interface g0/5

	IC	spanning-tree guard loop

5.3.13 Access Control Lists (ACL) Creation

Standard Numbered ACLs

When compared Aruba and Cisco configuration of standard numbered ACLs are exactly same where all ACLs have an implicit deny at the bottom. In HPE configuration of standard numbered ACLs are quite dissimilar to Aruba or Cisco. All ACLs in HPE have an implicit allow at the bottom unlike Cisco and Aruba. Whilst Cisco and Aruba standard numbered ACL numbers range from 1 to 99 HPE ACL numbers range from 2000 to 2999.

Table 5.34 Creation of Standard Numbered ACLs

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	ip access-list standard 50
	SNACLIC	permit 192.168.1.50 0.0.0.0
	SNACLIC	deny ip 192.168.1.0 0.0.0.255 192.168.1.1 0.0.0.0
	SNACLIC	permit ip any any
HPE Comware5	System-View	acl number 2500
	SNACLIS	rule permit source 192.168.1.50 0.0.0.0
	SNACLIS	rule deny ip source 192.168.1.0 0.0.0.255 destination 192.168.1.1 0.0.0.0
	N.A.	Command not necessary - implicit allow at the bottom
HPE Comware7	System-View	acl number 2500
	SNACLIS	rule permit source 192.168.1.50 0.0.0.0

	SNACLIS	rule deny ip source 192.168.1.0 0.0.0.255 destination 192.168.1.1 0.0.0.0
	N.A.	Command not necessary - implicit allow at the bottom
Cisco	Config-Mode	ip access-list standard 50
	SNACLIC	permit 192.168.1.50 0.0.0.0
	SNACLIC	deny ip 192.168.1.0 0.0.0.255 192.168.1.1 0.0.0.0
	SNACLIC	permit ip any any

Standard Named ACLs

When compared Aruba and Cisco configuration of standard named ACLs are exactly same where all ACLs have an implicit deny at the bottom. In HPE configuration of standard named ACLs are quite dissimilar to Aruba or Cisco. All ACLs in HPE have an implicit allow at the bottom unlike Cisco and Aruba. Whilst Cisco and Aruba standard named ACL numbers range from 1 to 99 HPE ACL numbers range from 2000 to 2999. Further in HPE named ACL creation require ACL number as well unlike Cisco and Aruba.

Table 5.35 Creation of Standard named ACLs

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	ip access-list standard standard_named_acl
	SNAACLIC	permit 192.168.1.50 0.0.0.0
	SNAACLIC	deny ip 192.168.1.0 0.0.0.255 192.168.1.1 0.0.0.0
	SNAACLIC	permit ip any any
HPE Comware5	System-View	acl number 2600 name standard_named_acl

	SNAACLIS	rule permit source 192.168.1.50 0.0.0.0
	SNAACLIS	rule deny ip source 192.168.1.0 0.0.0.255 destination 192.168.1.1 0.0.0.0
	N.A.	Command not necessary - implicit allow at the bottom
HPE Comware7	System-View	acl number 2600 name standard_named_acl
	SNAACLIS	rule permit source 192.168.1.50 0.0.0.0
	SNAACLIS	rule deny ip source 192.168.1.0 0.0.0.255 destination 192.168.1.1 0.0.0.0
	N.A.	Command not necessary - implicit allow at the bottom
Cisco	Config-Mode	ip access-list standard standard_named_acl
	SNAACLIC	permit 192.168.1.50 0.0.0.0
	SNAACLIC	deny ip 192.168.1.0 0.0.0.255 192.168.1.1 0.0.0.0
	SNAACLIC	permit ip any any

Extended Numbered ACLs

When compared Aruba and Cisco configuration of extended numbered ACLs are exactly same where all ACLs have an implicit deny at the bottom. In HPE configuration of extended numbered ACLs are quite dissimilar to Aruba or Cisco. All ACLs in HPE have an implicit allow at the bottom unlike Cisco and Aruba. Whilst Cisco and Aruba extended numbered ACL numbers range from 100 to 199 HPE ACL numbers range from 3000 to 3999.

Table 5.36 Creation of Extended numbered ACLs

Brand	Mode	CLI Command
Aruba	Config-Mode	ip access-list extended 150
	ENACLIC	permit 192.168.1.50 0.0.0.0
	ENACLIC	deny ip 192.168.1.0 0.0.0.255 192.168.1.1 0.0.0.0
	ENACLIC	permit ip any any
HPE Comware5	System-View	acl number 3500
	ENACLIS	rule permit source 192.168.1.50 0.0.0.0
	ENACLIS	rule deny ip source 192.168.1.0 0.0.0.255 destination 192.168.1.1 0.0.0.0
	N.A.	Command not necessary - implicit allow at the bottom
HPE Comware7	System-View	acl number 3500
	ENACLIS	rule permit source 192.168.1.50 0.0.0.0
	ENACLIS	rule deny ip source 192.168.1.0 0.0.0.255 destination 192.168.1.1 0.0.0.0
	N.A.	Command not necessary - implicit allow at the bottom
Cisco	Config-Mode	ip access-list extended 150
	ENACLIC	permit 192.168.1.50 0.0.0.0
	ENACLIC	deny ip 192.168.1.0 0.0.0.255 192.168.1.1 0.0.0.0
	ENACLIC	permit ip any any

Extended Named ACLs

When compared Aruba and Cisco configuration of extended named ACLs are exactly same where all ACLs have an implicit deny at the bottom. In HPE configuration of extended named ACLs are quite dissimilar to Aruba or Cisco. All ACLs in HPE have an implicit allow at the bottom unlike Cisco and Aruba. Whilst Cisco and Aruba extended named ACL numbers range from 100 to 199 HPE ACL numbers range from 3000 to 3999. Further in HPE named ACL creation require ACL number as well unlike Cisco and Aruba.

Table 5.37 Creation of Extended named ACLs

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	ip access-list standard extended_named_acl
	ENAACLIC	permit 192.168.1.50 0.0.0.0
	ENAACLIC	deny ip 192.168.1.0 0.0.0.255 192.168.1.1 0.0.0.0
	ENAACLIC	permit ip any any
HPE Comware5	System-View	acl number 3600 name extended_named_acl
	ENAACLIS	rule permit source 192.168.1.50 0.0.0.0
	ENAECLIS	rule deny ip source 192.168.1.0 0.0.0.255 destination 192.168.1.1 0.0.0.0
	N.A.	Command not necessary - implicit allow at the bottom
HPE Comware7	System-View	acl number 3600 name extended_named_acl
	ENAACLIS	rule permit source 192.168.1.50 0.0.0.0
	ENAACLIS	rule deny ip source 192.168.1.0 0.0.0.255 destination 192.168.1.1 0.0.0.0
	N.A.	Command not necessary - implicit allow at the bottom

Cisco	Config-Mode	ip access-list standard extended_named_acl
	ENAACLIC	permit 192.168.1.50 0.0.0.0
	ENAACLIC	deny ip 192.168.1.0 0.0.0.255 192.168.1.1 0.0.0.0
	ENAACLIC	permit ip any any

Routed ACL Assignment

Standard Numbered ACLs

Routed ACL assignment is quite similar in all brands except for syntactic differences. However mode of operation of ACL assignment is dissimilar in Aruba.

Table 5.38 Assignment of Standard numbered ACLs to VLANs

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	vlan 200
	VIC	ip access-group 50 [in | out]
HPE Comware5	System-View	interface Vlan-interface 200
	VVIS	packet-filter 2500 [inbound | outbound]
HPE Comware7	System-View	interface Vlan-interface 200
	VVIS	packet-filter 2500 [inbound | outbound]
Cisco	Config-Mode	interface vlan 200
	VVIC	ip access-group 50 [in | out]

Standard Named ACLs

Routed ACL assignment is quite similar in all brands except for syntactic differences. However mode of operation of ACL assignment is dissimilar in Aruba.

Table 5.39 Assignment of Standard named ACLs to VLANs

Brand	Mode	CLI Command
Aruba	Config-Mode	vlan 400
	VIC	ip access-group standard_named_acl [in | out]
HPE Comware5	System-View	interface Vlan-interface 400
	VVIS	packet-filter 2600 [inbound | outbound]
HPE Comware7	System-View	interface Vlan-interface 400
	VVIS	packet-filter 2600 [inbound | outbound]
Cisco	Config-Mode	interface vlan 400
	VVIC	ip access-group standard_named_acl [in | out]

Extended Numbered ACLs

Routed ACL assignment is quite similar in all brands except for syntactic differences. However mode of operation of ACL assignment is dissimilar in Aruba.

Table 5.40 Assignment of Extended numbered ACLs to VLANs

Brand	Mode	CLI Command
Aruba	Config-Mode	vlan 200
	VIC	ip access-group 150 [in | out]
HPE Comware5	System-View	interface Vlan-interface 200
	VVIS	packet-filter 3500 [inbound | outbound]
HPE Comware7	System-View	interface Vlan-interface 200
	VVIS	packet-filter 3500 [inbound | outbound]
Cisco	Config-Mode	interface vlan 200
	VVIC	ip access-group 150 [in | out]

Extended Named ACLs

Routed ACL assignment is quite similar in all brands except for syntactic differences. However mode of operation of ACL assignment is dissimilar in Aruba.

Table 5.41 Assignment of Extended named ACLs to VLANs

Brand	Mode	CLI Command
Aruba	Config-Mode	vlan 400
	VIC	ip access-group extended_named_acl [in | out]
HPE Comware5	System-View	interface Vlan-interface 400
	VVIS	packet-filter 3600 [inbound | outbound]
HPE Comware7	System-View	interface Vlan-interface 400
	VVIS	packet-filter 3600 [inbound | outbound]
Cisco	Config-Mode	interface vlan 400
	VVIC	ip access-group extended_named_acl [in | out]

Port ACL Assignment

Standard Numbered ACLs

Port ACL assignment is quite similar in all brands except for syntactic differences.

Table 5.42 Assignment of Standard numbered ACLs to Ports

Brand	Mode	CLI Command
Aruba	Config-Mode	Interface 4
	IC	ip access-group 50 [in | out]
HPE Comware5	System-View	interface g1/0/4
	IS	packet-filter 2500 [inbound | outbound]
	System-View	interface g1/0/4

HPE Comware7	IS	packet-filter 2500 [inbound | outbound]
Cisco	Config-Mode	interface g0/4
	IC	ip access-group 50 [in | out]

Standard Named ACLs

Port ACL assignment is quite similar in all brands except for syntactic differences.

Table 5.43 Assignment of Standard named ACLs to Ports

Brand	Mode	CLI Command
Aruba	Config-Mode	Interface 5
	IC	ip access-group standard_named_acl [in | out]
HPE Comware5	System-View	interface g1/0/5
	IS	packet-filter 2600 [inbound | outbound]
HPE Comware7	System-View	interface g1/0/5
	IS	packet-filter 2600 [inbound | outbound]
Cisco	Config-Mode	interface g0/5
	IC	ip access-group standard_named_acl [in | out]

Extended Numbered ACLs

Port ACL assignment is quite similar in all brands except for syntactic differences.

Table 5.44 Assignment of Extended numbered ACLs to Ports

Brand	Mode	CLI Command
Aruba	Config-Mode	Interface 6
	IC	ip access-group 150 [in | out]

HPE Comware5	System-View	interface g1/0/6
	IS	packet-filter 3500 [inbound | outbound]
HPE Comware7	System-View	interface g1/0/6
	IS	packet-filter 3500 [inbound | outbound]
Cisco	Config-Mode	interface g0/6
	IC	ip access-group 150 [in | out]

Extended Named ACLs

Port ACL assignment is quite similar in all brands except for syntactic differences.

Table 5.45 Assignment of Extended named ACLs to Ports

Brand	**Mode**	**CLI Command**
Aruba	Config-Mode	Interface 7
	IC	ip access-group extended_named_acl [in | out]
HPE Comware5	System-View	interface g1/0/7
	IS	packet-filter 3600 [inbound | outbound]
HPE Comware7	System-View	interface g1/0/7
	IS	packet-filter 3600 [inbound | outbound]
Cisco	Config-Mode	interface g0/7
	IC	ip access-group extended_named_acl [in | out]

5.4 Implementation of Vendor Neutral Command Converter Sub-System

Development of vendor neutral command converter sub-system was done based on the same set of configuration routines considered in vendor specific command converter sub-system. The design of new vendor neutral CLI that is capable of managing all three selected brands was done based on the guidelines mentioned below. Further the design

process was aided by the command comparison tables given under vendor specific command converter sub-system. The vendor neutral CLI design guidelines were designed similar to IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) Revision 1003.1-2017 [16].

Guidelines of vendor neutral CLI design:

1. Commands are designed in two formats namely;
    i. Short format - syntax is shorter making it convenient to type.
    ii. Long format - syntax is longer making the command more descriptive of its function.
2. Commands having options will be defined as follows;
    i. Short format - single hyphen "-" followed by single character option descriptor followed by option input value.
    ii. Long format - double hyphen "--" followed by option descriptor followed by option input value. All options should be separated by a single space.
3. All options and option values should be given separated by single space " ".
4. Command could be input in either short format or long format. It could also be input as a mix of short and long format.
5. All commands and option descriptors should be defined in a case sensitive manner.
6. Commands that are exactly or substantially similar within all the selected brands should be designed in a substantially similar syntax.

Guideline No. 6 was introduced to reduce the learning curve of Network Engineers who are already familiar with a certain brands CLI commands.

Initial vendor neutral command converter sub-system was developed using Python 3 and it was decided that commands that needs conversion to be stored in Lists inside multidimensional Dictionaries similar to vendor specific sub-system. The initial prototype was developed to get the vendor neutral command through standard input (stdin) and to convert it to vendor specific command and provide the output to standard output (stdout). See Appendix E for code.

5.4.1 Transitioning Between Modes

The commands in the tables given below were developed with reference to guideline No. 1, 5 and 6

Table 5.46 Activate Enable Mode using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	User-Mode	en
Example	U	
Long format	User-Mode	enable
Example	U	

Table 5.47 Activate Configure Mode using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Privileged-Mode	config
Example	P	
Long format	User-Mode	configure
Example	P	

Table 5.48 Quit an Operations Mode using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short/long format	Any	exit
Example	Any	exit

Virtual Terminal Configuration Mode

Commands given in the table below would work for all other brands except for Aruba due to Aruba OS not supporting this mode.

Table 5.49 Switch to Virtual Terminal Interface Configuration using Vendor Neutral CLI

Vendor Neutral	Mode	Command
Short/long format	Config-Mode	Interface vty [START #] [END #]
Example	C	Interface vty 0 15

Interface Configuration Mode

This command designed to match with the design guidelines mentioned above support for all three brands.

Table 5.50 Switch to Interface Configuration Mode using Vendor Neutral CLI

Vendor Neutral	Mode	Command
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 1 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 1 --type g

Interface Range Configuration Mode

This command designed to match with the design guidelines mentioned above support only for Cisco and HPE Comware 7 due to the fact that other brands and firmware versions not supporting this mode.

Table 5.51 Switch to Interface Range Configuration Mode using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface -r [START ITNERFACE #] [END ITNERFACE #] -t [e | f | g | T]

Example	C	Interface -r 1 4 -t g
Long format	Config-Mode	Interface --range [START ITNERFACE #] [END ITNERFACE #] --type [e | f | g | T]
Example	C	Interface --range 1 4 --type g

VLAN Configuration Mode

This command designed to match with the design guidelines mentioned above support for all three brands.

Table 5.52 Switch to VLAN Configuration Interface using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	vlan -n [VLAN # <1-4094>]
Example	C	vlan -n 200
Long format	Config-Mode	vlan --number [VLAN # <1-4094>]
Example	C	vlan --number 200

VLAN Interface Configuration Mode

This command designed to match with the design guidelines mentioned above support only for Cisco and HPE. Aruba does not support this mode instead Aruba Switch firmware makes use of VIC mode for the same configuration.

Table 5.53 Switch to VLAN Interface Configuration mode using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface vlan -n [VLAN # <1-4094>]
Example	C	Interface vlan -n 300
Long format	Config-Mode	Interface vlan --number [VLAN # <1-4094>]
Example	C	Interface vlan --number 300

Link Aggregation Configuration Mode

This command designed to match with the design guidelines mentioned above support for all three brands. However the implementation of link aggregation with respect to Aruba was done with a different approach due to the reason of Aruba implementation of link aggregation being distinctly different from other brands.

Table 5.54 Switch to Link Aggregation Configuration mode using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short/long format	Config-Mode	Interface aggregation [AGGREGATION#]
Example	C	Interface aggregation 1

STP Region Configuration Mode

Even though this command designed to match with the design guidelines mentioned above supports for all three brands the mode of operation of this set of commands is different in Aruba.

Table 5.55 Switch to MSTP Configuration Mode using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface stp -r
Example	C	
Long format	Config-Mode	Interface stp --region-configuration
Example	C	

Standard Numbered ACL Configuration Mode

This command designed to match with the design guidelines mentioned above support for all three brands. But the implementation of the command with respect to HPE had to be done in a different approach, where intermediate inputs would be requested by the end

user due to the reason of HPE ACL number ranges being different from Cisco and Aruba brands.

Table 5.56 Switch to Standard numbered ACL Configuration Mode using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface acl -s [ACL # <1-99>]
Example	C	Interface acl -s 50
Long format	Config-Mode	Interface acl --standard [ACL # <1-99>]
Example	C	Interface acl --standard 50

Standard Named ACL Configuration Mode

This command designed to match with the design guidelines mentioned above support for all three brands. The command had to be designed to take the ACL number as an optional input, though it is not required for Aruba and Cisco brands. The reason for the design of the command with unnecessary input option was due to HPE brand specific implementation requiring the ACL number.

Table 5.57 Switch to Standard named ACL Configuration Mode using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface acl -s [ACL # <1-99>] -n [ACL NAME STRING]
Example	C	Interface acl -s 60 -n standard_named_acl
Long format	Config-Mode	Interface acl --standard [ACL # <1-99>] --name [ACL NAME STRING]
Example	C	Interface acl --standard 60 --name standard_named_acl

Extended Numbered ACL Configuration Mode

This command designed to match with the design guidelines mentioned above support for all three brands. But the implementation of the command with respect to HPE had to

be done in a different approach, where intermediate inputs would be requested by the end user due to the reason of HPE ACL number ranges being different from Cisco and Aruba brands.

Table 5.58 Switch to Extended numbered ACL Configuration Mode using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface acl -e [ACL # <100-199>]
Example	C	Interface acl -e 150
Long format	Config-Mode	Interface acl --extended [ACL # <100-199>]
Example	C	Interface acl --extended 150

Extended Named ACL Configuration Mode

This command designed to match with the design guidelines mentioned above support for all three brands. The command had to be designed to take the ACL number as an optional input, though it is not required for Aruba and Cisco brands. The reason for the design of the command with unnecessary input option was due to HPE brand specific implementation requiring the ACL number.

Table 5.59 Switch to Extended named ACL Configuration Mode using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface acl -e [ACL # <100-199>] -n [ACL NAME STRING]
Example	C	Interface acl -e 160 -n extended_named_acl
Long format	Config-Mode	Interface acl --extended [ACL # <100-199>] --name [ACL NAME STRING]
Example	C	Interface acl --extended 160 --name extended_named_acl

5.4.2 Save Configurations

Table 5.60 Save Configurations using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short/long format	Config-Mode	save
Example	C	save

5.4.3 View Configurations

These commands given in the table below were designed to match with the design guidelines mentioned above and supports all three brands.

Table 5.61 View Configuration Details using Vendor Neutral CLI

Vendor Neutral Short format Command	**Vendor Neutral Long format Command**
Privileged-Mode	Privileged-Mode
View general configuration details	
show -f	show --flash
show -v	show --version
show -s	show --sysinfo
show -m	show --modules
show -R	show --Running-config
show -S	show --Startup-config
show -h	show --history
show -l	show --logging
show ip -r	show ip --route
show -t	show --tech-support
show -F	show --Fan
show -P	show --Power

show -T	show --Temperature
View ssh configuration details	
show -u	show --users
show ssh	show ssh
show ssh -c	show ssh --crypto
View LLDP details	
show lldp -n	show lldp --neighbors
show lldp -n -I [ITNERFACE #] -t [e | f | g | T]	show lldp --neighbors --Interface [ITNERFACE #] --type [e | f | g | T]
View interface details	
show ip -i	show ip --interface-brief
show -i	show --interface-brief
show -I [ITNERFACE #] -t [e | f | g | T] -b	show --Interface [ITNERFACE #] --type [e | f | g | T] --brief
show -I [ITNERFACE #] -t [e | f | g | T]	show --Interface [ITNERFACE #] --type [e | f | g | T]
View VLAN details	
show vlan	show vlan
show vlan [VLAN # <1-4094>]	show vlan [VLAN # <1-4094>]
View LACP details	
show lacp	show lacp
show lacp -p	show lacp --port
View STP details	
show stp	show stp
show stp -r	show stp --region-configuration
show stp -i [INSTANCE #]	show stp --instance [INSTANCE #]

View STP hardening details	
show stp -d	show stp --dldp
show stp -D	show stp --Dldp-statistics
View Command details	
show	show

5.4.4 SSH Configurations

These commands were designed to match with the design guidelines mentioned above support for all three brands. A subset of commands mentioned in the table given below are not supported in Aruba due to limitations in Aruba firmware, but this would not affect the configuration of connectivity to Aruba through ssh.

