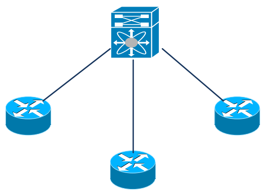
NDNA User Guide ©
THE NETWORK DISCOVERY "N" AUTOMATION PROGRAM

FEATURING: THE NDNA DIAGRAM GENERATOR ©

Plugin for:

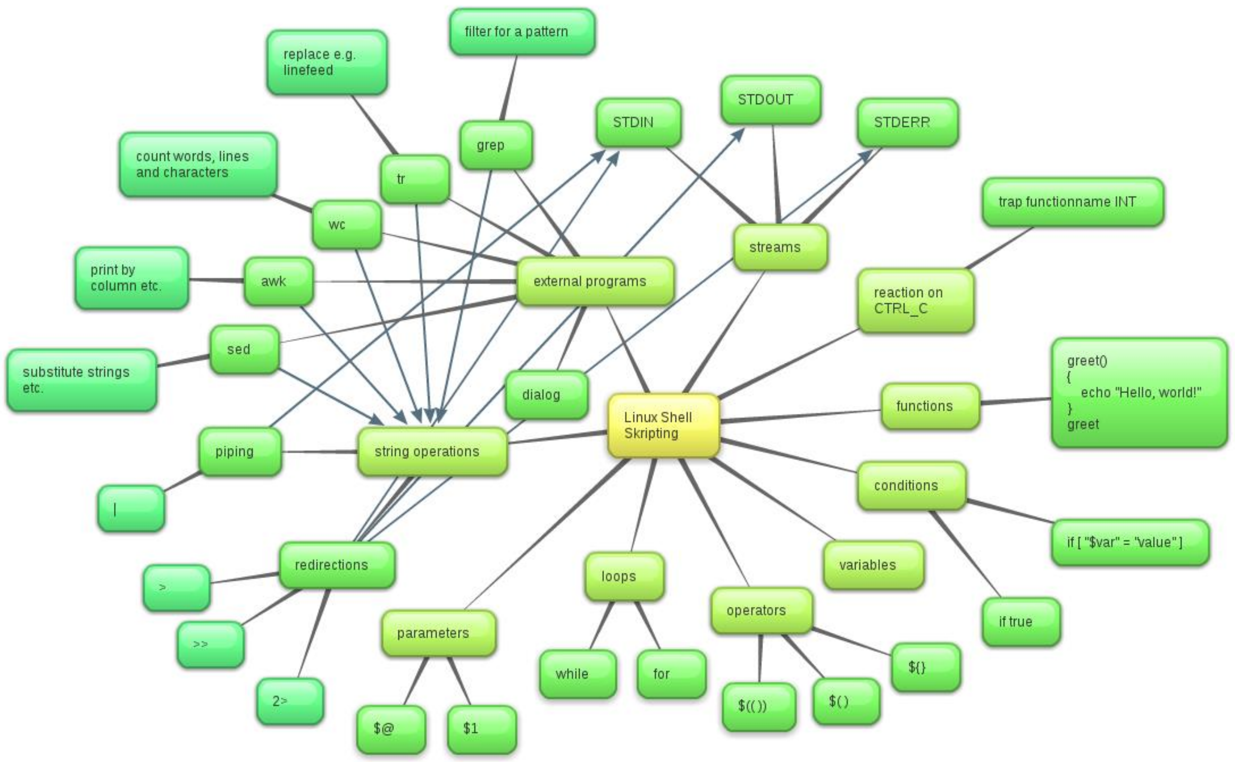


NDNA Virtual Machine Powered by:

&

&

Contents


Linux Shell Skripting
external programs
tr
replace e.g. linefeed
grep
filter for a pattern
wc
count words, lines and characters
awk
print by column etc.
sed
substitute strings etc.
dialog
streams
STDIN
STDOUT
STDERR
reaction on CTRL_C
trap functionname INT
functions
greet()
{
echo "Hello, world!"
}
greet
conditions
if ["$var" = "value"]
if true
variables
operators
$(( ))
$( )
${}
loops
while
for
parameters
$@
$1
string operations
piping
|
redirections
>
>>
2>

Getting Started

To get started on the path to network automation, and automating your day to day tasks, follow the steps provided in this document to get all needed client software installed.

Please refer to this guide and to the online instructional videos often to get the most out of the NDNA program/s.

Thank you!

Minimum Hardware Requirements:

1. Host CPU running an Intel I5 or better. Set your VM CPU to match. (Will be covered in the chapter on setting up your VM)
2. Host has at least 8 GB of RAM, Giving the Guest VM at least 2GB of RAM
3. At least 15 GB free hard disk space for the Virtual Machine (20GB recommended). The VM has a dynamic hard disk that has 6 GB used and can grow to 20 GB. *To check the current disk space in the Linux appliance, from the CLI issue the command "df -h"*
4. Wired access to the network. ***Do not run the program on a wireless connection***
   In fact, we've experienced issues with DHCP and/or vmnet0 even working on a wireless connection. Bottom line is, even if you can get the network running (DHCP or static) on a wireless connection, we do NOT recommend running the NDNA application on a wireless connection.

Minimum Software Requirements:

1. VMware workstation PRO or free player (For non-commercial use) (Not supported on vSphere/ESXi)
2. Oracle Virtual Box (Mac, Windows or Linux)
3. AMI (Amazon Machine Image via Amazon Web Services) is not supported
4. vSphere/ESXi is not supported
5. NOTE: It's OK to run this VM inside of another VM, e.g. a Server running on ESXi. You can install VM player, or VirtualBox on the Server, and run the VM in either one of these (VM player, or VirtualBox). You will get performance warnings, but it will still run and function just fine.

Credits:

Network Discovery "N" Automation Program

Author: Brett M. Spunt: Design & Programming
CCIE #12745

NDNA Diagram Generator:

Jointly developed by Chris Roth – AKA "The Visio Guy" http://www.visguy.com/ and Brett M. Spunt

Design: Brett M. Spunt: Design & Overall App Framework, Meta-Data & Linux side programming (app data that gets fed into the Diagram Generator)

Author (Design and Programming): Chris Roth: NDNA Diagram Generator Coding and Design - All Windows side programming (e.g. Author of the NDNA Diagram Generator Visio Plugin)

Download and Install Companion Software (Freeware)

WinSCP (Windows)

https://winscp.net/eng/download.php#download2

*This program *is* required to effectively run the application, allowing you to navigate, view and transfer files quickly and easily between the host (Your physical machine hosting the VM) and guest machine (The NDNA VM)*

Cyberduck (Mac)

https://cyberduck.io/

*This program *is* required to effectively run the application, allowing you to navigate, view and transfer files quickly and easily between the host (Your physical machine hosting the VM) and guest machine (The NDNA VM)*

Sublime Text Editor (Windows, Mac)

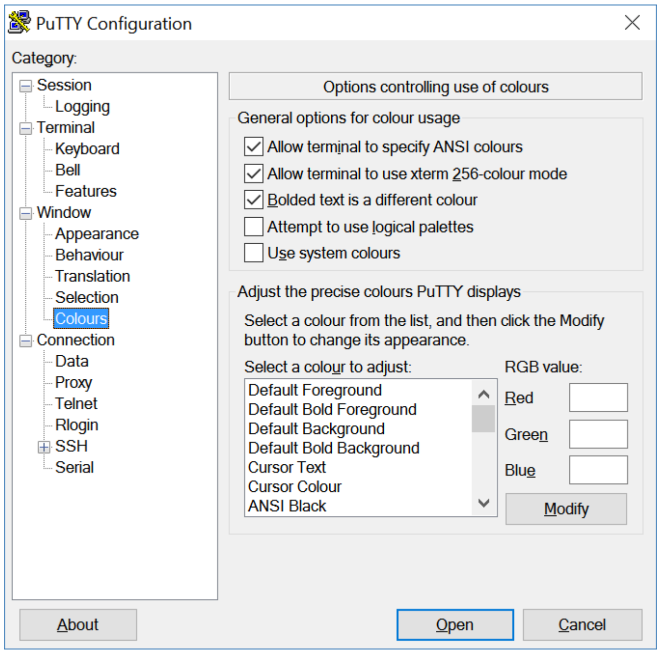
https://www.sublimetext.com/2

This is needed to view the CDP neighbor's flat files. Don't use notepad, use a real text editor. You'll need this as well to view files you work with as you customize your workflow, since these files will have Unix line endings (which won't view properly in notepad). You need to use a real text editor. If you're still using notepad, now is the time to change ☺