Table 5.62 SSH Configurations using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short/long format	Config-Mode	hostname [HOST NAME]
Example	C	hostname SWITCH
Short/long format	Config-Mode	domain [DOMAIN NAME]
Example	C	domain domain.com
Short format	Config-Mode	ssh -c gen
Example	C	
Long format	Config-Mode	ssh --crypto generate
Example	C	
Short format	Config-Mode	ssh en
Example	C	
Long format	Config-Mode	ssh enable

Example	C	
Short/long format	Config-Mode	Interface vty [START #] [END #]
Example	C	Interface vty 0 15
Short format	VTYIC	login -m [password | AAA]
Example	VTYIC	login -m password
Long format	VTYIC	login --mode [password | AAA]
Example	VTYIC	login --mode password
Short format	VTYIC	inbound -p [ssh | telnet | all]
Example	VTYIC	inbound -p ssh
Long format	VTYIC	inbound --protocol [ssh | telnet | all]
Example	VTYIC	inbound --protocol ssh

5.4.5 Link Layer Discovery Protocol (LLDP) Configurations

This command was designed to match with the design guidelines mentioned above support for all three brands.

Table 5.63 Activate LLDP using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	lldp en
Example	C	
Long format	Config-Mode	lldp enable
Example	C	

5.4.6 Port Management

These commands were designed to match with the design guidelines mentioned above, support for all three brands. However due to Aruba firmware only having single command

to configure both duplex and speed the conversion to Aruba would require the end user to input an intermediate value.

Table 5.64 Port Management Configurations using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 1 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 1 --type g
Short/long format	IC	description [DESCRIPTION STRING]
Example	IC	description link-to-core
Short/long format	IC	duplex [auto | full | half]
Example	IC	duplex auto
Short/long format	IC	speed [10 | 100 | 1000 | 10000 | auto]
Example	IC	speed auto
Short format	IC	dis
Example	IC	
Long format	IC	disable
Example	IC	
Short format	IC	en
Example	IC	
Long format	IC	enable
Example	IC	

5.4.7 VLAN Management

Data

These commands were designed to match with the design guidelines mentioned above, support for all three brands.

Table 5.65 Creation and Naming data VLANs using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	vlan -n [VLAN #]
Example	C	vlan -n 800
Long format	Config-Mode	vlan --number [VLAN #]
Example	C	vlan --number 800
Short format	VIC	name [DESCRIPTION STRING] -t [voice | data]
Example	VIC	name test -t data
Example	VIC	name test
Long format	VIC	name [DESCRIPTION STRING] --type [voice | data]
Example	VIC	name test --type data
Example	VIC	name test

Voice

These commands were designed to match with the design guidelines mentioned above, support for all three brands.

Table 5.66 Creation and Naming of voice VLANs using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	vlan -n [VLAN #]

Example	C	vlan -n 500
Long format	Config-Mode	vlan --number [VLAN #]
Example	C	vlan --number 500
Short format	VIC	name [DESCRIPTION STRING] -t [voice | data]
Example	VIC	name voice -t voice
Long format	VIC	name [DESCRIPTION STRING] --type [voice | data]
Example	VIC	name voice --type voice

5.4.8 VLAN Assignment

Trunk

These commands were designed to match with the design guidelines mentioned above, support for all three brands. However due to the reason of Aruba not having compatibility for permitting all the VLANs at once intermediate inputs are required by end users when conversion is done to Aruba brand. Further configuration of multiple physical ports at once would only be supported with HPE Comware 7 and Cisco due to other brands and firmware versions not supporting it.

Table 5.67 Assignment of Trunk Ports using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 2 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 2 --type g

Short format	IC	trunk -p -v [VLAN# .. <1-4094>]
Example	IC	trunk -p -v 200 100 240
Long format	IC	trunk --permit --vlan [VLAN# .. <1-4094>]
Example	IC	trunk --permit --vlan 200 100 240
Short format	IC	trunk -p -v all
Example	IC	
Long format	IC	trunk --permit --vlan all
Example	IC	
Short format	Config-Mode	Interface -r [START ITNERFACE #] [END ITNERFACE #] -t [e | f | g | T]
Example	C	Interface -r 1 2 -t g
Long format	Config-Mode	Interface --range [START ITNERFACE #] [END ITNERFACE #] --type [e | f | g | T]
Example	C	Interface --range 1 2 --type g
Short format	IC	trunk -p -v all
Example	IC	
Long format	IC	trunk --permit --vlan all
Example	IC	

Access

These commands were designed to match with the design guidelines mentioned above, support for all three brands. Configuration of multiple physical ports at once would only be supported with HPE Comware 7 and Cisco due to other brands and firmware versions not supporting it.

Table 5.68 Assignment of Access Ports using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 4 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 4 --type g
Short format	IC	access -v [VLAN# <1-4094>]
Example	IC	access -v 400
Long format	IC	access --vlan [VLAN# <1-4094>]
Example	IC	access --vlan 400
Short format	Config-Mode	Interface -r [START ITNERFACE #] [END ITNERFACE #] -t [e | f | g | T]
Example	C	Interface -r 4 8 -t g
Long format	Config-Mode	Interface --range [START ITNERFACE #] [END ITNERFACE #] --type [e | f | g | T]
Example	C	Interface --range 4 8 --type g
Short format	IC	access -v [VLAN# <1-4094>]
Example	IC	access -v 400
Long format	IC	access --vlan [VLAN# <1-4094>]
Example	IC	access --vlan 400

Voice

These commands were designed to match with the design guidelines mentioned above, support for all three brands. Configuration of multiple physical ports at once would only be supported with HPE Comware 7 and Cisco due to other brands and firmware versions not supporting it.

Table 5.69 Assignment of Voice Ports using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 10 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 10 --type g
Short format	IC	hybrid tagged -v [VLAN#]
Example	IC	hybrid tagged -v 500
Long format	IC	hybrid tagged --vlan [VLAN#]
Example	IC	hybrid tagged --vlan 500
Short format	IC	hybrid untagged -v [VLAN#]
Example	IC	hybrid untagged -v 800
Long format	IC	hybrid untagged --vlan [VLAN#]
Example	IC	hybrid untagged --vlan 800
Short format	Config-Mode	Interface -r [START ITNERFACE #] [END ITNERFACE #] -t [e | f | g | T]
Example	C	Interface -r 10 15 -t g
Long format	Config-Mode	Interface --range [START ITNERFACE #] [END ITNERFACE #] --type [e | f | g | T]
Example	C	Interface --range 10 15 --type g
Short format	IC	hybrid tagged -v [VLAN#]
Example	IC	hybrid tagged -v 500
Long format	IC	hybrid tagged --vlan [VLAN#]
Example	IC	hybrid tagged --vlan 500
Short format	IC	hybrid untagged -v [VLAN#]
Example	IC	hybrid untagged -v 800

Long format	IC	hybrid untagged --vlan [VLAN#]
Example	IC	hybrid untagged --vlan 800

5.4.9 VLAN Interface IP Address Assignment

These commands were designed to match with the design guidelines mentioned above, support for all three brands. Design of this command was done with special consideration to guideline No. 6 given above.

Table 5.70 Address assignment for VLAN interfaces using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface vlan -n [VLAN # <1-4094>]
Example	C	Interface vlan -n 100
Long format	Config-Mode	Interface vlan --number [VLAN # <1-4094>]
Example	C	Interface vlan --number 100
Short/long format	VVIC	ip address [IP ADDRESS] [SUBNET MASK]
Example	VVIC	ip address 192.168.1.1 255.255.255.0

5.4.10 Default Route Configuration

These commands were designed to match with the design guidelines mentioned above, support for all three brands. Design of this command was done with special consideration to guideline No. 6 given above.

Table 5.71 Default Route Configuration using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short/long format	Config-Mode	ip route [SOURCE IP] [WILDCARD MASK] [NEXTHOP IP]
Example	C	ip route 0.0.0.0 0.0.0.0 192.168.1.1

5.4.11 Link Aggregation Control Protocol (LACP) Configurations

These commands were designed to match with the design guidelines mentioned above, support for all three brands. However implementation specific to Aruba was done in a distinct way due to difference in Aruba configuration routine.

Table 5.72 LACP Configurations using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short/long format	Config-Mode	Interface aggregation [AGGREGATION#]
Example	C	Interface aggregation 1
Short format	BAIC	trunk -p -v [VLAN # .. <1-4094>]
Example	BAIC	trunk -p -v 200 100 240
Long format	BAIC	trunk --permit --vlan [VLAN # .. <1-4094>]
Example	BAIC	trunk --permit --vlan 200 100 240
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 23 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 23 --type g
Short format	IC	aggregation -n [AGGREGATION#]
Example	IC	aggregation -n 1
Long format	IC	aggregation --number [AGGREGATION#]
Example	IC	aggregation --number 1
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 24 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 24 --type g
Short format	IC	aggregation -n [AGGREGATION#]

Example	IC	aggregation -n 1
Long format	IC	aggregation --number [AGGREGATION#]
Example	IC	aggregation --number 1
Short format	Config-Mode	Interface -r [START ITNERFACE #] [END ITNERFACE #] -t [e | f | g | T]
Example	C	Interface -r 23 24 -t g
Long format	Config-Mode	Interface --range [START ITNERFACE #] [END ITNERFACE #] --type [e | f | g | T]
Example	C	Interface --range 23 24 --type g
Short format	IRC	aggregation -n [AGGREGATION#]
Example	IRC	aggregation -n 1
Long format	IRC	aggregation --number [AGGREGATION#]
Example	IRC	aggregation --number 1

5.4.12 Multiple Spanning Tree Protocol (MSTP) Configurations

MSTP Configurations

These commands were designed to match with the design guidelines mentioned above, support for all three brands. However the implementation of conversion had to be implemented in a distinct way due to the reason of Aruba mode of operation in configuring MSTP being different from other brands.

Table 5.73 MSTP Configurations using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	stp en
Example	C	
Long format	Config-Mode	stp enable
Example	C	

Short format	Config-Mode	Interface stp -r
Example	C	
Long format	Config-Mode	Interface stp --region-configuration
Example	C	
Short format	STPRIC	region -n [NAME STRING]
Example	STPRIC	region -n SWMSTP
Long format	STPRIC	region --name [NAME STRING]
Example	STPRIC	region --name SWMSTP
Short format	STPRIC	region -r [REVISION #]
Example	STPRIC	region -r 1
Long format	STPRIC	region --revision [REVISION #]
Example	STPRIC	region --revision 1
Short format	STPRIC	instance [INSTANCE#] -v [VLAN#.. <1-4094>]
Example	STPRIC	instance 1 -v 200 100 240
Example	STPRIC	instance 2 -v 500 800
Long format	STPRIC	instance [INSTANCE#] --vlan [VLAN#.. <1-4094>]
Example	STPRIC	instance 1 --vlan 200 100 240
Example	STPRIC	instance 2 --vlan 500 800
Short format	Config-Mode	stp -p [PRIORITY#]
Example	C	stp -p 4096
Long format	Config-Mode	stp --priority [PRIORITY#]
Example	C	stp --priority 4096
Short format	Config-Mode	stp -i [INSTANCE#] -p [PRIORITY#]
Example	C	stp -i 1 -p 8192
Example	C	stp -i 2 -p 12288

Long format	Config-Mode	stp --instance [INSTANCE#] --priority [PRIORITY#]
Example	C	stp --instance 1 --priority 8192
Example	C	stp --instance 2 --priority 12288
Short/long format	Config-Mode	stp pathcost [8021d | 8021t]
Example	C	stp pathcost 8021t
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 20 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 20 --type g
Short format	IC	stp -c [COST VALUE]
Example	IC	stp -c 20000
Long format	IC	stp --cost [COST VALUE]
Example	IC	stp --cost 20000
Short format	IC	stp -p [PRIORITY VALUE]
Example	IC	stp -p 16394
Long format	IC	stp --priority [PRIORITY VALUE]
Example	IC	stp --priority 16394
Short format	IC	stp -i [INSTANCE #] -c [COST VALUE]
Example	IC	stp -i 1 -c 20000
Long format	IC	stp --instance [INSTANCE #] --cost [COST VALUE]
Example	IC	stp --instance 1 --cost 20000
Short format	IC	stp -i [INSTANCE #] -p [PRIORITY VALUE]
Example	IC	stp -i 1 -p 16384

Long format	IC	stp --instance [INSTANCE #] --priority [PRIORITY VALUE]
Example	IC	stp --instance 1 --priority 16384

Table 5.74 MSTP Configurations using Vendor Neutral CLI

MSTP Hardening

Data Link Discovery Protocol (DLDP)

These commands were designed to match with the design guidelines mentioned above, support for all three brands.

Table 5.75 DLDP Configurations using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 1 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 1 --type g
Short format	IC	stp dldp en
Example	IC	
Long format	IC	stp dldp enable
Example	IC	

Edge-port

These commands were designed to match with design guidelines mentioned above, which support all three brands. However implementation in terms of Aruba brand had to be done in a distinct manner due to Aruba configuration mode being different from other brands.

Table 5.76 Edge port Configurations using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 2 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 2 --type g
Short format	IC	stp edge-port en
Example	IC	
Long format	IC	stp edge-port enable
Example	IC	

Bridge Protocol Data Unit (BPDU) Guard

These commands were designed to match with the design guidelines mentioned above, support for all three brands. However implementation in terms of Aruba brand had to be done in a distinct manner due to Aruba configuration mode being different from other brands.

Table 5.77 BPDU guard Configurations using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 3 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 3 --type g
Short format	IC	stp bpdu-guard en
Example	IC	
Long format	IC	stp bpdu-guard enable
Example	IC	

Root Guard

These commands were designed to match with the design guidelines mentioned above, support all three brands. However implementation in terms of Aruba brand had to be done in a distinct manner due to Aruba configuration mode being different from other brands.

Table 5.78 Root guard Configurations using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 4 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 4 --type g
Short format	IC	stp root-guard en
Example	IC	
Long format	IC	stp root-guard enable
Example	IC	

Loop Guard

These commands were designed to match with the design guidelines mentioned above, support all three brands. However implementation in terms of Aruba brand had to be done in a distinct manner due to Aruba configuration mode being different from other brands.

Table 5.79 Loop guard Configuration using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 5 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 5 --type g
Short format	IC	stp loop-guard en

Example	IC	
Long format	IC	stp loop-guard enable
Example	IC	

5.4.13 Access Control Lists (ACL) Creation

Standard Numbered ACLs

These commands were designed to match with the design guidelines mentioned above, support for all three brands. There seems to be some differences in ACL implementation in HPE compared with Aruba and Cisco, where Aruba and Cisco have an implicit deny at the bottom when HPE has a implicit allow at the bottom.

Table 5.80 Creation of Standard numbered ACLs using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface acl -s [ACL # <1-99>]
Example	C	Interface acl -s 50
Long format	Config-Mode	Interface acl --standard [ACL # <1-99>]
Example	C	Interface acl --standard 50
Short format	SNACLIC	permit -i [IP ADDR] -w [WILDCARD]
Example	SNACLIC	permit -i 192.168.1.50 -w 0.0.0.0
Long format	SNACLIC	permit --ip-address [IP ADDR] --wildcard [WILDCARD]
Example	SNACLIC	permit --ip-address 192.168.1.50 --wildcard 0.0.0.0
Short format	SNACLIC	deny src -i [IP ADDR] -w [WILDCARD] dst -i [IP ADDR] -w [WILDCARD]
Example	SNACLIC	deny src -i 192.168.1.0 -w 0.0.0.255 dst dst -i 192.168.1.1 -w 0.0.0.0

Long format	SNACLIC	deny source --ip-address [IP ADDR] --wildcard [WILDCARD] destination --ip-address [IP ADDR] --wildcard [WILDCARD]
Example	SNACLIC	deny source --ip-address 192.168.1.0 --wildcard 0.0.0.255 destination --ip-address 192.168.1.1 --wildcard 0.0.0.0
Short format	SNACLIC	permit -i any any
Example	SNACLIC	
Long format	SNACLIC	permit --ip-address any any
Example	SNACLIC	

Standard Named ACLs

These commands were designed to match with the design guidelines mentioned above, support all three brands. There seems to be some differences in ACL implementation in HPE compared with Aruba and Cisco, where Aruba and Cisco have an implicit deny at the bottom when HPE has an implicit allow at the bottom.

Table 5.81 Creation of Standard named ACLs using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface acl -s [ACL # <1-99>] -n [ACL NAME STRING]
Example	C	Interface acl -s 60 -n standard_named_acl
Long format	Config-Mode	Interface acl --standard [ACL # <1-99>] --name [ACL NAME STRING]
Example	C	Interface acl --standard 60 --named standard_named_acl
Short format	SNAACLIC	permit -i [IP ADDR] -w [WILDCARD]
Example	SNAACLIC	permit -i 192.168.1.50 -w 0.0.0.0

Long format	SNAACLIC	permit --ip-address [IP ADDR] --wildcard [WILDCARD]
Example	SNAACLIC	permit --ip-address 192.168.1.50 --wildcard 0.0.0.0
Short format	SNAACLIC	deny src -i [IP ADDR] -w [WILDCARD] dst -i [IP ADDR] -w [WILDCARD]
Example	SNAACLIC	deny src -i 192.168.1.0 -w 0.0.0.255 dst dst -i 192.168.1.1 -w 0.0.0.0
Long format	SNAACLIC	deny source --ip-address [IP ADDR] --wildcard [WILDCARD] destination --ip-address [IP ADDR] --wildcard [WILDCARD]
Example	SNAACLIC	deny source --ip-address 192.168.1.0 --wildcard 0.0.0.255 destination --ip-address 192.168.1.1 --wildcard 0.0.0.0
Short format	SNAACLIC	permit -i any any
Example	SNAACLIC	
Long format	SNAACLIC	permit --ip-address any any
Example	SNAACLIC	

Extended Numbered ACLs

These commands were designed to match with the design guidelines mentioned above, support all three brands. There seems to be some differences in ACL implementation in HPE compared with Aruba and Cisco, where Aruba and Cisco have an implicit deny at the bottom when HPE has an implicit allow at the bottom.

Table 5.82 Creation of Extended numbered ACLs using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface acl -e [ACL # <100-`99>]
Example	C	Interface acl -e 150

Long format	Config-Mode	Interface acl --extended [ACL # <100-199>]
Example	C	Interface acl --extended 150
Short format	ENACLIC	permit -i [IP ADDR] -w [WILDCARD]
Example	ENACLIC	permit -i 192.168.1.50 -w 0.0.0.0
Long format	ENACLIC	permit --ip-address [IP ADDR] --wildcard [WILDCARD]
Example	ENACLIC	permit --ip-address 192.168.1.50 --wildcard 0.0.0.0
Short format	ENACLIC	deny src -i [IP ADDR] -w [WILDCARD] dst -i [IP ADDR] -w [WILDCARD]
Example	ENACLIC	deny src -i 192.168.1.0 -w 0.0.0.255 dst dst -i 192.168.1.1 -w 0.0.0.0
Long format	ENACLIC	deny source --ip-address [IP ADDR] --wildcard [WILDCARD] destination --ip-address [IP ADDR] --wildcard [WILDCARD]
Example	ENACLIC	deny source --ip-address 192.168.1.0 --wildcard 0.0.0.255 destination --ip-address 192.168.1.1 --wildcard 0.0.0.0
Short format	ENACLIC	permit -i any any
Example	ENACLIC	
Long format	ENACLIC	permit --ip-address any any
Example	ENACLIC	

Extended Named ACLs

These commands were designed to match with the design guidelines mentioned above, support for all three brands. There seems to be some differences in ACL implementation in HPE compared with Aruba and Cisco, where Aruba and Cisco have an implicit deny at the bottom when HPE has an implicit allow at the bottom.