Putty (Windows) or SecureCRT

https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

This is needed to SSH to the NDNA VM from your host machine



Options controlling use of colours

General options for colour usage

- [✓] Allow terminal to specify ANSI colours
- [✓] Allow terminal to use xterm 256-colour mode
- [✓] Bolded text is a different colour
- [ ] Attempt to use logical palettes
- [ ] Use system colours

Adjust the precise colours PuTTY displays

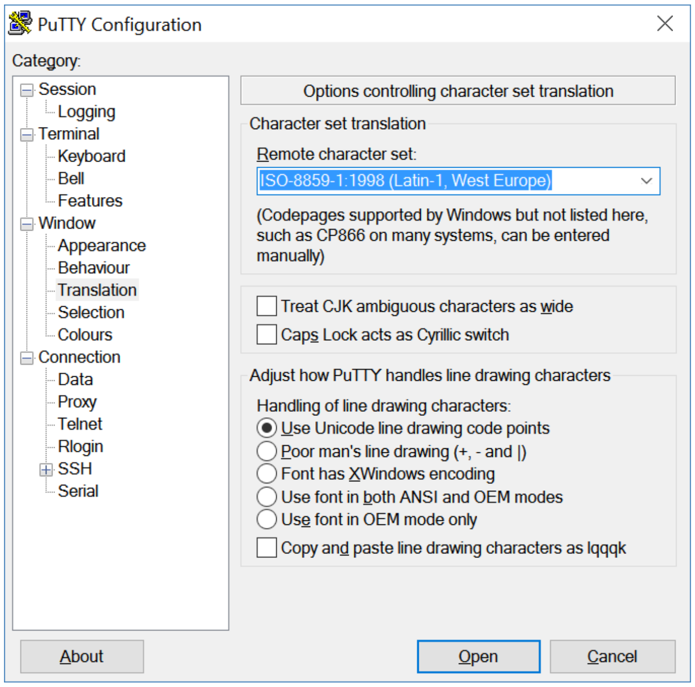
Select a colour from the list, and then click the Modify button to change its appearance.

Select a colour to adjust: Default Foreground, Default Bold Foreground, Default Background, Default Bold Background, Cursor Text, Cursor Colour, ANSI Black

RGB value: Red, Green, Blue

On the NDNA connection - Setup the following in Putty:

Window/Translation settings – set as shown in screenshot below:



Options controlling character set translation

Character set translation

Remote character set: ISO-8859-1:1998 (Latin-1, West Europe)

(Codepages supported by Windows but not listed here, such as CP866 on many systems, can be entered manually)

- [ ] Treat CJK ambiguous characters as wide
- [ ] Caps Lock acts as Cyrillic switch

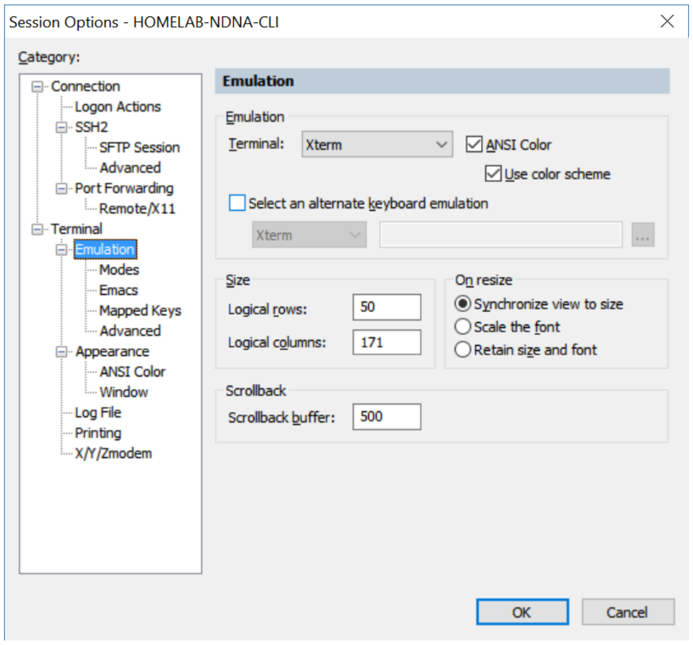
Adjust how PuTTY handles line drawing characters

Handling of line drawing characters:

- [✓ Use Unicode line drawing code points]
- Poor man's line drawing (+, - and |)
- Font has XWindows encoding
- Use font in both ANSI and OEM modes
- Use font in OEM mode only