Table 5.83 Creation of Extended named ACLs using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface acl -e [ACL # <100-199>] -n [ACL NAME STRING]
Example	C	Interface acl -e 160 -n extended_named_acl
Long format	Config-Mode	Interface acl --extended [ACL # <100-199>] --name [ACL NAME STRING]
Example	C	Interface acl --extended 160 --named extended_named_acl
Short format	ENAACLIC	permit -i [IP ADDR] -w [WILDCARD]
Example	ENAACLIC	permit -i 192.168.1.50 -w 0.0.0.0
Long format	ENAACLIC	permit --ip-address [IP ADDR] --wildcard [WILDCARD]
Example	ENAACLIC	permit --ip-address 192.168.1.50 --wildcard 0.0.0.0
Short format	ENAACLIC	deny src -i [IP ADDR] -w [WILDCARD] dst -i [IP ADDR] -w [WILDCARD]
Example	ENAACLIC	deny src -i 192.168.1.0 -w 0.0.0.255 dst dst -i 192.168.1.1 -w 0.0.0.0
Long format	ENAACLIC	deny source --ip-address [IP ADDR] --wildcard [WILDCARD] destination --ip-address [IP ADDR] --wildcard [WILDCARD]
Example	ENAACLIC	deny source --ip-address 192.168.1.0 --wildcard 0.0.0.255 destination --ip-address 192.168.1.1 --wildcard 0.0.0.0
Short format	ENAACLIC	permit -i any any
Example	ENAACLIC	

Long format	ENAACLIC	permit --ip-address any any
Example	ENAACLIC	

Routed ACL Assignment

Standard Numbered ACLs

These commands were designed to match with the design guidelines mentioned above, support all three brands. However the mode of operation in which assignment of ACLs are done is different from other brands. Therefore implementation had to be done in a distinct manner for Aruba.

Table 5.84 Assignment of Standard numbered ACLs to VLANs using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface vlan -n [VLAN # <1-4094>]
Example	C	Interface vlan -n 200
Long format	Config-Mode	Interface vlan --number [VLAN # <1-4094>]
Example	C	Interface vlan --number 200
Short format	VVIC	acl [ACL# <1-99>] -m [inbound | outbound]
Example	VVIC	acl 50 -m inbound
Long format	VVIC	acl [ACL# <1-99>] --mode [inbound | outbound]
Example	VVIC	acl 50 --mode inbound

Standard Named ACLs

These commands were designed to match with the design guidelines mentioned above, support all three brands. However the mode of operation in which assignment of ACLs are done is different from other brands. Therefore implementation had to be done in a distinct manner for Aruba.

Table 5.85 Assignment of Standard named ACLs to VLANs using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface vlan -n [VLAN # <1-4094>]
Example	C	Interface vlan -n 400
Long format	Config-Mode	Interface vlan --number [VLAN # <1-4094>]
Example	C	Interface vlan --number 400
Short format	IC	acl -n [ACL NAME STRING] -m [inbound | outbound]
Example	IC	acl -n standard_named_acl -m inbound
Long format	IC	acl --name [ACL NAME STRING] --mode [inbound | outbound]
Example	IC	acl --name standard_named_acl --mode inbound

Extended Numbered ACLs

These commands were designed to match with the design guidelines mentioned above, support all three brands. However the mode of operation in which assignment of ACLs are done is different from other brands. Therefore implementation had to be done in a distinct manner for Aruba.

Table 5.86 Assignment of Extended numbered ACLs to VLANs using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface vlan -n [VLAN # <1-4094>]
Example	C	Interface vlan -n 200
Long format	Config-Mode	Interface vlan --number [VLAN # <1-4094>]
Example	C	Interface vlan --number 200
Short format	IC	acl [ACL# <100-199>] -m [inbound | outbound]
Example	IC	acl 150 -m inbound

Long format	IC	acl [ACL# <100-199>] --mode [inbound | outbound]
Example	IC	acl 150 --mode inbound

Extended Named ACLs

These commands were designed to match with the design guidelines mentioned above, support all three brands. However the mode of operation in which assignment of ACLs are done is different from other brands. Therefore implementation had to be done in a distinct manner for Aruba.

Table 5.87 Assignment of Extended named ACLs to VLANs using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface vlan -n [VLAN # <1-4094>]
Example	C	Interface vlan -n 400
Long format	Config-Mode	Interface vlan --number [VLAN # <1-4094>]
Example	C	Interface vlan --number 400
Short format	IC	acl -n [ACL NAME STRING] -m [inbound | outbound]
Example	IC	acl -n extended_named_acl -m inbound
Long format	IC	acl --name [ACL NAME STRING] --mode [inbound | outbound]
Example	IC	acl --name extended_named_acl --mode inbound

Port ACL Assignment

Standard Numbered ACLs

These commands were designed to match with the design guidelines mentioned above, support all three brands.

Table 5.88 Assignment of Standard numbered ACLs to Ports using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 4 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 4 --type g
Short format	IC	acl [ACL# <1-99>] -m [inbound | outbound]
Example	IC	acl 50 -m inbound
Long format	IC	acl [ACL# <1-99>] --mode [inbound | outbound]
Example	IC	acl 50 --mode inbound

Standard Named ACLs

These commands were designed to match with the design guidelines mentioned above, support all three brands.

Table 5.89 Assignment of Standard named ACLs to Ports using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 5 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 5 --type g
Short format	IC	acl -n [ACL NAME STRING] -m [inbound | outbound]
Example	IC	acl -n standard_named_acl -m inbound
Long format	IC	acl --name [ACL NAME STRING] --mode [inbound | outbound]
Example	IC	acl --name standard_named_acl --mode inbound

Extended Numbered ACLs

These commands were designed to match with the design guidelines mentioned above, supports all three brands.

Table 5.90 Assignment of Extended numbered ACLs to Ports using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 6 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 6 --type g
Short format	IC	acl [ACL# <100-199>] -m [inbound | outbound]
Example	IC	acl 150 -m inbound
Long format	IC	acl [ACL# <100-199>] --mode [inbound | outbound]
Example	IC	acl 150 --mode inbound

Extended Named ACLs

These commands were designed to match with the design guidelines mentioned above, supports all three brands.

Table 5.91 Assignment of Extended Named ACLs to Ports using Vendor Neutral CLI

Vendor Neutral	**Mode**	**Command**
Short format	Config-Mode	Interface [ITNERFACE #] -t [e | f | g | T]
Example	C	Interface 7 -t g
Long format	Config-Mode	Interface [ITNERFACE #] --type [e | f | g | T]
Example	C	Interface 7 --type g
Short format	IC	acl -n [ACL NAME STRING] -m [inbound | outbound]

Example	IC	acl -n extended_named_acl -m inbound
Long format	IC	acl --name [ACL NAME STRING] --mode [inbound | outbound]
Example	IC	acl --name extended_named_acl --mode inbound

5.5 Integration of Netmiko Connection Sub-System and Command Converter Sub-Systems

The development of Netmiko connection sub-system, vendor specific command converter sub-system and vendor neutral command converter sub-system were completed using the process of incremental prototyping technique and tested to satisfaction that commands get converted properly.
Then the connection sub-system was combined with the command converter subsystems to create a complete API that is capable of connecting to a switch of selected brand through ssh, get the end user input of through any preferred CLI supported and to convert the input command to the supported CLI of connected switch.

In the combining process of the said sub-systems Netmiko write_channel() function was utilized instead of python print() command where there wouldn't be any outputs generated by the switch as an output to the command given [17]. But in case of the command having an output that needs to be displayed to the end users (e.g. show commands) the Netmiko send_command() function used [17].

Commands used to change the mode of operation of the switch were dealt using different functions specific to brands. Therefore it can be noted that the development process utilized specific Netmiko functions required for each individual task as required. See Appendix E for code.

In addition to above mentioned development techniques a new command named "state" was introduced to all the APIs to show the current mode of operation due to the reason of certain modes of operation being different from one brand to another.

As a key requirement of any software development this prototype was also developed having “man pages” similar to that of Linux, which explains its end users how to make use of the API to do supported configuration routines on network devices. See Appendix F for code.

5.6 Summary

This chapter describe in detail how the implementation process was conducted. It further represents vendor based command comparisons in tabular form. Then it tabulates the newly designed vendor neutral CLI that was developed to support the three selected brands.

Chapter 6

6.0 Conclusion and Suggestions for Further Works

6.1 Conclusion

This study successfully proves the concept of being able to have a unified vendor neutral interface for network configurations management of heterogeneous network devices. Further it provides a feasible solution to make network device configuration vendor neutral until a vendor neutral interface gets adopted by manufacturers. Adoption of such interface would benefit the networking industry immensely by improving the quality, efficiency and cost effectiveness network configuration management.

Though the study was able to prove the concept, it should be noted that expanding similar solution that would facilitate network configuration management of all vendor platforms supporting all standardized features would be a challenging task that would require the hard work and effort of workgroups and willingness of manufacturers.

This study can be considered as Proof of Concept (PoC) and a guidance on how it can be achieved. Further it shows that implementation of such an interface would be beneficial towards the industry in many ways by increasing the efficiency and cost effectiveness of configurations management field.

6.2 Suggestions for Future Work

The said two interfaces both vendor neutral and vendor specific can be enhanced in many ways. One such improvement would be to increase their abilities to do conversions to support all the standard features in both IPv6 and IPv4 domains. Further they could be improved to have an ability to support a broader range of network brands and products ranging from layer 2 to layer 7.

Other than the improvements to increase support, the said Interfaces can be improved to have a GUI based CLI similar to Terminals in Linux/OSX or Command Prompt in Windows. The above mentioned GUI can provide more informative feed back to the user

about the commands being converted and the current state or mode of operation in connected switch.

Furthermore the prototype implemented in this study can be used as a platform to further improve a fully graphical interface that can be used to manage any vendor based device. Such interface could be implemented having a visual appearance similar to the IDE used in visual programing languages like Scratch.

The amount of effort that has to be utilized in order to maintain a similar interface having support for many brands with full featured CLI would a huge task, requiring lot of man hours of skilled and enthusiastic personnel. Therefore the most suitable way for the sustainable continuity of this king of API would be to maintain through an open source cloud based work group having lots of members with experience in many different vendor specific platforms.

6.3 Limitations

The main limitation of the APIs developed in this study would be the support of only three selected brands. Further it's again limited to a selected group of configuration routines supporting only IPv4. There is another limitation that would be common to even future developments of the interface, where proprietary protocols and configurations cannot be supported. There is another minor limitation that would be affective to the same interface, where initial ssh and ip address configurations have to be done using the switch brands native CLI in order to make use of this API.

References

[1] H. Zimmermann, “OSI Reference Model-The IS0 Model of Architecture for Open Systems Interconnection,” *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425 - 432, 1980.

[2] J, Case; R, Mundy; D, Partain; B, Stewart, “RFC3410 Introduction and Applicability Statements for Internet Standard Management Framework,” December 2002. [Online]. Available: https://tools.ietf.org/html/rfc3410. [Accessed 28 March 2019].

[3] J, Schoenwaelder, “RFC3535 IAB Network Management Workshop,” May 2003. [Online]. Available: https://tools.ietf.org/html/rfc3535. [Accessed 28 March 2019].

[4] R, Enns; M, Bjorklund; J, Schoenwaelder; A, Bierman, “RFC6241 Network Configuration Protocol (NETCONF),” June 2011. [Online]. Available: https://tools.ietf.org/html/rfc6241. [Accessed 28 March 2019].

[5] S, Chisholm; H, Trevino, “RFC5277 NETCONF Event Notifications,” July 2008. [Online]. Available: https://tools.ietf.org/html/rfc5277. [Accessed 28 March 2019].

[6] M, Bjorklund, “RFC6020 YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF),” October 2010. [Online]. Available: https://tools.ietf.org/html/rfc6020. [Accessed 28 March 2019].

[7] SNMPcenter, “Why use NETCONF/YANG when you can use SNMP and CLI?,” Social Neural Mobile Predictive Center, 25 October 2016. [Online]. Available: https://snmpcenter.com/why-use-netconf/. [Accessed 28 March 2019].

[8] IEEE, IEEE Guide to POSIX Open System Enviorenment (OSE), New York: The Institute of Electrical and Electronics Engineers, Inc., 1995.

[9] Z. H, “Posix Standard,” Linux Hint LLC, 28 March 2018. [Online]. Available: https://linuxhint.com/posix-standard/. [Accessed 28 March 2019].

[10] J. Forcier, “Welcome to Paramiko’s documentation!,” Paramiko, 2018. [Online]. Available: http://docs.paramiko.org/en/2.4/#welcome-to-paramiko-s-documentation. [Accessed 28 March 2019].

[11] K. Byers, “Netmiko,” Netmiko, 19 March 2019. [Online]. Available: https://github.com/ktbyers/netmiko. [Accessed 28 March 2019].

[12] Aruba, ArubaOS-Switch, Comware and Cisco IOS CLI Reference Guide, Santa Clara, CA: Hewlett Packard Enterprise Company, 2017.

[13] Python, “Sunsetting Python 2,” python.org, [Online]. Available: https://www.python.org/doc/sunset-python-2/. [Accessed 05 04 2020].

[14] Tutorialspoint, “SDLC - Software Prototype Model - Tutorialspoint,” tutorialspoint.com, [Online]. Available: https://www.tutorialspoint.com/sdlc/sdlc_software_prototyping.htm. [Accessed 05 04 2020].

[15] K. Byers, “Netmiko Library,” Twin Bridges Technology, 02 01 2019. [Online]. Available: https://pynet.twb-tech.com/blog/automation/netmiko.html. [Accessed 05 04 2020].

[16] IEEE, “IEEE Standard for Information Technology— Portable Operating System Interface,” IEEE, New York, 2018.

[17] Netmiko, “Module netmiko,” [Online]. Available: https://ktbyers.github.io/netmiko/docs/netmiko/index.html. [Accessed 05 04 2020].

[18] R. Pointer and J. Forcier, “Paramiko,” Paramiko, 03 January 2019. [Online]. Available: https://github.com/paramiko/paramiko. [Accessed 28 March 2019].

[19] W. Odom, Cisco CCNA Routing and Switchning ICND2 200-101, Noida U. P: Dorling Kindersley, 2014.

[20] K. Byers, “BaseConnection,” Netmiko, 2016. [Online]. Available: https://netmiko.readthedocs.io/en/stable/classes/base_connection.html. [Accessed 28 March 2019].

[21] PythonSoftwareFoundation, “Python 3.7.3 documentation,” Python Software Foundation, 28 March 2019. [Online]. Available: https://docs.python.org/3/. [Accessed 28 March 2019].

Appendix A

```
#!/usr/bin/env python
import time
from netmiko import Netmiko
from getpass import getpass

HP_1920 = {
    "ip": "192.168.132.136",
    "username": "admin",
    "password": getpass(),
    "device_type": "hp_comware",
}

net_connect = Netmiko(**HP_1920)
command = "_cmdline-mode on"
print()
print(net_connect.find_prompt())
output = net_connect.send_command_timing(command)
if "All commands can be displayed and executed. Continue? [Y/N]" in output:
        output += net_connect.send_command_timing("y", strip_prompt=False,strip_command=False)

if "Please input password:" in output:
        output +=
net_connect.send_command_timing("Jinhua1920unauthorized",strip_prompt=False,strip_command=False)

command = "sys"
output = net_connect.send_command_timing(command)
if "System View: return to User View with Ctrl+Z." in output:
        output += net_connect.send_command_timing("display interface brief",
strip_prompt=False,strip_command=False)
        output += net_connect.send_command_timing(" ", strip_prompt=False,strip_command=False)


print(output)
print(net_connect.find_prompt())

net_connect.disconnect()
```

Appendix B

```
from __future__ import print_function
from __future__ import unicode_literals
import time
from netmiko.cisco_base_connection import CiscoSSHConnection


class HPComwareJinhuaBase(CiscoSSHConnection):

    def session_preparation(self):
        """
        Prepare the session after the connection has been established.
        Extra time to read HP banners.
        """
        delay_factor = self.select_delay_factor(delay_factor=0)
        i = 1
        while i <= 4:
            # Comware can have a banner that prompts you to continue
            # 'Press Y or ENTER to continue, N to exit.'
            time.sleep(.5 * delay_factor)
            self.write_channel("\n")
            i += 1

        #log into command line mode by command line break
        time.sleep(0.3 * delay_factor)                      # adding a delay to execute the command
        self.write_channel("_cmdline-mode on \n")           # enter command line break command to channel
        self.write_channel("y \n")                          # enter y
        self.write_channel("Jinhua1920unauthorized\n")  # enter command line break password

        time.sleep(.3 * delay_factor)
        self.clear_buffer()
        self._test_channel_read(pattern=r'[>\]]')
        self.set_base_prompt()
        command = self.RETURN + "screen-length disable"
        self.disable_paging(command=command)
        # Clear the read buffer
        time.sleep(.3 * self.global_delay_factor)
        self.clear_buffer()

    def config_mode(self, config_command='system-view'):
        """Enter configuration mode."""
        return super(HPComwareJinhuaBase, self).config_mode(config_command=config_command)

    def exit_config_mode(self, exit_config='return', pattern=r'>'):
        """Exit config mode."""
        return super(HPComwareJinhuaBase, self).exit_config_mode(exit_config=exit_config,
                                                                 pattern=pattern)

    def check_config_mode(self, check_string=']'):
        """Check whether device is in configuration mode. Return a boolean."""
        return super(HPComwareJinhuaBase, self).check_config_mode(check_string=check_string)

    def set_base_prompt(self, pri_prompt_terminator='>', alt_prompt_terminator=']',
                        delay_factor=1):
        """
        Sets self.base_prompt

        Used as delimiter for stripping of trailing prompt in output.

        Should be set to something that is general and applies in multiple contexts. For Comware
        this will be the router prompt with < > or [ ] stripped off.
```

```
        This will be set on logging in, but not when entering system-view
        """
        prompt = super(HPComwareJinhuaBase, self).set_base_prompt(
            pri_prompt_terminator=pri_prompt_terminator,
            alt_prompt_terminator=alt_prompt_terminator,
            delay_factor=delay_factor)

        # Strip off leading character
        prompt = prompt[1:]
        prompt = prompt.strip()
        self.base_prompt = prompt
        return self.base_prompt

    def enable(self, cmd='system-view'):
        """enable mode on Comware is system-view."""
        return self.config_mode(config_command=cmd)

    def exit_enable_mode(self, exit_command='return'):
        """enable mode on Comware is system-view."""
        return self.exit_config_mode(exit_config=exit_command)

    def check_enable_mode(self, check_string=']'):
        """enable mode on Comware is system-view."""
        return self.check_config_mode(check_string=check_string)

    def save_config(self, cmd='save force', confirm=False):
        """Save Config."""
        return super(HPComwareJinhuaBase, self).save_config(cmd=cmd, confirm=confirm)


class HPComwareJinhuaSSH(HPComwareJinhuaBase):
    pass


class HPComwareJinhuaTelnet(HPComwareJinhuaBase):
    def __init__(self, *args, **kwargs):
        default_enter = kwargs.get('default_enter')
        kwargs['default_enter'] = '\r\n' if default_enter is None else default_enter
        super(HPComwareJinhuaTelnet, self).__init__(*args, **kwargs)
```

########## __init__.py ##########

```
from __future__ import unicode_literals
from netmiko.hp.hp_procurve import HPProcurveSSH, HPProcurveTelnet
from netmiko.hp.hp_comware import HPComwareSSH, HPComwareTelnet
from netmiko.hp.hp_comwarejinhua import HPComwareJinhuaSSH, HPComwareJinhuaTelnet
from netmiko.hp.hp_comware512900 import HPComware512900SSH, HPComware512900Telnet
from netmiko.hp.hp_comwarefoes import HPComwareFoesSSH, HPComwareFoesTelnet

__all__ = ['HPProcurveSSH', 'HPProcurveTelnet', 'HPComwareSSH', 'HPComwareTelnet',
'HPComwareJinhuaSSH', 'HPComwareJinhuaTelnet', 'HPComware512900SSH', 'HPComware512900Telnet',
'HPComwareFoesSSH', 'HPComwareFoesTelnet']
```

##########

########## ssh_dispatcher.py ##########

```
"""Controls selection of proper class based on the device type."""
from __future__ import unicode_literals

from netmiko.a10 import A10SSH
from netmiko.accedian import AccedianSSH
from netmiko.alcatel import AlcatelAosSSH
from netmiko.alcatel import AlcatelSrosSSH
from netmiko.arista import AristaSSH, AristaTelnet
from netmiko.arista import AristaFileTransfer
from netmiko.apresia import ApresiaAeosSSH, ApresiaAeosTelnet
from netmiko.aruba import ArubaSSH
from netmiko.calix import CalixB6SSH, CalixB6Telnet
from netmiko.checkpoint import CheckPointGaiaSSH
from netmiko.ciena import CienaSaosSSH
from netmiko.cisco import CiscoAsaSSH, CiscoAsaFileTransfer
from netmiko.cisco import CiscoIosSSH, CiscoIosFileTransfer, CiscoIosTelnet, CiscoIosSerial
from netmiko.cisco import CiscoNxosSSH, CiscoNxosFileTransfer
from netmiko.cisco import CiscoS300SSH
from netmiko.cisco import CiscoTpTcCeSSH
from netmiko.cisco import CiscoWlcSSH
from netmiko.cisco import CiscoXrSSH, CiscoXrFileTransfer
from netmiko.citrix import NetscalerSSH
from netmiko.coriant import CoriantSSH
from netmiko.dell import DellDNOS6SSH
from netmiko.dell import DellDNOS6Telnet
from netmiko.dell import DellForce10SSH
from netmiko.dell import DellOS10SSH, DellOS10FileTransfer
from netmiko.dell import DellPowerConnectSSH
from netmiko.dell import DellPowerConnectTelnet
from netmiko.dell import DellIsilonSSH
from netmiko.eltex import EltexSSH
from netmiko.enterasys import EnterasysSSH
from netmiko.extreme import ExtremeErsSSH
from netmiko.extreme import ExtremeExosSSH
from netmiko.extreme import ExtremeExosTelnet
from netmiko.extreme import ExtremeNetironSSH
from netmiko.extreme import ExtremeNetironTelnet
from netmiko.extreme import ExtremeNosSSH
from netmiko.extreme import ExtremeSlxSSH
from netmiko.extreme import ExtremeVspSSH
from netmiko.extreme import ExtremeWingSSH
from netmiko.f5 import F5TmshSSH
from netmiko.f5 import F5LinuxSSH
from netmiko.fortinet import FortinetSSH
```

```
from netmiko.hp import HPProcurveSSH, HPProcurveTelnet, HPComwareSSH, HPComwareTelnet,
HPComwareJinhuaSSH, HPComwareJinhuaTelnet, HPComware512900SSH, HPComware512900Telnet,
HPComwareFoesSSH, HPComwareFoesTelnet
from netmiko.huawei import HuaweiSSH, HuaweiVrpv8SSH
from netmiko.ipinfusion import IpInfusionOcNOSSSH, IpInfusionOcNOSTelnet
from netmiko.juniper import JuniperSSH, JuniperTelnet
from netmiko.juniper import JuniperFileTransfer
from netmiko.linux import LinuxSSH, LinuxFileTransfer
from netmiko.mellanox import MellanoxSSH
from netmiko.mrv import MrvOptiswitchSSH
from netmiko.netapp import NetAppcDotSSH
from netmiko.ovs import OvsLinuxSSH
from netmiko.paloalto import PaloAltoPanosSSH
from netmiko.paloalto import PaloAltoPanosTelnet
from netmiko.pluribus import PluribusSSH
from netmiko.quanta import QuantaMeshSSH
from netmiko.rad import RadETXSSH
from netmiko.rad import RadETXTelnet
from netmiko.ruckus import RuckusFastironSSH
from netmiko.ruckus import RuckusFastironTelnet
from netmiko.terminal_server import TerminalServerSSH
from netmiko.terminal_server import TerminalServerTelnet
from netmiko.ubiquiti import UbiquitiEdgeSSH
from netmiko.vyos import VyOSSSH


# The keys of this dictionary are the supported device_types
CLASS_MAPPER_BASE = {
    'a10': A10SSH,
    'accedian': AccedianSSH,
    'alcatel_aos': AlcatelAosSSH,
    'alcatel_sros': AlcatelSrosSSH,
    'apresia_aeos': ApresiaAeosSSH,
    'arista_eos': AristaSSH,
    'aruba_os': ArubaSSH,
    'avaya_ers': ExtremeErsSSH,
    'avaya_vsp': ExtremeVspSSH,
    'brocade_fastiron': RuckusFastironSSH,
    'brocade_netiron': ExtremeNetironSSH,
    'brocade_nos': ExtremeNosSSH,
    'brocade_vdx': ExtremeNosSSH,
    'brocade_vyos': VyOSSSH,
    'checkpoint_gaia': CheckPointGaiaSSH,
    'calix_b6': CalixB6SSH,
    'ciena_saos': CienaSaosSSH,
    'cisco_asa': CiscoAsaSSH,
    'cisco_ios': CiscoIosSSH,
    'cisco_nxos': CiscoNxosSSH,
    'cisco_s300': CiscoS300SSH,
    'cisco_tp': CiscoTpTcCeSSH,
    'cisco_wlc': CiscoWlcSSH,
    'cisco_xe': CiscoIosSSH,
    'cisco_xr': CiscoXrSSH,
    'coriant': CoriantSSH,
    'dell_dnos9': DellForce10SSH,
    'dell_force10': DellForce10SSH,
    'dell_os6': DellDNOS6SSH,
    'dell_os9': DellForce10SSH,
    'dell_os10': DellOS10SSH,
    'dell_powerconnect': DellPowerConnectSSH,
    'dell_isilon': DellIsilonSSH,
    'eltex': EltexSSH,
    'enterasys': EnterasysSSH,
    'extreme': ExtremeExosSSH
```

```
    'extreme_ers': ExtremeErsSSH,
    'extreme_exos': ExtremeExosSSH,
    'extreme_netiron': ExtremeNetironSSH,
    'extreme_nos': ExtremeNosSSH,
    'extreme_slx': ExtremeSlxSSH,
    'extreme_vdx': ExtremeNosSSH,
    'extreme_vsp': ExtremeVspSSH,
    'extreme_wing': ExtremeWingSSH,
    'f5_ltm': F5TmshSSH,
    'f5_tmsh': F5TmshSSH,
    'f5_linux': F5LinuxSSH,
    'fortinet': FortinetSSH,
    'generic_termserver': TerminalServerSSH,
    'hp_comware': HPComwareSSH,
    'hp_comwarejinhua': HPComwareJinhuaSSH,
    'hp_comware512900': HPComware512900SSH,
    'hp_comwarefoes': HPComwareFoesSSH,
    'hp_procurve': HPProcurveSSH,
    'huawei': HuaweiSSH,
    'huawei_vrpv8': HuaweiVrpv8SSH,
    'ipinfusion_ocnos': IpInfusionOcNOSSSH,
    'juniper': JuniperSSH,
    'juniper_junos': JuniperSSH,
    'linux': LinuxSSH,
    'mellanox': MellanoxSSH,
    'mrv_optiswitch': MrvOptiswitchSSH,
    'netapp_cdot': NetAppcDotSSH,
    'netscaler': NetscalerSSH,
    'ovs_linux': OvsLinuxSSH,
    'paloalto_panos': PaloAltoPanosSSH,
    'pluribus': PluribusSSH,
    'quanta_mesh': QuantaMeshSSH,
    'rad_etx': RadETXSSH,
    'ruckus_fastiron': RuckusFastironSSH,
    'ubiquiti_edge': UbiquitiEdgeSSH,
    'ubiquiti_edgeswitch': UbiquitiEdgeSSH,
    'vyatta_vyos': VyOSSSH,
    'vyos': VyOSSSH,
}

FILE_TRANSFER_MAP = {
    'arista_eos': AristaFileTransfer,
    'cisco_asa': CiscoAsaFileTransfer,
    'cisco_ios': CiscoIosFileTransfer,
    'dell_os10': DellOS10FileTransfer,
    'cisco_nxos': CiscoNxosFileTransfer,
    'cisco_xe': CiscoIosFileTransfer,
    'cisco_xr': CiscoXrFileTransfer,
    'juniper_junos': JuniperFileTransfer,
    'linux': LinuxFileTransfer,
}

# Also support keys that end in _ssh
new_mapper = {}
for k, v in CLASS_MAPPER_BASE.items():
    new_mapper[k] = v
    alt_key = k + u"_ssh"
    new_mapper[alt_key] = v
CLASS_MAPPER = new_mapper

new_mapper = {}
for k, v in FILE_TRANSFER_MAP.items():
    new_mapper[k] = v
```

```
    alt_key = k + u"_ssh"
    new_mapper[alt_key] = v
FILE_TRANSFER_MAP = new_mapper

# Add telnet drivers
CLASS_MAPPER['apresia_aeos_telnet'] = ApresiaAeosTelnet
CLASS_MAPPER['arista_eos_telnet'] = AristaTelnet
CLASS_MAPPER['brocade_fastiron_telnet'] = RuckusFastironTelnet
CLASS_MAPPER['brocade_netiron_telnet'] = ExtremeNetironTelnet
CLASS_MAPPER['calix_b6_telnet'] = CalixB6Telnet
CLASS_MAPPER['cisco_ios_telnet'] = CiscoIosTelnet
CLASS_MAPPER['dell_dnos6_telnet'] = DellDNOS6Telnet
CLASS_MAPPER['dell_powerconnect_telnet'] = DellPowerConnectTelnet
CLASS_MAPPER['extreme_telnet'] = ExtremeExosTelnet
CLASS_MAPPER['extreme_exos_telnet'] = ExtremeExosTelnet
CLASS_MAPPER['extreme_netiron_telnet'] = ExtremeNetironTelnet
CLASS_MAPPER['generic_termserver_telnet'] = TerminalServerTelnet
CLASS_MAPPER['hp_procurve_telnet'] = HPProcurveTelnet
CLASS_MAPPER['hp_comware_telnet'] = HPComwareTelnet
CLASS_MAPPER['hp_comwarejunhua_telnet'] = HPComwareJinhuaTelnet
CLASS_MAPPER['hp_comware512900_telnet'] = HPComware512900Telnet
CLASS_MAPPER['hp_comwarefoes_telnet'] = HPComwareFoesTelnet
CLASS_MAPPER['ipinfusion_ocnos_telnet'] = IpInfusionOcNOSTelnet
CLASS_MAPPER['juniper_junos_telnet'] = JuniperTelnet
CLASS_MAPPER['paloalto_panos_telnet'] = PaloAltoPanosTelnet
CLASS_MAPPER['rad_etx_telnet'] = RadETXTelnet
CLASS_MAPPER['ruckus_fastiron_telnet'] = RuckusFastironTelnet

# Add serial drivers
CLASS_MAPPER['cisco_ios_serial'] = CiscoIosSerial

# Add general terminal_server driver and autodetect
CLASS_MAPPER['terminal_server'] = TerminalServerSSH
CLASS_MAPPER['autodetect'] = TerminalServerSSH

platforms = list(CLASS_MAPPER.keys())
platforms.sort()
platforms_base = list(CLASS_MAPPER_BASE.keys())
platforms_base.sort()
platforms_str = "\n".join(platforms_base)
platforms_str = "\n" + platforms_str

scp_platforms = list(FILE_TRANSFER_MAP.keys())
scp_platforms.sort()
scp_platforms_str = "\n".join(scp_platforms)
scp_platforms_str = "\n" + scp_platforms_str


def ConnectHandler(*args, **kwargs):
    """Factory function selects the proper class and creates object based on device_type."""
    if kwargs['device_type'] not in platforms:
        raise ValueError('Unsupported device_type: '
                         'currently supported platforms are: {}'.format(platforms_str))
    ConnectionClass = ssh_dispatcher(kwargs['device_type'])
    return ConnectionClass(*args, **kwargs)


def ssh_dispatcher(device_type):
    """Select the class to be instantiated based on vendor/platform."""
    return CLASS_MAPPER[device_type]


def redispatch(obj, device_type, session_prep=True):
    """Dynamically change Netmiko object's class to proper class.
```

```
    Generally used with terminal_server device_type when you need to redispatch after interacting
    with terminal server.
    """
    new_class = ssh_dispatcher(device_type)
    obj.device_type = device_type
    obj.__class__ = new_class
    if session_prep:
        obj._try_session_preparation()


def FileTransfer(*args, **kwargs):
    """Factory function selects the proper SCP class and creates object based on device_type."""
    if len(args) >= 1:
        device_type = args[0].device_type
    else:
        device_type = kwargs['ssh_conn'].device_type
    if device_type not in scp_platforms:
        raise ValueError('Unsupported SCP device_type: '
                         'currently supported platforms are: {}'.format(scp_platforms_str))
    FileTransferClass = FILE_TRANSFER_MAP[device_type]
    return FileTransferClass(*args, **kwargs)

##########################################################################################################
```