- [ ] Copy and paste line drawing characters as lqqqk

*Or if you have a paid version of Secure CRT – You *must* setup as shown below,* ***<u>using Xterm, check ANSI Color and Use color scheme</u>: If you do not, the NDNA program will not look or run correctly in Secure CRT***



Emulation

Emulation

Terminal: Xterm

[✓] ANSI Color

[✓] Use color scheme

[ ] Select an alternate keyboard emulation

Xterm

Size

Logical rows: 50

Logical columns: 171

On resize

● Synchronize view to size

○ Scale the font

○ Retain size and font

Scrollback

Scrollback buffer: 500

VMware Workstation or Free Player (Only for non-commercial use)

If you are not already using VM Workstation, download either the VMware free player for Windows and Linux, or VirtualBox (Any platform, e.g. Mac). *We recommend VMware.*

https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/12_0

This is needed to run your VM.

ALERT: *To shutdown your VM, you must do this properly from the command line. Use the following command:*

shutdown -h now

VirtualBox

https://www.virtualbox.org/wiki/Downloads

This is needed to run your VM if you can't or don't want to use VMware.

ALERT: *To shutdown your VM, you must do this properly from the command line. Use the following command:*

shutdown -h now

Download and Set Up the Virtual Machine

Download the OVA (Virtual Machine Archive)

If you are not already a VMware or VirtualBox user, and you need a tutorial on setup, see the following links for instruction on importing an OVA into either one of these applications:

VMware Setup (Recommended):

https://pubs.vmware.com/workstation-9/index.jsp?topic=%2Fcom.vmware.ws.using.doc%2FGUID-DDCBE9C0-0EC9-4D09-8042-18436DA62F7A.html

See documentation below which was taken directly from the link above

Import an Open Virtualization Format Virtual Machine

You can import an Open Virtualization Format (OVF) virtual machine and run it in Workstation. Workstation converts the virtual machine from OVF format to VMware runtime (.vmx) format. You can import both. ovf and. ova files.

OVF is a platform-independent, efficient, extensible, and open packaging and distribution format for virtual machines. For example, you can import OVF virtual machines exported from VMware Fusion™ into Workstation. You can import OVF 1.0 and later files only.

You can also use the standalone OVF Tool to convert an OVF virtual machine to VMware runtime format. The standalone version of the OVF Tool is installed in the Workstation installation directory under OVFTool. See the *OVF Tool User Guide* on the VMware Web site for information on using the OVF Tool.

Procedure

1 In Workstation, select File > Open.

2 Browse to the. ovf or. ova file and click Open.

3 Type a name for the virtual machine, type or browse to the directory for the virtual machine files, and click Import.

Workstation performs OVF specification conformance and virtual hardware compliance checks. A status bar indicates the progress of the import process.

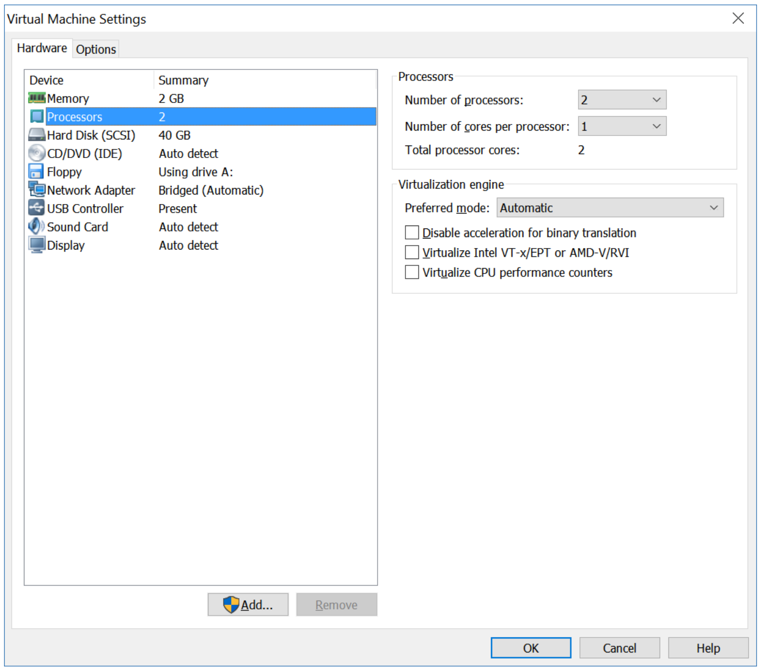
4 *If the import fails, click Retry to try again, or click Cancel to cancel the import.*

If you retry the import, Workstation relaxes the OVF specification conformance and virtual hardware compliance checks and you might not be able to use the virtual machine in Workstation.