Appendix D

```
############################ Comware5_to_other.py ##############################################
# Sample Code Block

#!/usr/bin/env python3

EXIT = False

COMWARE5_STATE = 'S'
INTERFACE_NUMBER = 0
PATH_COST_STANDARD = ''
SYSTEM_VIEW_COUNT = 2

########## Display commands ########################

comware5_show = {'dir': {'cisco': ['show','flash'], 'aruba': ['show','flash'], 'comware7': ['dir']},
            'version': {'cisco': ['show','version'], 'aruba': ['show','version'], 'comware7': ['display','version']},
            'device': {'manuinfo': {'cisco': ['show','inventory'], 'aruba': ['show','system','information'], 'comware7':
['display','device','manuinfo']},
                       'verbose': {'cisco': ['show','version'], 'aruba': ['show','modules'], 'comware7':
['display','device','verbose']}},
            'current-configuration': {'cisco': ['show','run'], 'aruba': ['show','run'], 'comware7': ['display','current-
configuration']},
            'saved-configuration': {'cisco': ['show','start'], 'aruba': ['show','config'], 'comware7': ['display','saved-
configuration']},
            'history': {'cisco': ['show','history'], 'aruba': ['show','history'], 'comware7': ['display','history']},
            'info-center': {'cisco': ['show','logging'], 'aruba': ['show','logging'], 'comware7': ['display','info-center']},
            'ip': {'routing-table': {'cisco': ['show','ip','route'], 'aruba': ['show','ip','route'], 'comware7':
['display','ip','routing-table']},
                   'interface': {'brief': {'cisco': ['show','ip','interface','brief'], 'aruba': ['show','ip'], 'comware7':
['display','ip','interface','brief']}}},
            'diagnostic-information': {'cisco': ['show','tech-support'], 'aruba': ['show','tech'], 'comware7':
['display','diagnostic-information']},
            'fan': {'cisco': ['show','env','fan'], 'aruba': ['show','system','fans'], 'comware7': ['display','fan']},
            'power': {'cisco': ['show','env','power'], 'aruba': ['show','system','power-supply'], 'comware7':
['display','power']},
            'enviorenment': {'cisco': ['show','env','temperature'], 'aruba': ['show','system','temperature'], 'comware7':
['display','enviorenment']},
            'users': {'cisco': ['show','users'], 'aruba': ['show','telnet'], 'comware7': ['display','users']},
            'lldp': {'neighbor-information': {'list': {'cisco': ['show','lldp','neighbors'], 'aruba': ['show','lldp','info','remote-
device'], 'comware7': ['display','lldp','neighbor-information','list']},
                                              'brief': {'cisco': ['!# command not supported'],'aruba': ['#command not
supported'],'comware7': ['#command not supported']}}},
            'interface': {'brief': {'cisco': ['show','interfaces','status'], 'aruba': ['show','interfaces','brief'], 'comware7':
['display','interface','brief']}},
            'vlan': {'all': {'cisco': ['!#command not supported'], 'aruba': ['#command not supported'], 'comware7':
['display','vlan','all']},
                     'cisco': ['show','vlan','brief'], 'aruba': ['show','vlans'], 'comware7': ['display','vlan']},
            'link-aggregation': {'summary': {'cisco': ['show','ethernet','channel','summary'], 'aruba': ['show','lacp'],
'comware7': ['display','link-aggregation','summary']},
                       'member-port': {'cisco': ['show','interface','etherchannel'], 'aruba': ['show','lacp','peer'], 'comware7':
['display','link-aggregation','member-port']}},
            'stp': {'region-configuration': {'cisco': ['show','spanning-tree','mst','configuration'], 'aruba':
['show','spanning-tree','mst-config'], 'comware7': ['display','stp','region-configuration']},
                    'root': {'cisco': ['show','spanning-tree','root'], 'aruba': ['#command not supported'], 'comware7':
['display','stp','root']},
                      'cisco': ['show','spanning-tree'], 'aruba': ['show','spanning-tree'], 'comware7': ['display','stp']},
            'cisco': ['show'], 'aruba': ['show'], 'comware7': ['display']
            }

comware5_show_ext = {'lldp': {'neighbor-information': {'interface': {'cisco': ['show', 'lldp', 'neighbors', '0/', 'detail'],
'aruba': ['show', 'lldp', 'info', 'remote-device', None], 'comware7': ['display', 'lldp', 'neighbor-information', 'interface',
'1/0/']}}},
```

```
            'vlan': {'cisco': ['!#command not supported'], 'aruba': ['show', 'vlans', None], 'comware7': ['display', 'vlan',
None]},
            'stp': {'instance': {'cisco': ['show', 'spanning-tree', 'mst', None], 'aruba': ['show', 'spanning-tree', 'instance',
None], 'comware7': ['display', 'stp', 'instance', None]}},
            'interface': {'cisco': ['show', 'interfaces', '0/'], 'aruba': ['show', 'interfaces', None], 'comware7': ['display',
'interface', None]},
            'dldp': {'statistics': {'cisco': ['!# command not supported'], 'aruba': ['show link-keepalive statistics'],
'comware7': ['display dldp statistics']},
                'cisco': ['show','udld','0/'], 'aruba': ['show','link-keepalive'], 'comware7': ['display','dldp']}
            }

####################### system-view modes commands ##########################

comware5_modes = {'system-view'     :{'aruba': ['configure'], 'cisco': ['configure', 'terminal'], 'comware7': ['system-
view']},
          'interface'                :{'Bridge-Aggregation': {'aruba': ['# command not supported'], 'cisco':
['interface', 'port-channel', None], 'comware7': ['interface', 'Bridge-Aggregation', None]},
                    'Vlan-interface': {'aruba': ['vlan', None],'cisco': ['interface','vlan', None],'comware7':
['interface','Vlan-interface', None]},
                    'aruba': ['interface', None], 'cisco': ['interface', '0/'], 'comware7': ['interface', None]},
          'vlan'                     :{'aruba': ['vlan', None], 'cisco': ['vlan', None], 'comware7': ['vlan', None]},
          'stp'                      :{'region-configuration': {'aruba': ['# command not supported'], 'cisco':
['spanning-tree', 'mst', 'configuration'], 'comware7': ['stp', 'region-configuration']}},
          'acl'                      :{'number': {'aruba': ['ip', 'access-list', ['standard', 'extended'], None], 'cisco':
['ip', 'access-list', ['standard', 'extended'], None], 'comware7': ['acl', 'number', None]}},
          'quit'                     :{'aruba': ['exit'], 'cisco': ['exit'], 'comware7': ['quit']},
          'stp'      :{'region-configuration': {'aruba': ['# command not supported'],'cisco':['spanning-
tree','mst','configuration'],'comware7': ['stp','region-configuration']}},
          'user-interface'  :{'vty': {'aruba': ['# command not supported'],'cisco': ['line','vty', None, None],'comware7':
['user-interface','vty', None, None]}}
          }

####################### system-view commands ##########################

comware5_config_S = {'sysname'    : {'aruba': ['hostname', None],'cisco': ['hostname', None],'comware7': ['sysname',
None]},
          'domain'     : {'aruba': ['# command not supported'],'cisco': ['ip','domain-name', None],'comware7':
['domain', None]},
          'public-key' : {'local' : {'create': {'rsa': {'aruba': ['crypto','key','generate','ssh'], 'cisco':
['crypto','key','generate'], 'comware7': ['public-key','local','create','rsa']}}}},
          'ssh'       : {'server': {'enable': {'aruba': ['ip','ssh'],'cisco': ['ip','ssh','version','2'],'comware7':
['ssh','server','enable']}}},
          'user-interface': {'vty': {'aruba': ['# command not supported'],'cisco': ['line','vty', None,
None],'comware7': ['user-interface','vty', None, None]}},
          'protocol'   : {'inbound': {'aruba': ['# command not supported'], 'cisco': ['transport','input', None],
'comware7': ['protocol','inbound', None]}},
          'lldp'      : {'enable': {'aruba': ['lldp','run'],'cisco': ['lldp','run'],'comware7': ['lldp','global','enable']}},
          'ip'        : {'route-static': {'aruba': ['ip','route', None, None, None],'cisco': ['ip','route', None, None,
None],'comware7': ['ip','route-static', None, None, None]}},
          'stp'       : {'enable': {'aruba': ['spanning-tree'],'cisco': ['spanning-tree','mode','mst'],'comware7':
['stp','enable']},
                    'priority': {'aruba': ['spanning-tree','priority', None],'cisco': ['spanning-tree','mst','0','priority',
None],'comware7': ['stp','priority', None]},
                    'instance': {'aruba': ['spanning-tree','instance', None,'priority', None],'cisco': ['spanning-
tree','mst', None,'priority', None],'comware7': ['stp','instance', None,'priority', None]},
                    'pathcost-standard': {'aruba': ['spanning-tree','pathcost','mstp', None],'cisco': ['spanning-
tree','pathcost','method', None],'comware7': ['stp','pathcost-standard', None]}}
          }

comware5_config_IS = {'description' : {'aruba': ['name', None],'cisco': ['description', None],'comware7':
['description', None]},
           'duplex'     : {'aruba': ['speed-duplex', None],'cisco': ['duplex', None],'comware7': ['duplex', None]},
           'speed'      : {'aruba': ['speed-duplex', None],'cisco': ['speed', None],'comware7': ['speed', None]},
           'shutdown'    : {'aruba': ['disable'],'cisco': ['shutdown'],'comware7': ['shutdown']},
```

```
                'undo'       : {'shutdown': {'aruba': ['enable'],'cisco': ['no','shutdown'],'comware7': ['undo','shutdown']}},
                'port'       : {'link-type': {'trunk' : {'aruba': [['vlan', None],['tagged', None]],'cisco':
['switchport','trunk','encapsulation','dot1q'],'comware7': ['port','link-type','trunk']},
                                    'access': {'aruba': [['vlan', None],['untagged', None]],'cisco':
['switchport','mode','access'],'comware7': ['port','link-type','access']},
                                    'hybrid': {'aruba': [['vlan', None],['tagged', None],['untagged',
None]],'cisco':['switchport','mode','access'],'comware7':['port','link-type','hybrid']}},
                                    'trunk': {'permit': {'vlan': {'aruba': ['# command not supported'],'cisco':
['switchport','trunk','allowed','vlan', None],'comware7': ['port','trunk','permit','vlan', None]}}},
                                    'access': {'vlan': {'aruba': ['# command not supported'],'cisco': ['switchport','access','vlan',
None],'comware7': ['port','access','vlan', None]}},
                                    'hybrid': {'vlan': {'aruba': ['# command not supported'],'cisco':['switchport', None,'vlan',
None],'comware7':['port','hybrid','vlan', None, None]}},
                                    'link-aggregation':{'group': {'aruba': ['# command not supported'],'cisco': ['channel-group',
None,'mode', 'active'],'comware7': ['port','link-aggregation','group', None]}}},
                'stp'        : {'cost': {'aruba': ['spanning-tree',None,'path-cost', None],'cisco': ['spanning-tree','cost',
None],'comware7': ['stp','cost', None]},
                                    'port': {'priority': {'aruba': ['spanning-tree', None,'priority', None],'cisco': ['spanning-
tree','port-priority', None],'comware7': ['stp','port','priority', None]}},
                                    'edge-port': {'enable': {'aruba': ['spanning-tree', None,'admin-edge-port'],'cisco': ['spanning-
tree','portfast'],'comware7': ['stp','edge-port']}},
                                    'bpdu-protection': {'aruba': ['spanning-tree', None,'bpdu-protection'],'cisco': ['spanning-
tree','bpduguard','enable'],'comware7': ['stp','bpdu-protection']},
                                    'root-protection': {'aruba': ['spanning-tree', None,'root-guard'],'cisco': ['spanning-
tree','guard','root'],'comware7': ['stp','root-protection']}},
                'loopback-detection': {'enable': {'aruba': ['loop-protect', None,'receiver-action','send-disable'],'cisco':
['spanning-tree','guard','loop'],'comware7': ['loopback-detection','enable','vlan','all']},
                                        'action': {'shutdown': {'aruba': ['# command not supported'],'cisco': ['!#command not
supported'],'comware7': ['loopback-detection','action','shutdown']}}},
                'dldp'       : {'enable': {'aruba': ['link-keepalive'],'cisco': ['udld','port'],'comware7': ['dldp','enable']}}
               }

comware5_config_VIS = {'name'       : {'voice': {'aruba': ['voice'],'cisco': ['name','voice'],'comware7': ['name','voice']},
                                'aruba': ['name', None],'cisco': ['name', None],'comware7': ['name', None]}
                }

comware5_config_VVIS = {'ip'        :{'address': {'aruba':['ip','address', None, None],'cisco':['ip','address', None,
None],'comware7':['ip','address', None, None]}}
                 }

comware5_config_VTYS = {'authentication-mode': {'aruba': ['# command not supported'],'cisco': ['login',
None],'comware7': ['authentication-mode', None]},
                  'protocol': {'inbound': {'aruba': ['# command not supported'],'cisco': ['transport','input',
None],'comware7': ['protocol','inbound',None]}}
                 }

comware5_config_BAIS ={'link-aggregation'   :{'mode': {'aruba': ['# command not supported'],'cisco': ['!#command
not supported'],'comware7': ['link-aggregation','mode', None]}},
                'port'       : {'link-type': {'trunk' : {'aruba': ['# command not supported'],'cisco':
['switchport','trunk','encapsulation','dot1q'],'comware7': ['port','link-type','trunk']},
                                    'access': {'aruba': ['# command not supported'],'cisco':
['switchport','mode','access'],'comware7': ['port','link-type','access']}},
                                    'trunk': {'permit': {'vlan': {'aruba': ['# command not supported'],'cisco':
['switchport','trunk','allowed','vlan', None],'comware7': ['port','trunk','permit','vlan', None]}}},
                                    'access': {'vlan': {'aruba': ['# command not supported'],'cisco': ['switchport','access','vlan',
None],'comware7': ['port','access','vlan', None]}}}
                }

comware5_config_STPRIS = {'region-name' :{'aruba': ['spanning-tree','config-name', None],'cisco': ['name',
None],'comware7': ['region-name', None]},
                  'revision-level'  :{'aruba': ['spanning-tree','config-revision', None],'cisco': ['revision',
None],'comware7': ['revision-level', None]},
                  'instance': {'aruba': ['spanning-tree','instance', None,'vlan', None],'cisco': ['instance', None,'vlan',
None],'comware7': ['instance', None,'vlan', None]},
```

```
                    'active'  : {'region-configuration': {'aruba': ['# command not supported'],'cisco': ['!#command not
supported'],'comware7': ['active','region-configuration']}}
                    }

convertion_type = input("Enter convertion type [aruba | cisco | comware7 ]: ")
while EXIT == False:
    input_string = input("Enter the Comware5 command: ")
    if input_string == 'e':
        EXIT = True

    #print("The command you typed is : " + input_string)

    # using split()
    # to count words in string
    res = len(input_string.split())

    # printing result
    # print ("The number of words in command are : " +  str(res))
    num = int(res)

    command_list = []

    for i in range(num):
        command_list.append(input_string.split(" ")[i])

    # print(command_list)
    seperator = ' '

    # Comware5-t0-Cisco & Comware5-to-Aruba and Comware5-to-Comware7
    # blank input
    if len(command_list) == 0:
      print('#Invalid input')
    # system-view | sys
    elif (command_list[0] == 'system-view' or command_list[0] == 'sys') and len(command_list) == 1 and
COMWARE5_STATE == 'U' and SYSTEM_VIEW_COUNT == 0:
      COMWARE5_STATE = 'S'
      SYSTEM_VIEW_COUNT = SYSTEM_VIEW_COUNT + 1
      if convertion_type == 'aruba':
        print('enable')
      elif convertion_type == 'cisco':
        print('enable')
      elif convertion_type == 'comware7':
        print('system-view')
    # system-view
    elif (command_list[0] == 'system-view' or command_list[0] == 'sys') and len(command_list) == 1 and
COMWARE5_STATE == 'S' and SYSTEM_VIEW_COUNT == 1:
      SYSTEM_VIEW_COUNT = SYSTEM_VIEW_COUNT + 1
      print(seperator.join(comware5_modes['system-view'][convertion_type]))
      COMWARE5_STATE = 'S'
    # quit
    elif command_list[0] == 'quit' and len(command_list) == 1:
      if COMWARE5_STATE == 'VIS' or COMWARE5_STATE == 'VVIS' or COMWARE5_STATE == 'VTYS' or
COMWARE5_STATE == 'BAIS' or COMWARE5_STATE == 'STPRIS' or COMWARE5_STATE == 'SNACLIS' or
COMWARE5_STATE == 'SNAACLIS' or COMWARE5_STATE == 'ENACLIS' or COMWARE5_STATE ==
'ENAACLIS':
        print(seperator.join(comware5_modes[command_list[0]][convertion_type]))
      elif COMWARE5_STATE == 'IS':
        print(seperator.join(comware5_modes[command_list[0]][convertion_type]))
        INTERFACE_NUMBER = 0
        COMWARE5_STATE = 'S'
      elif COMWARE5_STATE == 'S':
        print(seperator.join(comware5_modes[command_list[0]][convertion_type]))
        if SYSTEM_VIEW_COUNT == 2:
          COMWARE5_STATE = 'S'
```

```
        SYSTEM_VIEW_COUNT = SYSTEM_VIEW_COUNT - 1
      elif SYSTEM_VIEW_COUNT == 1:
        COMWARE5_STATE = 'U'
        SYSTEM_VIEW_COUNT = SYSTEM_VIEW_COUNT - 1
  # shutdown
  elif command_list[0] == 'shutdown' and len(command_list) == 1 and COMWARE5_STATE == 'IS':
    if convertion_type == 'aruba':
      print(comware5_config_IS[command_list[0]][convertion_type][0])
    elif convertion_type == 'cisco':
      print(comware5_config_IS[command_list[0]][convertion_type][0])
    elif convertion_type == 'comware7':
      print(comware5_config_IS[command_list[0]][convertion_type][0])
  # dir
  elif command_list[0] == 'dir' and len(command_list) == 1 and COMWARE5_STATE == 'S':
    if convertion_type == 'aruba':
      print(seperator.join(comware5_show[command_list[0]][convertion_type]))
    elif convertion_type == 'cisco':
      print(seperator.join(comware5_show[command_list[0]][convertion_type]))
    elif convertion_type == 'comware7':
      print(seperator.join(comware5_show[command_list[0]][convertion_type]))
  # display
  elif command_list[0] == 'display' and len(command_list) == 1 and COMWARE5_STATE == 'S':
    if convertion_type == 'aruba':
      print(seperator.join(comware5_show[convertion_type]))
    elif convertion_type == 'cisco':
      print(seperator.join(comware5_show[convertion_type]))
    elif convertion_type == 'comware7':
      print(seperator.join(comware5_show[convertion_type]))
  # stp bpdu-protection
  elif command_list[0] == 'stp' and command_list[1] == 'bpdu-protection' and len(command_list) == 2 and
COMWARE5_STATE == 'IS':
    if convertion_type == 'aruba':
      # exit interface configuration mode
      print(seperator.join(comware5_modes['quit'][convertion_type]))
      print(comware5_config_IS[command_list[0]][command_list[1]][convertion_type][0]+'
'+str(INTERFACE_NUMBER)+' '+comware5_config_IS[command_list[0]][command_list[1]][convertion_type][2])
      # go inside interface configuration mode
      print(comware5_modes['interface'][convertion_type][0]+' '+str(INTERFACE_NUMBER))
    elif convertion_type == 'cisco':
      print(comware5_config_IS[command_list[0]][command_list[1]][convertion_type][0]+'
'+comware5_config_IS[command_list[0]][command_list[1]][convertion_type][1]+'
'+comware5_config_IS[command_list[0]][command_list[1]][convertion_type][2])
    elif convertion_type == 'comware7':
      print(comware5_config_IS[command_list[0]][command_list[1]][convertion_type][0]+'
'+comware5_config_IS[command_list[0]][command_list[1]][convertion_type][1])
  # stp root-protection
  elif command_list[0] == 'stp' and command_list[1] == 'root-protection' and len(command_list) == 2 and
COMWARE5_STATE == 'IS':
    if convertion_type == 'aruba':
      # exit interface configuration mode
      print(seperator.join(comware5_modes['quit'][convertion_type]))
      print(comware5_config_IS[command_list[0]][command_list[1]][convertion_type][0]+'
'+str(INTERFACE_NUMBER)+' '+comware5_config_IS[command_list[0]][command_list[1]][convertion_type][2])
      # go inside interface configuration mode
      print(comware5_modes['interface'][convertion_type][0]+' '+str(INTERFACE_NUMBER))
    elif convertion_type == 'cisco':
      print(comware5_config_IS[command_list[0]][command_list[1]][convertion_type][0]+'
'+comware5_config_IS[command_list[0]][command_list[1]][convertion_type][1]+'
'+comware5_config_IS[command_list[0]][command_list[1]][convertion_type][2])
    elif convertion_type == 'comware7':
      print(comware5_config_IS[command_list[0]][command_list[1]][convertion_type][0]+'
'+comware5_config_IS[command_list[0]][command_list[1]][convertion_type][1])

##########################################################################################
```

Appendix E

```
##################### neutral-to-other_config_show.py ############################
# Sample Code block

#!/usr/bin/env python3

import time
import re

from netmiko import Netmiko
from getpass import getpass

def netprint(print_command):
   return print_command

def split(word):
   return [char for char in word]

def commandfilter(start_length,end_length,unfiltered_command_list):
  filtered_command = []
  for x in range(start_length,end_length):
    if unfiltered_command_list[x].startswith('--'):
      filtered_command.append(unfiltered_command_list[x].lstrip('--'))
    elif unfiltered_command_list[x].startswith('-'):
      command_string = unfiltered_command_list[x].lstrip('-')
      if len(command_string) == 1:
        filtered_command.append(command_string)
      elif len(command_string) > 1:
        filtered_command = split(command_string)
    else:
      filtered_command.append(unfiltered_command_list[x])
  return filtered_command

EXIT = False
NEUTRAL_STATE = 'P'
ACL_NAME_NUM = {}
INTERFACE_NUMBER = 0
BRIDGE_AGGREGATION_NUMBER = 0



########### Show commands ########################

neutral_show = {'f'         :{'aruba': ['show','flash'],'comware5': ['dir'],'comware7': ['dir'],'cisco': ['show','flash']},
                'v'         :{'aruba': ['show','version'],'comware5': ['display','version'],'comware7': ['display','version'],'cisco':
['show','version']},
                's'         :{'aruba': ['show','system','information'],'comware5': ['display','device','manuinfo'],'comware7':
['display','device','manuinfo'],'cisco': ['show','inventory']},
                'm'         :{'aruba': ['show','modules'],'comware5': ['display','device','verbose'],'comware7':
['display','device','verbose'],'cisco': ['show','version']},
                'R'         :{'aruba': ['show','run'],'comware5': ['display','current-configuration'],'comware7': ['display','current-
configuration'],'cisco': ['show','run']},
                'S'         :{'aruba': ['show','config'],'comware5': ['display','saved-configuration'],'comware7': ['display','saved-
configuration'],'cisco': ['show','start']},
                'h'         : {'aruba': ['show','history'],'comware5': ['display','history'],'comware7': ['display','history'],'cisco':
['show','history']},
                'l'         : {'aruba': ['show','logging'],'comware5': ['display','info-center'],'comware7': ['display','info-
center'],'cisco': ['show','logging']},
                'ip'        : {'r':{'aruba': ['show','ip','route'],'comware5': ['display','ip','routing-table'],'comware7':
['display','ip','routing-table'],'cisco': ['show','ip','route']},'i':{'aruba': ['show','ip'],'comware5':
['display','ip','interface','brief'],'comware7': ['display','ip','interface','brief'],'cisco': ['show','ip','interface','brief']}},
                't'         :{'aruba': ['show','tech'],'comware5': ['display','diagnostic-information'],'comware7':
['display','diagnostic-information'],'cisco': ['show','tech-support']},
```

```
            'F'         :{'aruba': ['show','system','fans'],'comware5': ['display','fan'],'comware7': ['display','fan'],'cisco':
['show','env','fan']},
            'P'         :{'aruba': ['show','system','power-supply'],'comware5': ['display','power'],'comware7':
['display','power'],'cisco': ['show','env','power']},
            'T'         :{'aruba': ['show','system','temperature'],'comware5': ['display','enviorenment'],'comware7':
['display','enviorenment'],'cisco': ['show','env','temperature']},
            'u'         :{'aruba': ['show','telnet'],'comware5': ['display','users'],'comware7': ['display','users'],'cisco':
['show','users']},
            'ssh'       :{'c': {'aruba': ['show','crypto','host-public-key'],'comware5': ['# command not
supported'],'comware7': ['display','public-key','local','rsa','public'],'cisco': ['show','crypto','key','mypubkey','rsa']},
                        'aruba': ['show','ip','ssh'],'comware5': ['# command not supported'],'comware7':
['display','ssh','server','status'],'cisco': ['show','ip','ssh']},
            'lldp'      :{'n': {'I': {'aruba': ['show','lldp','info','remote-device', None],'comware5': ['display','lldp','neighbor-
information','interface','1/0/'],'comware7': ['display','lldp','neighbor-information','interface','1/0/'],'cisco':
['show','lldp','neighbors','0/','detail']},'aruba': ['show','lldp','info','remote-device'],'comware5': ['display','lldp','neighbor-
information','list'],'comware7': ['display','lldp','neighbor-information','list'],'cisco': ['show','lldp','neighbors']}},
            'i'         :{'aruba': ['show','interfaces','brief'],'comware5': ['display','interface','brief'],'comware7':
['display','interface','brief'],'cisco': ['show','interfaces','status']},
            'vlan'      :{'aruba': ['show','vlans'],'comware5': ['display','vlan'],'comware7': ['display','vlan'],'cisco':
['show','vlan','brief']},
            'lacp'      :{'p': {'aruba': ['show','lacp','peer'],'comware5': ['display','link-aggregation','member-
port'],'comware7': ['display','link-aggregation','member-port'],'cisco': ['show','interface','etherchannel']},
                        'aruba': ['show','lacp'],'comware5': ['display','link-aggregation','summary'],'comware7':
['display','link-aggregation','summary'],'cisco': ['show','etherchannel','summary']},
            'stp'       :{'i': {'aruba': ['show','spanning-tree','instance', None],'comware5': ['display','stp','instance',
None],'comware7': ['display','stp','instance', None],'cisco': ['show','spanning-tree','mst', None]},
                        'r': {'aruba': ['show','spanning-tree','mst-config'],'comware5': ['display','stp','region-
configuration'],'comware7': ['display','stp','region-configuration'],'cisco': ['show','spanning-tree','mst','configuration']},
                        'D': {'aruba': ['show','link-keepalive','statistics'],'comware5': ['display','dldp','statistics'],'comware7':
['display','dldp','statistics'],'cisco': ['!# command not supported']},
                        'd': {'aruba': ['show','link-keepalive'],'comware5': ['display','dldp'],'comware7':
['display','dldp'],'cisco': ['show','udld', None]},
                        'aruba': ['show','spanning-tree'],'comware5': ['display','stp'],'comware7': ['display','stp'],'cisco':
['show','spanning-tree']},
            'I'         :{'b': {'aruba': ['show','interfaces','brief', None],'comware5':
['display','interface','1/0/','brief'],'comware7': ['display','interface', '1/0/','brief'],'cisco':
['show','interfaces','0/','status']},'aruba': ['show','interfaces', None],'comware5': ['display','interface', '1/0/'],'comware7':
['display','interface', '1/0/'],'cisco': ['show','interfaces', '0/']},
            'aruba': ['show'],'comware5': ['display'],'comware7': ['display'],'cisco': ['show']
            }

###################### config modes commands ##########################

neutral_modes = {'enable'   : {'aruba': ['enable'],'comware5': ['system-view'],'comware7': ['system-view'],'cisco':
['enable']},
             'config'   : {'aruba': ['configure'],'comware5': ['system-view'],'comware7': ['system-view'],'cisco':
['configure','terminal']},
             'vlan'     : {'n': {'aruba': ['vlan', None],'comware5': ['vlan', None],'comware7': ['vlan', None],'cisco': ['vlan',
None]}},
             'Interface': {'vty': {'aruba': ['# command not supported'],'comware5': ['user-interface','vty', None,
None],'comware7': ['user-interface','vty', None, None],'cisco': ['line','vty', None, None]},
                        'r' : {'aruba': ['# command not supported'],'comware5': ['# command not supported'],'comware7':
['interface','range','1/0/','to','1/0/'],'cisco': ['interface','range','0/','-', None]},
                        'vlan': {'n': {'aruba': ['vlan', None],'comware5': ['interface','Vlan-interface', None],'comware7':
['interface','Vlan-interface', None],'cisco': ['interface','vlan', None]}},
                        'aggregation':  {'aruba': ['trunk', None,'-',None,'trk','lacp'],'comware5': ['interface','Bridge-
Aggregation', None],'comware7': ['interface','Bridge-Aggregation', None],'cisco': ['interface','port-channel', None]},
                        'stp': {'r': {'aruba': ['# command not supported'],'comware5': ['stp','region-
configuration'],'comware7': ['stp','region-configuration'],'cisco': ['spanning-tree','mst','configuration']}},
                        'acl': {'s': {'n': {'aruba': ['ip','access-list','standard', None],'comware5': ['acl','number', None,'name',
None],'comware7': ['acl','number', None,'name', None],'cisco': ['ip','access-list','standard', None]},'aruba': ['ip','access-
list','standard', None],'comware5': ['acl','number', None],'comware7': ['acl','number', None],'cisco': ['ip','access-
list','standard', None]},
```

```
                                'e': {'n': {'aruba': ['ip','access-list','extended', None],'comware5': ['acl','number', None,'name',
None],'comware7': ['acl','number', None,'name', None],'cisco': ['ip','access-list','extended', None]},'aruba': ['ip','access-
list','extended', None],'comware5': ['acl','number', None],'comware7': ['acl','number', None],'cisco': ['ip','access-
list','extended', None]}},
                          'aruba': ['interface', None],'comware5': ['interface','1/0/'],'comware7': ['interface','1/0/'],'cisco':
['interface','0/']},
               'exit'     : {'aruba': ['exit'],'comware5': ['quit'],'comware7': ['quit'],'cisco': ['exit']}
              }

######################## config commands ##########################

neutral_config_C = {'hostname'  :{'aruba': ['hostname', None],'comware5': ['sysname', None],'comware7': ['sysname',
None],'cisco': ['hostname', None]},
                    'domain'    :{'aruba': ['# command not supported'],'comware5': ['domain', None],'comware7': ['domain',
None],'cisco': ['ip','domain-name', None]},
                    'ssh'       :{'c':{'gen':{'aruba': ['crypto','key','generate','ssh'],'comware5': ['public-
key','local','create','rsa'],'comware7': ['public-key','local','create','rsa'],'cisco': ['crypto','key','generate']}},
                                  'en':{'aruba': ['ip','ssh'],'comware5': ['ssh','server','enable'],'comware7':
['ssh','server','enable'],'cisco': ['ip','ssh','version','2']}},
                    'lldp'      :{'en': {'aruba': ['lldp','run'],'comware5': ['lldp','enable'],'comware7':
['lldp','global','enable'],'cisco': ['lldp','run']}},
                    'ip'        :{'route': {'aruba': ['ip','route', None, None, None],'comware5': ['ip','route-static', None, None,
None],'comware7': ['ip','route-static', None, None, None],'cisco': ['ip','route', None, None, None]}},
                    'stp'       :{'en': {'aruba': ['spanning-tree'],'comware5': ['stp','enable'],'comware7': ['stp','enable'],'cisco':
['spanning-tree','mode','mst']},
                                  'pathcost': {'aruba': ['spanning-tree','pathcost','mstp', None],'comware5': ['stp','pathcost-standard',
None],'comware7': ['stp','pathcost-standard', None],'cisco': ['spanning-tree','pathcost','method', None]},
                                  'p': {'aruba': ['spanning-tree','priority', None],'comware5': ['stp','priority', None],'comware7':
['stp','priority', None],'cisco': ['spanning-tree','mst','0','priority', None]},
                                  'i': {'p':{'aruba': ['spanning-tree','instance', None,'priority', None],'comware5': ['stp','instance',
None,'priority', None],'comware7': ['stp','instance', None,'priority', None],'cisco': ['spanning-tree','mst', None,'priority',
None]}}}
                   }

neutral_config_VTYIC = {'login'     :{'m': {'aruba': ['# command not supported'],'comware5': ['authentication-mode',
None],'comware7': ['authentication-mode', None],'cisco': ['login', None]}},
                        'inbound'   :{'p': {'aruba': ['# command not supported'],'comware5': ['protocol','inbound',
None],'comware7': ['protocol','inbound', None],'cisco': ['transport','input', None]}}
                       }

neutral_config_IC = {'description'     :{'aruba': ['name', None],'comware5': ['description', None],'comware7':
['description', None],'cisco': ['description', None]},
                     'duplex'          :{'aruba': ['speed-duplex', None],'comware5': ['duplex', None],'comware7': ['duplex',
None],'cisco': ['duplex', None]},
                     'speed'           :{'aruba': ['speed-duplex', None],'comware5': ['speed', None],'comware7': ['speed',
None],'cisco': ['speed', None]},
                     'en'              :{'aruba': ['enable'],'comware5': ['undo','shutdown'],'comware7': ['undo','shutdown'],'cisco':
['no','shutdown']},
                     'dis'             :{'aruba': ['disable'],'comware5': ['shutdown'],'comware7': ['shutdown'],'cisco': ['shutdown']},
                     'trunk'           :{'aruba': [['vlan', None],['tagged', None]],'comware5': [['port','link-
type','trunk'],['port','trunk','permit','vlan', None]],'comware7': [['port','link-type','trunk'],['port','trunk','permit','vlan',
None]],'cisco': [['switchport','trunk','encapsulation','dot1q'],['switchport','trunk','allowed','vlan', None]]},
                     'access'          :{'aruba': [['vlan', None],['untagged', None]],'comware5': [['port','link-
type','access'],['port','access','vlan', None]],'comware7': [['port','link-type','access'],['port','access','vlan', None]],'cisco':
[['switchport','mode','access'],['switchport','access','vlan', None]]},
                     'hybrid'          :{'tagged': {'aruba': [['vlan', None],['tagged', None]],'comware5': [['port','link-
type','hybrid'],['port','hybrid','vlan', None,'tagged']],'comware7': [['port','link-type','hybrid'],['port','hybrid','vlan',
None,'tagged']],'cisco': [['switchport','mode','access'],['switchport','voice','vlan', None]]},
                                       'untagged': {'aruba': [['vlan', None],['untagged', None]],'comware5': [['port','link-
type','hybrid'],['port','hybrid','vlan', None,'untagged']],'comware7': [['port','link-type','hybrid'],['port','hybrid','vlan',
None,'untagged']],'cisco': [['switchport','mode','access'],['switchport','access','vlan', None]]}},
                     'aggregation'     :{'n': {'aruba': ['# command not supported'],'comware5': ['port','link-aggregation','group',
None],'comware7': ['port','link-aggregation','group', None],'cisco': ['channel-group', None,'mode','active']}},
```

```
                'stp'           :{'dldp': {'aruba': ['link-keepalive'],'comware5': ['dldp','enable'],'comware7':
['dldp','enable'],'cisco': ['udld','port']},
                                'edge-port': {'aruba': ['spanning-tree', None,'admin-edge-port'],'comware5': ['stp','edge-
port','enable'],'comware7': ['stp','edge-port'],'cisco': ['spanning-tree','portfast']},
                                'bpdu-guard': {'aruba': ['spanning-tree', None,'bpdu-protection'],'comware5': ['stp','bpdu-
protection'],'comware7': ['stp','bpdu-protection'],'cisco': ['spanning-tree','bpduguard','enable']},
                                'root-guard': {'aruba': ['spanning-tree', None,'root-guard'],'comware5': ['stp','root-
protection'],'comware7': ['stp','root-protection'],'cisco': ['spanning-tree','guard','root']},
                                'loop-guard': {'aruba': ['loop-protect', None,'receiver-action','send-disable'],'comware5':
['loopback-detection','enable'],'comware7': ['loopback-detection','enable','vlan','all'],'cisco': ['spanning-
tree','guard','loop']},
                                'c': {'aruba': ['spanning-tree', None,'path-cost', None],'comware5': ['stp','cost',
None],'comware7': ['stp','cost', None],'cisco': ['spanning-tree','cost', None]},
                                'p': {'aruba': ['spanning-tree', None,'priority', None],'comware5': ['stp','port','priority',
None],'comware7': ['stp','port','priority', None],'cisco': ['spanning-tree','port-priority', None]},
                                'i': {'c':{'aruba': ['spanning-tree','instance', None, None,'path-cost', None],'comware5':
['stp','instance', None,'cost', None],'comware7': ['stp','instance', None,'cost', None],'cisco': ['spanning-tree','mst',
None,'cost', None]},
                                        'p':{'aruba': ['spanning-tree','instance', None, None,'priority', None],'comware5':
['stp','instance', None,'port','priority', None],'comware7': ['stp','instance', None,'port','priority', None],'cisco':
['spanning-tree','mst', None,'port-priority', None]}}},
                'acl': {'n': {'aruba': ['ip','access-group', None, None],'comware5': ['packet-filter', None,
None],'comware7': ['packet-filter', None, None],'cisco': ['ip','access-group', None, None]},
                        'aruba': ['ip','access-group', None, None],'comware5': ['packet-filter', None, None],'comware7':
['packet-filter', None, None],'cisco': ['ip','access-group', None, None]}
                }

neutral_config_VIC = {'name'            :{'aruba': ['name', None],'comware5': ['name', None],'comware7': ['name',
None],'cisco': ['name', None]}
                }

neutral_config_VVIC = {'ip' : {'address': {'aruba': ['ip','address', None, None],'comware5': ['ip','address', None,
None],'comware7': ['ip','address', None, None],'cisco': ['ip','address', None, None]}},
                'acl': {'n': {'aruba': ['ip','access-group', None, None],'comware5': ['packet-filter', None,
None],'comware7': ['packet-filter', None, None],'cisco': ['ip','access-group', None, None]},
                        'aruba': ['ip','access-group', None, None],'comware5': ['packet-filter', None, None],'comware7':
['packet-filter', None, None],'cisco': ['ip','access-group', None, None]}
                }

neutral_config_BAIC = {'trunk'          :{'aruba': ['vlan', None,'tagged','trk'],'comware5': [['port','link-
type','trunk'],['port','trunk','permit','vlan', None]],'comware7': [['port','link-type','trunk'],['port','trunk','permit','vlan',
None]],'cisco': [['switchport','trunk','encapsulation','dot1q'],['switchport','trunk','allowed','vlan', None]]},
                'access'        :{'aruba': ['vlan', None,'untagged','trk'],'comware5': [['port','link-
type','access'],['port','access','vlan', None]],'comware7': [['port','link-type','access'],['port','access','vlan', None]],'cisco':
[['switchport','mode','access'],['switchport','access','vlan', None]]}
                }

neutral_config_IRIC = {'aggregation'    :{'n': {'aruba': ['# command not supported'],'comware5': ['# command not
supported'],'comware7': ['port','link-aggregation','group', None],'cisco': ['channel-group', None,'mode','active']}},
                'trunk'         :{'aruba': ['# command not supported'],'comware5': ['# command not
supported'],'comware7': [['port','link-type','trunk'],['port','trunk','permit','vlan', None]],'cisco':
[['switchport','trunk','encapsulation','dot1q'],['switchport','trunk','allowed','vlan', None]]},
                'access'        :{'aruba': ['# command not supported'],'comware5': ['# command not
supported'],'comware7': [['port','link-type','access'],['port','access','vlan', None]],'cisco':
[['switchport','mode','access'],['switchport','access','vlan', None]]},
                'hybrid'        :{'tagged': {'aruba': ['# command not supported'],'comware5': ['# command not
supported'],'comware7': [['port','link-type','hybrid'],['port','hybrid','vlan', None,'tagged']],'cisco':
[['switchport','mode','access'],['switchport','voice','vlan', None]]},
                                'untagged': {'aruba': ['# command not supported'],'comware5': ['# command not
supported'],'comware7': [['port','link-type','hybrid'],['port','hybrid','vlan', None,'untagged']],'cisco':
[['switchport','mode','access'],['switchport','access','vlan', None]]}}
                }
```

```
neutral_config_STPRIC = {'region'       :{'n': {'aruba': ['spanning-tree','config-name', None],'comware5': ['region-
name', None],'comware7': ['region-name', None],'cisco': ['name', None]},
                                'r': {'aruba': ['spanning-tree','config-revision', None],'comware5': ['revision-level',
None],'comware7': ['revision-level', None],'cisco': ['revision', None]}},
                    'instance'     :{'aruba': ['spanning-tree','instance', None,'vlan', None],'comware5': ['instance',
None,'vlan', None],'comware7': ['instance', None,'vlan', None],'cisco': ['instance', None,'vlan', None]}
                    }

neutral_config_SNACLIC = {'permit'      :{'i': {'w': {'aruba': ['permit', None, None],'comware5':
['rule','permit','source', None, None],'comware7': ['rule','permit','source', None, None],'cisco': ['permit', None,
None]},'aruba': ['permit','ip','any','any'],'comware5': ['#command not required since HPE has an implicit allow at the
bottom'],'comware7': ['#command not required since HPE has an implicit allow at the bottom'],'cisco':
['permit','ip','any','any']}},
                     'deny'       :{'aruba': ['deny','ip', None, None, None, None],'comware5': ['rule','deny','ip','source',
None, None,'destination', None, None],'comware7': ['rule','deny','ip','source', None, None,'destination', None,
None],'cisco': ['deny','ip', None, None, None, None]}
                    }

neutral_config_SNAACLIC = {'permit'      :{'i': {'w': {'aruba': ['permit', None, None],'comware5':
['rule','permit','source', None, None],'comware7': ['rule','permit','source', None, None],'cisco': ['permit', None,
None]},'aruba': ['permit','ip','any','any'],'comware5': ['#command not required since HPE has an implicit allow at the
bottom'],'comware7': ['#command not required since HPE has an implicit allow at the bottom'],'cisco':
['permit','ip','any','any']}},
                     'deny'       :{'aruba': ['deny','ip', None, None, None, None],'comware5': ['rule','deny','ip','source',
None, None,'destination', None, None],'comware7': ['rule','deny','ip','source', None, None,'destination', None,
None],'cisco': ['deny','ip', None, None, None, None]}
                    }

neutral_config_ENACLIC = {'permit'      :{'i': {'w': {'aruba': ['permit', None, None],'comware5':
['rule','permit','source', None, None],'comware7': ['rule','permit','source', None, None],'cisco': ['permit', None,
None]},'aruba': ['permit','ip','any','any'],'comware5': ['#command not required since HPE has an implicit allow at the
bottom'],'comware7': ['#command not required since HPE has an implicit allow at the bottom'],'cisco':
['permit','ip','any','any']}},
                     'deny'       :{'aruba': ['deny','ip', None, None, None, None],'comware5': ['rule','deny','ip','source',
None, None,'destination', None, None],'comware7': ['rule','deny','ip','source', None, None,'destination', None,
None],'cisco': ['deny','ip', None, None, None, None]}
                    }

neutral_config_ENAACLIC = {'permit'      :{'i': {'w': {'aruba': ['permit', None, None],'comware5':
['rule','permit','source', None, None],'comware7': ['rule','permit','source', None, None],'cisco': ['permit', None,
None]},'aruba': ['permit','ip','any','any'],'comware5': ['#command not required since HPE has an implicit allow at the
bottom'],'comware7': ['#command not required since HPE has an implicit allow at the bottom'],'cisco':
['permit','ip','any','any']}},
                     'deny'       :{'aruba': ['deny','ip', None, None, None, None],'comware5': ['rule','deny','ip','source',
None, None,'destination', None, None],'comware7': ['rule','deny','ip','source', None, None,'destination', None,
None],'cisco': ['deny','ip', None, None, None, None]}
                    }

###################### Netmico Connection ############################################

# print API description
print('#################################################################################################
######')
print('######## Command Converter API converts Vendor Neutral CLI Commands to following CLIs #########')
print('########              Cisco | Aruba | HPE Comware 5 | HPE Comware 7              #########')
print('#################################################################################################
######')

# get ip address of the switch to connect
ipaddr_tmp = input("Enter the switch ip address: ")
# regular expression to validate ip address
regex = '''^(25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(
        25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(
        25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(
```

```
        25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)'"

# validate ip address
if (re.search(regex, ipaddr_tmp)):
    ipaddr = ipaddr_tmp
else:
    print("Please enter a valid ip address")
    exit()

# get the username of the switch to connect
user = input("Enter the switch username: ")

# get the convertion type
convertion_type = input("Enter convertion type [ cisco | aruba | comware5 | comware7 ]: ")

if convertion_type == 'aruba':
  NEUTRAL_STATE = 'P'
  dtype = "hp_procurve"
elif convertion_type == 'cisco':
  NEUTRAL_STATE = 'P'
  #SYS_COUNT = 1
  dtype = "cisco_ios"
elif convertion_type == 'comware5':
  NEUTRAL_STATE = 'U'
  device_model = input("Enter switch model [ 1910 | 1920 | 5500 ] : ")
  if device_model == '1910':
    dtype = "hp_comware512900"
  elif device_model == '1920':
    dtype = "hp_comwarejinhua"
  elif device_model == '5500':
    dtype = "hp_comware"
  else:
    print("Please enter a valid switch model [ 1910 | 1920 | 5500 ]")
    exit()
elif convertion_type == 'comware7':
  NEUTRAL_STATE = 'U'
  device_model = input("Enter switch model [ 1950 | 5510 ] : ")
  if device_model == '1950':
    dtype = "hp_comwarefoes"
  elif device_model == '5510':
    dtype = "hp_comware"
  else:
    print("Please enter a valid switch model [ 1950 | 5510 ]")
    exit()
else:
  print("Invalid convertion type [ cisco | aruba | comware5 | comware7 ]")
  exit()

'"
ipaddr = "192.168.50.102"
dtype = "hp_comwarejinhua"
convertion_type = "comware5"
user = "admin"
'"

SWITCH = {
    "ip": ipaddr,
    "username": user,
    "password": getpass(),
    "device_type": dtype,
}

net_connect = Netmiko(**SWITCH)  # initiate an ssh session with the switch
```

```
#convertion_type = input("Enter convertion type [ aruba | comware5 | comware7 | cisco ]: ")
while EXIT == False:
    #input_string = input("Enter the Vendor Neutral command: ")
    #show current prompt
    print(net_connect.find_prompt(), end = '')
    # get command input
    input_string = input("")

    if input_string == 'e':
        EXIT = True
        net_connect.disconnect()

    res = len(input_string.split())
    num = int(res)

    # local Variables
    command_list = []
    command = []
    cfg_command = ''
    output = ''
    seperator = ' '

    for i in range(num):
        command_list.append(input_string.split(" ")[i])

    #print(command_list)


    # blank input
    if len(command_list) == 0:
      print('#Invalid input')
    # state
    elif command_list[0] == 'configure' and len(command_list) == 1:
      print('#Current state: '+NEUTRAL_STATE)
    # en | enable
    elif (command_list[0] == 'en' or command_list[0] == 'enable') and len(command_list) == 1 and
NEUTRAL_STATE == 'U':
      if convertion_type == 'aruba':
        #print(seperator.join(neutral_modes[command_list[0][:2]][convertion_type]))
        net_connect.enable()
        NEUTRAL_STATE = 'P'
      elif convertion_type == 'comware5':
        #print(seperator.join(neutral_modes[command_list[0][:2]][convertion_type]))
        if net_connect.check_enable_mode():
          print('#Already in enable mode')
          NEUTRAL_STATE = 'P'
        else:
          net_connect.enable()
          NEUTRAL_STATE = 'P'
      elif convertion_type == 'comware7':
        #print(seperator.join(neutral_modes[command_list[0][:2]][convertion_type]))
        if net_connect.check_enable_mode():
          print('#Already in enable mode')
          NEUTRAL_STATE = 'P'
        else:
          net_connect.enable()
          NEUTRAL_STATE = 'P'
      elif convertion_type == 'cisco':
        net_connect.write_channel(seperator.join(neutral_modes[command_list[0][:2]][convertion_type]))
        NEUTRAL_STATE = 'P'
    # config | configure
```

```
    elif (command_list[0] == 'config' or command_list[0] == 'configure') and len(command_list) == 1 and
NEUTRAL_STATE == 'P':
      if convertion_type == 'aruba':
        net_connect.write_channel(seperator.join(neutral_modes[command_list[0][0:6]][convertion_type]))
        NEUTRAL_STATE = 'C'
      elif convertion_type == 'comware5':
        #print(seperator.join(neutral_modes[command_list[0][0:6]][convertion_type]))
        if net_connect.check_config_mode():
          print('#Already in config mode')
          NEUTRAL_STATE = 'C'
        else:
          net_connect.config_mode()
          NEUTRAL_STATE = 'C'
      elif convertion_type == 'comware7':
        #print(seperator.join(neutral_modes[command_list[0][0:6]][convertion_type]))
        if net_connect.check_config_mode():
          print('#Already in config mode')
          NEUTRAL_STATE = 'C'
        else:
          net_connect.config_mode()
          NEUTRAL_STATE = 'C'
      elif convertion_type == 'cisco':
        net_connect.write_channel(seperator.join(neutral_modes[command_list[0][0:6]][convertion_type]))
        NEUTRAL_STATE = 'C'
    # exit
    elif command_list[0] == 'exit' and len(command_list) == 1:
      if NEUTRAL_STATE == 'STPRIC' or NEUTRAL_STATE == 'IRIC' or NEUTRAL_STATE == 'VVIC' or
NEUTRAL_STATE == 'VIC' or NEUTRAL_STATE == 'VTYIC':
        net_connect.write_channel(seperator.join(neutral_modes[command_list[0]][convertion_type]))
        NEUTRAL_STATE = 'C'
      elif NEUTRAL_STATE == 'BAIC':
        net_connect.write_channel(seperator.join(neutral_modes[command_list[0]][convertion_type]))
        BRIDGE_AGGREGATION_NUMBER = 0
      elif NEUTRAL_STATE == 'IC':
        net_connect.write_channel(seperator.join(neutral_modes[command_list[0]][convertion_type]))
        NEUTRAL_STATE = 'C'
        INTERFACE_NUMBER = 0
      elif NEUTRAL_STATE == 'SNACLIC' or NEUTRAL_STATE == 'SNAACLIC' or NEUTRAL_STATE ==
'ENACLIC' or NEUTRAL_STATE == 'ENAACLIC':
        net_connect.write_channel(seperator.join(neutral_modes[command_list[0]][convertion_type]))
        NEUTRAL_STATE = 'C'
      elif NEUTRAL_STATE == 'C':
        if convertion_type == 'aruba':
          net_connect.write_channel(seperator.join(neutral_modes[command_list[0]][convertion_type]))
          NEUTRAL_STATE = 'P'
        elif convertion_type == 'comware5':
          #print(seperator.join(neutral_modes[command_list[0]][convertion_type]))
          if net_connect.check_config_mode():
            net_connect.exit_config_mode()
            NEUTRAL_STATE = 'U'
          else:
            print('#Not in config mode')
        elif convertion_type == 'comware7':
          #print(seperator.join(neutral_modes[command_list[0]][convertion_type]))
          if net_connect.check_config_mode():
            net_connect.exit_config_mode()
            NEUTRAL_STATE = 'U'
          else:
            print('#Not in config mode')
        elif convertion_type == 'cisco':
          net_connect.write_channel(seperator.join(neutral_modes[command_list[0]][convertion_type]))
          NEUTRAL_STATE = 'P'
      elif NEUTRAL_STATE == 'P':
        if convertion_type == 'aruba':
```

```
            net_connect.write_channel(seperator.join(neutral_modes[command_list[0]][convertion_type]))
            NEUTRAL_STATE = 'U'
          elif convertion_type == 'comware5':
            #print(seperator.join(neutral_modes[command_list[0]][convertion_type]))
            if net_connect.check_enable_mode():
              net_connect.exit_enable_mode()
              NEUTRAL_STATE = 'U'
            else:
              print('#Not in enable mode')
          elif convertion_type == 'comware7':
            #print(seperator.join(neutral_modes[command_list[0]][convertion_type]))
            if net_connect.check_enable_mode():
              net_connect.exit_enable_mode()
              NEUTRAL_STATE = 'U'
            else:
              print('#Not in enable mode')
          elif convertion_type == 'cisco':
            net_connect.write_channel(seperator.join(neutral_modes[command_list[0]][convertion_type]))
            NEUTRAL_STATE = 'U'
      # save
      elif command_list[0] == 'save' and len(command_list) == 1 and NEUTRAL_STATE == 'P':
        if convertion_type == 'aruba':
          #print('save')
          net_connect.save_config()
        elif convertion_type == 'comware5':
          #print('save force')
          net_connect.save_config()
        elif convertion_type == 'comware7':
          #print('save force')
          net_connect.save_config()
        elif convertion_type == 'cisco':
          #print('copy run start')
          net_connect.save_config()
      # dis | disable
      elif (command_list[0] == 'dis' or command_list[0] == 'disable') and len(command_list) == 1 and
NEUTRAL_STATE == 'IC':
        net_connect.write_channel(neutral_config_IC[command_list[0][:3]][convertion_type][0])
      # en | enable
      elif (command_list[0] == 'en' or command_list[0] == 'enable') and len(command_list) == 1 and
NEUTRAL_STATE == 'IC':
        net_connect.write_channel(seperator.join(neutral_config_IC[command_list[0][:2]][convertion_type]))
#     # show
      if command_list[0] == 'show' and len(command_list) == 1 and NEUTRAL_STATE == 'P':
        if convertion_type == 'aruba':
          output = net_connect.send_command(seperator.join(neutral_show[convertion_type]))
          print(output)
        elif convertion_type == 'comware5':
          output = net_connect.send_command(seperator.join(neutral_show[convertion_type]))
          print(output)
        elif convertion_type == 'comware7':
          output = net_connect.send_command(seperator.join(neutral_show[convertion_type]))
          print(output)
        elif convertion_type == 'cisco':
          output = net_connect.send_command(seperator.join(neutral_show[convertion_type]))
          print(output)
#     ## show
      elif command_list[0] == 'show' and len(command_list) == 2 and (command_list[1] != 'ssh' or command_list[1] !=
'lacp' or command_list[1] != 'vlan' or command_list[1] != 'stp') and NEUTRAL_STATE == 'P':
        command = commandfilter(1,len(command_list),command_list)
        # show -f | show --flash
        if command[0] == 'f' or command[0] == 'flash':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -v | show --version
```

```
        elif command[0] == 'v' or command[0] == 'version':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -s | show --sysinfo
        elif command[0] == 's' or command[0] == 'sysinfo':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -m | show --modules
        elif command[0] == 'm' or command[0] == 'modules':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -R | show --Running-config
        elif command[0] == 'R' or command[0] == 'Running-config':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -S | show --Startup-config
        elif command[0] == 'S' or command[0] == 'Startup-config':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -h | show --history
        elif command[0] == 'h' or command[0] == 'history':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -l | show --logging
        elif command[0] == 'l' or command[0] == 'logging':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -t | show --tech-support
        elif command[0] == 't' or command[0] == 'tech-support':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -F | show --Fan
        elif command[0] == 'F' or command[0] == 'Fan':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -P | show --Power
        elif command[0] == 'P' or command[0] == 'Power':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -T | show --Temperature
        elif command[0] == 'T' or command[0] == 'Temperature':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -u | show --users
        elif command[0] == 'u' or command[0] == 'users':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
        # show -i | show --interface-brief
        elif command[0] == 'i' or command[0] == 'interface-brief':
          output = net_connect.send_command(seperator.join(neutral_show[command[0][0]][convertion_type]))
          print(output)
    #    ## show
       elif command_list[0] == 'show' and len(command_list) == 2 and (command_list[1] == 'ssh' or command_list[1] ==
    'lacp' or command_list[1] == 'vlan' or command_list[1] == 'stp') and NEUTRAL_STATE == 'P':
        command = commandfilter(1,len(command_list),command_list)
        # show ssh
        if command[0] == 'ssh':
          output = net_connect.send_command(seperator.join(neutral_show[command[0]][convertion_type]))
          print(output)
        # show vlan
        elif command[0] == 'vlan':
          output = net_connect.send_command(seperator.join(neutral_show[command[0]][convertion_type]))
          print(output)
        # show lacp
```

```
      elif command[0] == 'lacp':
        output = net_connect.send_command(seperator.join(neutral_show[command[0]][convertion_type]))
        print(output)
      # show stp
      elif command[0] == 'stp':
        output = net_connect.send_command(seperator.join(neutral_show[command[0]][convertion_type]))
        print(output)
    # stp en | stp enable
    elif command_list[0] == 'stp' and (command_list[1] == 'en' or command_list[1] == 'enable') and
len(command_list) == 2 and NEUTRAL_STATE == 'C':

net_connect.write_channel(seperator.join(neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type
]))
    # name [DESCRIPTION STRING]
    elif command_list[0] == 'name' and command_list[-1] != 'voice' and len(command_list) == 2 and
NEUTRAL_STATE == 'VIC':
      net_connect.write_channel(neutral_config_VIC[command_list[0]][convertion_type][0]+' '+command_list[-1])
    # name voice
    elif command_list[0] == 'name' and command_list[-1] == 'voice' and len(command_list) == 2 and
NEUTRAL_STATE == 'VIC':
      if convertion_type == 'aruba':
        net_connect.write_channel('voice')
      elif convertion_type != 'aruba':
        net_connect.write_channel(neutral_config_VIC[command_list[0]][convertion_type][0]+' '+command_list[-1])
    # description [DESCRIPTION STRING]
    elif command_list[0] == 'description' and len(command_list) == 2 and NEUTRAL_STATE == 'IC':
      net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+command_list[-1])
    # duplex [ auto | full | half ]
    elif command_list[0] == 'duplex' and len(command_list) == 2 and NEUTRAL_STATE == 'IC':
      speed_duplex = command_list[-1]
      if convertion_type == 'aruba':
        if speed_duplex == 'auto':
          net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
        elif speed_duplex == 'full':
          speed = input("Enter speed [10 | 100 | 1000 | 10000]:")
          speed_duplex = speed +'-'+ speed_duplex
          net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
        elif speed_duplex == 'half':
          speed = input("Enter speed [10 | 100]:")
          speed_duplex = speed +'-'+ speed_duplex
          net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
      elif convertion_type == 'comware5':
        net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
      elif convertion_type == 'comware7':
        net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
      elif convertion_type == 'cisco':
        net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
    # speed [ 10 | 100 | 1000 | 10000 | auto ]
    elif command_list[0] == 'speed' and len(command_list) == 2 and NEUTRAL_STATE == 'IC':
      speed_duplex = command_list[-1]
      if convertion_type == 'aruba':
        if speed_duplex == 'auto':
          net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
        elif speed_duplex == '10':
          duplex = input("Enter duplex [full | half]:")
          speed_duplex = speed_duplex +'-'+ duplex
          net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
        elif speed_duplex == '100':
          duplex = input("Enter duplex [full | half]:")
          speed_duplex = speed_duplex +'-'+ duplex
          net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
        elif speed_duplex == '1000':
          duplex = input("Enter duplex [full]:")
          speed_duplex = speed_duplex +'-'+ duplex
```

```
        net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
       elif speed_duplex == '10000':
        duplex = input("Enter duplex [full]:")
        speed_duplex = speed_duplex +'-'+ duplex
        net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
      elif convertion_type == 'comware5':
       net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
      elif convertion_type == 'comware7':
       net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
      elif convertion_type == 'cisco':
       net_connect.write_channel(neutral_config_IC[command_list[0]][convertion_type][0]+' '+speed_duplex)
    # hostname [HOST NAME STRING]
    elif command_list[0] == 'hostname' and len(command_list) == 2 and NEUTRAL_STATE == 'C':
     net_connect.write_channel(neutral_config_C[command_list[0]][convertion_type][0]+' '+command_list[-1])
    # lldp en | lldp enable
    elif command_list[0] == 'lldp' and (command_list[1] == 'en' or command_list[1] == 'enable') and
len(command_list) == 2 and NEUTRAL_STATE == 'C':
      if convertion_type == 'aruba':
       net_connect.write_channel(neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][0]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][1])
      elif convertion_type == 'comware5':
       net_connect.write_channel(neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][0]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][1])
      elif convertion_type == 'comware7':
       net_connect.write_channel(neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][0]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][1]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][2])
      elif convertion_type == 'cisco':
       net_connect.write_channel(neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][0]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][1])
    # domain [DOMAIN NAME STRING]
    elif command_list[0] == 'domain' and len(command_list) == 2 and NEUTRAL_STATE == 'C':
      if convertion_type == 'aruba':
       #net_connect.write_channel(neutral_config_C[command_list[0]][convertion_type][0])
       print('#Command not supported')
      elif convertion_type == 'comware5':
       net_connect.write_channel(neutral_config_C[command_list[0]][convertion_type][0]+' '+command_list[-1])
      elif convertion_type == 'comware7':
       net_connect.write_channel(neutral_config_C[command_list[0]][convertion_type][0]+' '+command_list[-1])
      elif convertion_type == 'cisco':
       net_connect.write_channel(neutral_config_C[command_list[0]][convertion_type][0]+'
'+neutral_config_C[command_list[0]][convertion_type][1]+' '+command_list[-1])
    # ssh en | ssh enable
    elif command_list[0] == 'ssh' and (command_list[1] == 'en' or command_list[1] == 'enable') and
len(command_list) == 2 and NEUTRAL_STATE == 'C':
      if convertion_type == 'aruba':
       net_connect.write_channel(neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][0]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][1])
      elif convertion_type == 'comware5':
       net_connect.write_channel(neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][0]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][1]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][2])
      elif convertion_type == 'comware7':
       net_connect.write_channel(neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][0]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][1]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][2])
      elif convertion_type == 'cisco':
       net_connect.write_channel(neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][0]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][1]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][2]+'
'+neutral_config_C[command_list[0]][command_list[1][:2]][convertion_type][3])

##########################################################################################
```

Appendix F

############### General commands ##
NAMES
enable -- allows user to enter in to priviledge mode of the switch
configure -- allows user to save configurations to the switch.
exit -- allows user to enter in to configuration mode of the switch.
save -- allows user to to exit from current mode of configuration and to go to the previous mode.
state -- allows user to view current mode or state of configuration.
e -- allows user to terminate the ssh connection with the switch and to exit from the program.
SYNOPSIS
en | enable
config | configure
exit
save
state
e
DESCRIPTION
enable command is used to enter in to priviledge mode of the switch.
save command is used to save configurations to the switch.
configure command is used to enter in to configuration mode of the switch.
exit command is used to exit from current mode of configuration and to go to the previous mode.
state command is used to view current mode or state of configuration.
e command is used to terminate the ssh connection with the switch and to exit from the program.
EXAMPLES OF VALID MODES
en | enable
config | configure
exit
save
state
e
############### Show commands ##
NAME
show -- allow users to view various details about switch's configurations
SYNOPSIS
show [-f flash] [-v version] [-s sysinfo] [-m modules] [-R Running-config] [-S Startup-config] [-h history] [-l logging] [-t tech-support] [-F Fan] [-P Power] [-T Temperature] [-u users] [-i interface-brief]
show [-I] [INTERFACE #] [-t] [INTERFACE TYPE]
show [-I] [INTERFACE #] [-t] [INTERFACE TYPE] [-b brief]
show ip [-r route] [-i interface-brief]
show ssh [-c crypto]
show lldp [-n neighbors]
show lldp [-n neighbors] [-I] [INTERFACE #] [-t] [INTERFACE TYPE]
show vlan
show vlan [VLAN #]
show lacp
show lacp [-p port]
show stp
show stp [-r region-configuration]
show stp -i [INSTANCE #]
show stp [-d dldp] [-D Dldp-statistics]
DESCRIPTION
show command is used with various options in priviledge mode to view many details about the switch configurations
The options are as follows:
-f --flash
-v --version
-s --sysinfo
-m --modules
-R --Running-config
-S --Startup-config
-i --interface-brief
-h --history
-l --logging
-t --tech-support

-F --Fan
-P --Power
-T --Temperature
-u --users
-I --Interface
-t --type
-b --brief
ip
-r --route
-i --interface-brief
ssh
-c --crypto
lldp
-n --neighbors
-I --Interface
-t --type
lacp
-p --port
stp
-r --region-configuration
-i --instance
-d --dldp
-D --Dldp-statistics

EXAMPLES OF VALID MODES

show
show -f | show --flash
show -v | show --version
show -s | show --sysinfo
show -m | show --modules
show -R | show --Running-config
show -S | show –Startup-config
show -h | show --history
show -l | show --logging
show -t | show --tech-support
show -F | show --Fan
show -P | show --Power
show -T | show --Temperature
show -u | show --users
show -i | show --interface-brief
show -I [ITNERFACE #] -t [e | f | g | T] -b | show --Interface [ITNERFACE #] --type [e | f | g | T] --brief
example: show -I 1 -t g -b
show --Interface 1 --type g --brief
show -I [ITNERFACE #] -t [e | f | g | T] | show --Interface [ITNERFACE #] --type [e | f | g | T]
example: show -I 1 -t g
show --Interface 1 --type g
show ip -r | show ip --route
show ip -i | show ip --interface-brief
show ssh
show ssh -c | show ssh --crypto
show lldp -n | show lldp –neighbors
show lldp -n -I [ITNERFACE #] -t [e | f | g | T] | show lldp –neighbors --Interface [ITNERFACE #] --type [e | f | g | T]
example: show lldp -n -I 3 -t g
show lldp –neighbors --Interface 3 --type g
show lacp
show lacp -p | show lacp --port
show stp
show stp -r | show stp --region-configuration
show stp -i [INSTANCE #] | show stp --instance [INSTANCE #]
example: show stp -i 0
show stp --instance 0
show stp -d | show stp --dldp
show stp -D | show stp --Dldp-statistics

############### Interrface commands ####################################

NAME

Interface -- allows users to go inside switch's specific configuration modes

SYNOPSIS

Interface vty [START #] [END #]
Interface [ITNERFACE #] [-t --type] [INTERFACE TYPE]
Interface [-r --range] [START ITNERFACE #] [END ITNERFACE #] [-t --type] [INTERFACE TYPE]
Interface vlan [-n --number] [VLAN #]
Interface aggregation [AGGREGATION#]
Interface stp [-r --region-configuration]
Interface acl [-s --standard] [ACL #]
Interface acl [-s --standard] [ACL #] [-n --name] [ACL NAME STRING]
Interface acl [-e --extended] [ACL #]
Interface acl [-e --extended] [ACL #] [-n --name] [ACL NAME STRING]

DESCRIPTION

Interface commands are used to go inside specific configuration modes of the switch
The options are as follows:
Interface:
vty [START #] [END #]
[ITNERFACE #] -t --type [INTERFACE TYPE]
-r --range [START ITNERFACE #] [END ITNERFACE #] -t --type [INTERFACE TYPE]
vlan -n --number [VLAN #]
aggregation [AGGREGATION#]
stp -r --region-configuration
acl -s --standard [ACL #]
acl -s --standard [ACL #] -n --name [ACL NAME STRING]
acl -e --extended [ACL #]
acl -e --extended [ACL #] -n --name [ACL NAME STRING]

EXAMPLES OF VALID MODES

Interface vty [START #] [END #]
Interface [ITNERFACE #] -t [e | f | g | T] | Interface [ITNERFACE #] --type [e | f | g | T]
example: Interface 3 -t g
Interface 3 --type g
Interface -r [START ITNERFACE #] [END ITNERFACE #] -t [e | f | g | T] | Interface --range [START ITNERFACE #] [END ITNERFACE #] --type [e | f | g | T]
example: Interface -r 4 8 -t g
Interface --range 4 8 --type g
Interface vlan -n [VLAN #] | Interface vlan --number [VLAN #]
example: Interface vlan -n 200
Interface vlan --number 200
Interface aggregation [AGGREGATION#]
Interface stp -r | Interface stp --region-configuration
Interface acl -s [ACL #] | Interface acl --standard [ACL #]
example: Interface acl -s 50
Interface acl --standard 50
Interface acl -s [ACL #] -n [ACL NAME STRING] | Interface acl --standard [ACL #] --name [ACL NAME STRING]
example: Interface acl -s 60 -n standard_acl
Interface acl --standard 60 --name standard_acl
Interface acl -e [ACL #] | Interface acl --extended [ACL #]
example: Interface acl -s 150
Interface acl --extended 150
Interface acl -e [ACL #] -n [ACL NAME STRING] | Interface acl --extended [ACL #] --name [ACL NAME STRING]
example: Interface acl -e 160 -n extended_acl
Interface acl --extended 160 --name extended_acl

################ SSH commands ##

NAMES

hostname -- set host name of the switch
domain -- set domain name
ssh -- set ssh of the switch
login -- set login mode
inbound -- set inbound protocol

SYNOPSIS

hostname [HOST NAME]

domain [DOMAIN NAME]
ssh [-c --crypto] [gen | generate]
ssh [en | enable]
login [-m --mode] [password | AAA]
inbound [-p --protocol] [ssh | telnet | all]

DESCRIPTION

These commands are used to do ssh configurations of the switch.
The options are as follows:
hostname:
[HOST NAME]
domain:
[DOMAIN NAME]
ssh:
-c --crypto [gen | generate]
[en | enable]
login:
-m --mode [password | AAA]
inbound:
-p --protocol [ssh | telnet | all]

EXAMPLES OF VALID MODES

hostname [HOST NAME]
example: hostname SW1
domain [DOMAIN NAME]
example: domain example.com
ssh -c gen | ssh –crypto generate
ssh en | ssh enable
login [-m --mode] [password | AAA] | login –mode [password | AAA]
example: login -m password
login –mode password
inbound -p [ssh | telnet | all] | inbound --protocol [ssh | telnet | all]
example: inbound -p ssh
inbound --protocol ssh

################ LLDP commands ##

NAMES

lldp -- set link layer discovery protocol

SYNOPSIS

lldp [en | enable]

DESCRIPTION

lldp command can be used to set LLDP in switch
The options are as follows:
[en | enable]

EXAMPLES OF VALID MODES

lldp en | lldp enable

################ LACP commands ##

NAMES

Interface -- allows user to go inside switch's configuration modes
trunk -- set the permited vlans for trunk port
access -- set the access vlan for access port
aggregation -- sets the aggregation number for a port or ports

SYNOPSIS

Interface aggregation [AGGREGATION#]
trunk [-p --permit] [-v --vlan] [VLAN #]
trunk [-p --permit] [-v --vlan] [VLAN # ..]
trunk [-p --permit] [-v --vlan] all
access [-v --vlan] [VLAN #]
Interface [ITNERFACE #] [-t --type] [INTERFACE TYPE]
Interface [-r --range] [START ITNERFACE #] [END ITNERFACE #] [-t --type] [e | f | g | T]
aggregation [-n --number] [AGGREGATION#]

DESCRIPTION

These commands are used to set link aggregation configuration on switch ports
The options are as follows:
Interface aggregation:
[AGGREGATION#]
trunk:

[-p --permit] [-v --vlan] [VLAN #]
[-p --permit] [-v --vlan] [VLAN # ..]
[-p --permit] [-v --vlan] all
access:
[-v --vlan] [VLAN #]
Interface:
[ITNERFACE #] [-t --type] [INTERFACE TYPE]
[-r --range] [START ITNERFACE #] [END ITNERFACE #] [-t --type] [e | f | g | T]
aggregation:
[-n --number] [AGGREGATION#]

EXAMPLES OF VALID MODES

Interface aggregation [AGGREGATION#]
example: Interface aggregation 2
trunk -p -v [VLAN # ..] | trunk --permit --vlan [VLAN # ..]
example: trunk -p -v 100 200 1000 5
trunk --permit --vlan 100 200 1000 5
access -v [VLAN #] | access --vlan [VLAN #]
example: access -v 100
access --vlan 100
Interface [ITNERFACE #] -t [e | f | g | T] | Interface [ITNERFACE #] --type [e | f | g | T]
example: Interface 2 -t g
Interface 2 --type g
Interface -r [START ITNERFACE #] [END ITNERFACE #] -t [e | f | g | T] | Interface --range [START ITNERFACE #] [END ITNERFACE #] --type [e | f | g | T]
example: Interface -r 4 8 -t g
Interface --range 4 8 --type g
aggregation -n [AGGREGATION#] | aggregation --number [AGGREGATION#]
example: aggregation -n 2
aggregation --number 2

############### VLAN Management commands ##############################

NAMES

vlan -- allows user create a vlan for a specific vlan number
name -- set vlan name

SYNOPSIS

vlan [-n --number] [VLAN #]
name [DESCRIPTION STRING] [-t --type] [voice | data]

DESCRIPTION

these commands aree used to create a specific vlan with a paticular vlan number and to name the said vlan
The options are as follows:
vlan:
[-n --number] [VLAN #]
name:
[DESCRIPTION STRING] [-t --type] [voice | data]

EXAMPLES OF VALID MODES

vlan -n [VLAN #] | vlan --number [VLAN #]
example: vlan -n 200
vlan --number 200
name [DESCRIPTION STRING] -t [voice | data] | name [DESCRIPTION STRING] --type [voice | data]
example: name hr_dept -t data
name hr_dept
name voice -t voice

############### VLAN Assignment commands ##############################

NAMES

Interface -- allows user to go inside switch's configuration modes
trunk -- set the permited vlans for trunk port
access -- set the access vlan for access port

SYNOPSIS

Interface [ITNERFACE #] [-t --type] [INTERFACE TYPE]
Interface [-r --range] [START ITNERFACE #] [END ITNERFACE #] [-t --type] [INTERFACE TYPE]
trunk [-p --permit] [-v --vlan] [VLAN# <range: 1 – 4094>]
trunk [-p --permit] [-v --vlan] [VLAN# ..]
trunk [-p --permit] [-v --vlan] all
access [-v --vlan] [VLAN#]

DESCRIPTION

these commands are used to set the port type to trunk or access mode and to set permited vlan or vlans.
The options are as follows:
Interface:
[ITNERFACE #] [-t --type] [INTERFACE TYPE]
[-r --range] [START ITNERFACE #] [END ITNERFACE #] [-t --type] [INTERFACE TYPE]
trunk:
[-p --permit] [-v --vlan] [VLAN#]
[-p --permit] [-v --vlan] [VLAN# ..]
[-p --permit] [-v --vlan] all
access:
[-v --vlan] [VLAN#]

EXAMPLES OF VALID MODES

Interface [ITNERFACE #] -t [e | f | g | T] | Interface [ITNERFACE #] --type [e | f | g | T]
example: Interface 3 -t g
Interface 3 --type g

Interface -r [START ITNERFACE #] [END ITNERFACE #] -t [e | f | g | T] | Interface --range [START ITNERFACE #] [END ITNERFACE #] --type [e | f | g | T]
example: Interface -r 4 8 -t g
Interface --range 4 8 --type g

trunk -p -v [VLAN#] | trunk --permit --vlan [VLAN#]
example: trunk -p -v 300
trunk --permit --vlan 300

trunk -p -v [VLAN# ..] | trunk --permit --vlan [VLAN# ..]
example: trunk -p -v 200 250 60 1000
trunk --permit --vlan 200 250 60 1000

trunk -p -v all | trunk --permit --vlan all
example: trunk -p -v all
trunk --permit --vlan all

access -v [VLAN#] | access --vlan [VLAN#]
example: access -v 200
access --vlan 200

VOICE VLAN Assignment commands

NAMES

Interface -- allows user to go inside switch's configuration modes
hybrid -- set the tagged and untagged vlans for hybrid port

SYNOPSIS

Interface [ITNERFACE #] [-t --type] [INTERFACE TYPE]
Interface [-r --range] [START ITNERFACE #] [END ITNERFACE #] [-t --type] [INTERFACE TYPE]
hybrid tagged [-v --vlan] [VLAN#]
hybrid untagged [-v --vlan] [VLAN#]

DESCRIPTION

These commands are used to set the port type to hybrid and to set tagged and untagged vlans in order to be used for voice.
The options are as follows:
Interface:
[ITNERFACE #] [-t --type] [INTERFACE TYPE]
[-r --range] [START ITNERFACE #] [END ITNERFACE #] [-t --type] [INTERFACE TYPE]
hybrid:
tagged [-v --vlan] [VLAN#]
untagged [-v --vlan] [VLAN#]

EXAMPLES OF VALID MODES

Interface [ITNERFACE #] -t [e | f | g | T] | Interface [ITNERFACE #] --type [e | f | g | T]
example: Interface 3 -t g
Interface 3 --type g

Interface -r [START ITNERFACE #] [END ITNERFACE #] -t [e | f | g | T] | Interface --range [START ITNERFACE #] [END ITNERFACE #] --type [e | f | g | T]
example: Interface -r 4 8 -t g
Interface --range 4 8 --type g

hybrid tagged -v [VLAN#] | hybrid tagged --vlan [VLAN#]
example: hybrid tagged -v 200
hybrid tagged --vlan 200

hybrid untagged -v [VLAN#] | hybrid untagged --vlan [VLAN#]
example: hybrid untagged -v 300
hybrid untagged --vlan 300

############### VLAN IP address commands ##############################
NAMES
Interface -- allows user to go inside switch's configuration modes
ip address -- set the ip address of a paticular vlan interface
SYNOPSIS
Interface vlan [-n --number] [VLAN #]
ip address [IP ADDRESS] [SUBNET MASK]
DESCRIPTION
These commands are used to set ip address to a paticular vlan interface.
The options are as follows:
vlan [-n --number] [VLAN #]
[IP ADDRESS] [SUBNET MASK]
EXAMPLES OF VALID MODES
Interface vlan -n [VLAN #] | Interface vlan --number [VLAN #]
example: Interface vlan -n 200
Interface vlan --number 200
ip address [IP ADDRESS] [SUBNET MASK]
example: ip address 192.168.1.1 255.255.255.0
############### Port Mgt. commands ####################################
NAMES
Interface -- allows user to go inside switch's configuration modes
description -- set port description
duplex -- set duplex
speed -- set speed
disable -- disable port
enable -- enable port
SYNOPSIS
Interface [ITNERFACE #] [-t --type] [INTERFACE TYPE]
Interface [-r --range] [START ITNERFACE #] [END ITNERFACE #] [-t --type] [INTERFACE TYPE]
description [DESCRIPTION STRING]
duplex [auto | full | half]
speed [10 | 100 | 1000 | 10000 | auto]
disable
enable
DESCRIPTION
These commands are used to configure port name, speed and duplex. further they are used to enable or disable a paticular ports
The options are as follows:
Interface:
[ITNERFACE #] -t --type [INTERFACE TYPE]
-r --range [START ITNERFACE #] [END ITNERFACE #] -t --type [INTERFACE TYPE]
description:
[DESCRIPTION STRING]
duplex:
[auto | full | half]
speed:
[10 | 100 | 1000 | 10000 | auto]

EXAMPLES OF VALID MODES
Interface [ITNERFACE #] -t [e | f | g | T] | Interface [ITNERFACE #] --type [e | f | g | T]
example: Interface 3 -t g
Interface -r [START ITNERFACE #] [END ITNERFACE #] -t [e | f | g | T] | Interface --range [START ITNERFACE #] [END ITNERFACE #] --type [e | f | g | T]
example: Interface -r 4 8 -t g
description [DESCRIPTION STRING]
duplex [auto | full | half]
speed [10 | 100 | 1000 | 10000 | auto]
dis | disable
en | enable
############### MSTP commands ##
NAMES
stp -- allow user to configure mstp

Interface --- allows user to go inside switch's configuration modes
region -- allow user to configure mstp regional configurations
instance -- allow user to configure mstp instance configurations

SYNOPSIS

stp [en | enable]
Interface stp [-r --region-configuration]
region [-n --name] [NAME STRING]
region [-r --revision] [REVISION #]
instance [INSTANCE#] [-v --vlan] [VLAN #]
instance [INSTANCE#] [-v --vlan] [VLAN #..]
stp [-p --priority] [PRIORITY#]
stp [-i --instance] [INSTANCE#] [-p --priority] [PRIORITY#]
stp pathcost [8021d | 8021t]
Interface [ITNERFACE #] [-t --type] [INTERFACE TYPE]
stp [-c --cost] [COST VALUE <8021d: range 1 – 65,535 | e = 99 | f = 19 | g = 4 | T = 2> or <8021t: range 1 – 200,000,000 | e = 2,000,000 | f = 200,000 | g = 20,000 | T = 2000>]
stp [-p --priority] [PRIORITY VALUE <0 – 61440> multiples of 4096]
stp [-i --instance] [INSTANCE #] [-c --cost] [COST VALUE]
stp [-i --instance] [INSTANCE #] [-p --priority] [PRIORITY VALUE]

DESCRIPTION

These commands are used to set mstp in a switch
The options are as follows:
stp:
[en | enable]
[-p --priority] [PRIORITY#]
[-i --instance] [INSTANCE#] [-p --priority] [PRIORITY#]
[-c --cost] [COST VALUE]
[-p --priority] [PRIORITY VALUE]
[-i --instance] [INSTANCE #] [-c --cost] [COST VALUE]
[-i --instance] [INSTANCE #] [-p --priority] [PRIORITY VALUE]
Interface stp:
[-r --region-configuration]
instance:
[INSTANCE#] [-v --vlan] [VLAN #]
[INSTANCE#] [-v --vlan] [VLAN #..]
Interface:
[ITNERFACE #] [-t --type] [INTERFACE TYPE]

EXAMPLES OF VALID MODES

stp en | stp enable
show stp -r | show stp –region-configuration
region -n [NAME STRING] | region --name [NAME STRING]
example: region -n mstp_r1
region --name mstp_r1
region -r [REVISION #] | region --revision [REVISION #]
example: region -r 1
region --revision 1
instance [INSTANCE#] -v [VLAN#..] | instance [INSTANCE#] --vlan [VLAN#..]
example: instance 1 -v 100 200 3 50
instance [INSTANCE#] --vlan 100 200 3 50
stp -p [PRIORITY#] | stp --priority [PRIORITY#]
example: stp -p 4096
stp --priority 4096
stp -i [INSTANCE#] -p [PRIORITY#] | stp --instance [INSTANCE#] --priority [PRIORITY#]
example: stp -i 1 -p 8192
stp --instance 1 --priority 8192
stp pathcost [8021d | 8021t]
example: stp pathcost 8021d
stp pathcost 8021t
Interface [ITNERFACE #] -t [e | f | g | T] | Interface [ITNERFACE #] --type [e | f | g | T]
example: Interface 3 -t g
Interface 3 --type g
stp -c [COST VALUE] | stp --cost [COST VALUE]
example: stp -c 4
stp --cost 20000

stp -p [PRIORITY VALUE] | stp --priority [PRIORITY VALUE]
example: stp -p 12288
stp --priority 12288
stp -i [INSTANCE #] -c [COST VALUE] | stp --instance [INSTANCE #] --cost [COST VALUE]
example: stp -i 1 -c 4
stp --instance 1 --cost 20000
stp -i [INSTANCE #] -p [PRIORITY VALUE] | stp --instance [INSTANCE #] --priority [PRIORITY VALUE]
example: stp -i 1 -p 61440
stp --instance 1 --priority 61440

################ MSTP Hardening commands ################################

NAMES

stp -- allow user to configure mstp
Interface -- allows user to go inside switch's configuration modes

SYNOPSIS

Interface [ITNERFACE #] [-t --type] [e | f | g | T]
stp dldp [en | enable]
stp edge-port [en | enable]
stp bpdu-guard [en | enable]
stp root-guard [en | enable]
stp loop-guard [en | enable]

DESCRIPTION

These commands are used to set mstp hardening configurations in a switch
The options are as follows:
Interface:
[ITNERFACE #] [-t --type] [e | f | g | T]
stp:
dldp [en | enable]
edge-port [en | enable]
bpdu-guard [en | enable]
root-guard [en | enable]
loop-guard [en | enable]

EXAMPLES OF VALID MODES

[ITNERFACE #] -t [e | f | g | T] | [ITNERFACE #] --type [e | f | g | T]
example: Interface 3 -t g
Interface 3 --type g
stp dldp en | stp dldp enable
stp edge-port en | stp edge-port enable
stp bpdu-guard en | stp bpdu-guard enable
stp root-guard en | stp root-guard enable
stp loop-guard en | stp loop-guard enable

################ Default route commands ###################################

NAMES

ip route -- set the default rout of a switch

SYNOPSIS

ip route [SOURCE IP] [WILDCARD MASK] [NEXTHOP IP]

DESCRIPTION

This command is used to set default route of a switch
The options are as follows:
[SOURCE IP] [WILDCARD MASK] [NEXTHOP IP]

EXAMPLES OF VALID MODES

ip route [SOURCE IP] [WILDCARD MASK] [NEXTHOP IP]
example: ip route 0.0.0.0 0.0.0.0 192.168.1.1

################ Standard ACL commands ###################################

NAMES

Interface -- allows user to go inside switch's configuration modes
permit -- allows configuration of permit rule in acl
deny -- allows configuration of deny rule in acl
acl -- allows setting up port based or routed acls

SYNOPSIS

Interface acl [-s --standard] [ACL # <range: 1 - 99>]
Interface acl [-s --standard] [ACL # <range: 1 - 99>] [-n --name] [ACL NAME STRING]
permit [-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD]
permit [-i --ip-address] any any

deny [src | source] [-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD] [dst | destination] [-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD]
Interface vlan [-n --number] [VLAN # <range: 1 – 4094>]
Interface [ITNERFACE #] [-t --type] [e | f | g | T]
acl [ACL#] [-m --mode] [inbound | outbound]
acl [-n --name] [ACL NAME STRING] [-m --mode] [inbound | outbound]

DESCRIPTION

These commands are used to set standard acls in a switch
The options are as follows:
Interface acl:
[-s --standard] [ACL #]
[-s --standard] [ACL #] [-n --name] [ACL NAME STRING]
permit:
[-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD]
[-i --ip-address] any any
deny:
[src | source] [-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD] [dst | destination] [-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD]
Interface:
vlan [-n --number] [VLAN #]
[ITNERFACE #] [-t --type] [e | f | g | T]
acl:
[ACL#] [-m --mode] [inbound | outbound]
[-n --name] [ACL NAME STRING] [-m --mode] [inbound | outbound]

EXAMPLES OF VALID MODES

Interface acl -s [ACL #] | Interface acl --standard [ACL #]
example: Interface acl -s 50
Interface acl --standard 50
Interface acl -s [ACL #] -n [ACL NAME STRING] | Interface acl --standard [ACL #] --name [ACL NAME STRING]
example: Interface acl -s 60 -n standard_numbered_acl
Interface acl --standard 60 --name standard_named_acl
permit -i [IP ADDR] -w [WILDCARD] | permit --ip-address [IP ADDR] --wildcard [WILDCARD]
example: permit -i 192.168.1.0 -w 0.0.0.255
permit --ip-address 192.168.1.0 --wildcard 0.0.0.255
permit -i any any | permit --ip-address any any
example: permit -i any any
permit --ip-address any any
deny src -i [IP ADDR] -w [WILDCARD] dst -i [IP ADDR] -w [WILDCARD] | deny source --ip-address [IP ADDR] --wildcard [WILDCARD] destination --ip-address [IP ADDR] --wildcard [WILDCARD]
example: deny src -i 192.168.1.0 -w 0.0.0.255 dst -i 192.168.1.254 -w 0.0.0.0
deny source --ip-address 192.168.1.0 --wildcard 0.0.0.255 destination --ip-address 192.168.1.254 --wildcard 0.0.0.0
Interface vlan -n [VLAN #] | Interface vlan --number [VLAN #]
example: Interface vlan -n 300
Interface vlan --number 300
Interface [ITNERFACE #] -t [e | f | g | T] | Interface [ITNERFACE #] --type [e | f | g | T]
example: Interface 4 -t g
Interface 4 --type g
acl [ACL#] -m [inbound | outbound] | acl [ACL#] --mode [inbound | outbound]
example: acl 50 -m inbound
acl 60 --mode outbound
acl -n [ACL NAME STRING] -m [inbound | outbound] | acl --name [ACL NAME STRING] --mode [inbound | outbound]
example: acl -n standard_named_acl -m inbound
acl --name standard_named_acl --mode outbound

################ Extended ACL commands ##################################

NAMES

Interface -- allows user to go inside switch's configuration modes
permit -- allows configuration of permit rule in acl
deny -- allows configuration of deny rule in acl
acl -- allows setting up port based or routed acls

SYNOPSIS

Interface acl [-e --extended] [ACL # <range: 100 - 199>]

Interface acl [-e --extended] [ACL # <range: 100 - 199>] [-n --name] [ACL NAME STRING]
permit [-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD]
permit [-i --ip-address] any any
deny [src | source] [-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD] [dst | destination] [-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD]
Interface vlan [-n --number] [VLAN # <range: 1 – 4094>]
Interface [ITNERFACE #] [-t --type] [e | f | g | T]
acl [ACL#] [-m --mode] [inbound | outbound]
acl [-n --name] [ACL NAME STRING] [-m --mode] [inbound | outbound]

DESCRIPTION

These commands are used to set extended acls in a switch
The options are as follows:
Interface acl:
[-e --extended] [ACL #]
[-e --extended] [ACL #] [-n --name] [ACL NAME STRING]
permit:
[-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD]
[-i --ip-address] any any
deny:
[src | source] [-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD] [dst | destination] [-i --ip-address] [IP ADDR] [-w --wildcard] [WILDCARD]
Interface:
vlan [-n --number] [VLAN #]
[ITNERFACE #] [-t --type] [e | f | g | T]
acl:
[ACL#] [-m --mode] [inbound | outbound]
[-n --name] [ACL NAME STRING] [-m --mode] [inbound | outbound]

EXAMPLES OF VALID MODES

Interface acl -e [ACL #] | Interface acl --extended [ACL #]
example: Interface acl -e 150
Interface acl --extended 150
Interface acl -e [ACL #] -n [ACL NAME STRING] | Interface acl --extended [ACL #] --name [ACL NAME STRING]
example: Interface acl -e 160 -n extended_numbered_acl
Interface acl --extended 160 --name extended_named_acl
permit -i [IP ADDR] -w [WILDCARD] | permit --ip-address [IP ADDR] --wildcard [WILDCARD]
example: permit -i 192.168.1.0 -w 0.0.0.255
permit --ip-address 192.168.1.0 --wildcard 0.0.0.255
permit -i any any | permit --ip-address any any
example: permit -i any any
permit --ip-address any any
deny src -i [IP ADDR] -w [WILDCARD] dst -i [IP ADDR] -w [WILDCARD] | deny source --ip-address [IP ADDR] --wildcard [WILDCARD] destination --ip-address [IP ADDR] --wildcard [WILDCARD]
example: deny src -i 192.168.1.0 -w 0.0.0.255 dst -i 192.168.1.254 -w 0.0.0.0
deny source --ip-address 192.168.1.0 --wildcard 0.0.0.255 destination --ip-address 192.168.1.254 --wildcard 0.0.0.0
Interface vlan -n [VLAN #] | Interface vlan --number [VLAN #]
example: Interface vlan -n 300
Interface vlan --number 300
Interface [ITNERFACE #] -t [e | f | g | T] | Interface [ITNERFACE #] --type [e | f | g | T]
example: Interface 4 -t g
Interface 4 --type g
acl [ACL#] -m [inbound | outbound] | acl [ACL#] --mode [inbound | outbound]
example: acl 50 -m inbound
acl 60 --mode outbound
acl -n [ACL NAME STRING] -m [inbound | outbound] | acl --name [ACL NAME STRING] --mode [inbound | outbound]
example: acl -n extended_named_acl -m inbound
acl --name extended_named_acl --mode outbound