After Workstation successfully imports the OVF virtual machine, the virtual machine appears in the virtual machine library.

- Within the VMware network settings, keep the network set to NAT.
- *Allocate the virtual machine at least 2GB of RAM*
- *Make sure the network adapter is set to NAT on VMware.*

- *Make sure you* to change the Number of processors and processor cores to match your hardware, e.g. 4 cores and 1 processor if you have a single quad core CPU, or 2 if you have a dual core CPU. Do not check off any additional boxes, e.g. Virtualize Intel VT-x etc. Just leave these unchecked.



Device	Summary
Memory	2 GB
Processors	2
Hard Disk (SCSI)	40 GB
CD/DVD (IDE)	Auto detect
Floppy	Using drive A:
Network Adapter	Bridged (Automatic)
USB Controller	Present
Sound Card	Auto detect
Display	Auto detect

Processors
Number of processors: 2
Number of cores per processor: 1
Total processor cores: 2

Virtualization engine
Preferred mode: Automatic
☐ Disable acceleration for binary translation
☐ Virtualize Intel VT-x/EPT or AMD-V/RVI
☐ Virtualize CPU performance counters

Linux VM network settings:

Even though we recommend leaving the Network adaptor using NAT and DHCP inside Linux, Here are instructions on how to set eth0 to either static or DHCP *inside* the Linux Virtual Machine:

Login to the VM via VMware directly (e.g. from the VMware console) using:
Username = root
Password = dcdp

1. Edit the /etc/network/interfaces file using the following command:
   nano /etc/network/interfaces
2. Change the following part of the config file as needed for eth0:

   auto eth0

   iface eth0 inet static ***(for static) or***

   iface eth0 inet dhcp ***(for dhcp – default setup)***

   If dhcp, keep the comments on the lines below:

   #address 192.168.88.185

   #netmask 255.255.255.0

   #gateway 192.168.88.1

   if static, change lines below to match the subnet and default GW you wish to be on and remove the comments # characters.

   address 192.168.88.185

   netmask 255.255.255.0

   gateway 192.168.88.1

3. Save the file by hitting ctrl O, hit enter, the ctrl X to exit. (note O is alpha, not numeric, e.g. not zero)
4. Stop and Start the network service using the command “service networking restart”
5. Now, you can and should connect to the VM via both SFTP Client (WinSCP or CyberDuck for file explorer type of file transfer and file/folder browsing) and SSH Client (Putty.py for CLI access) using the IP address of the Virtual Machine. You can get the IP by issuing the command “ifconfig”

Final Steps Before Running the Program

1. Read this User Guide.
2. Use bookmarks in WinSCP – These are referenced in the online instructional videos.
3. *Change the root password right away.* You can do so using the following command from the command line: passwd

- *Do not change the MySQL password. This will break the application, as it's written into the code.*

NDNA Program Operation

This chapter covers the NDNA main program operation. You run the main program, followed by numerous additional steps to ensure the most complete discovery, and to generate network diagrams. *In addition, these tasks also set you up for automation of the network (Discovery, Programming, and Troubleshooting).*

NDNA (Main) Program Operation Overview

When the main program runs, it does the following, in the following order:

1. Discovers the Cisco network in a loop free fashion using SSH, Python and CDP. This goes as far as it can see devices via CDP, up to 10 levels deep. In our extensive testing in lab and enterprise production environments, we never got more than 7 levels deep. (See CDP levels diagram below to visualize CDP levels). ***You will see the levels are not linear in nature…They go in all directions.***
2. After it's performed its main discovery, it uses SSH and Python automation to re-connect to the devices it successfully connected to using TACACs (or local or Radius) credentials.
3. It re-connects numerous times to these devices performing the following tasks:
    a. Grab Routing Protocol Information to build IP lists broken down by L2 and L3 devices
    b. Grab OS information to split IP lists by NXOS and IOS/IOS-XE
    c. Grab CDP information to build nicely formatted CDP-Neighbors files
    d. Grab various IOS/NXOS device information to build a CDP and IOS/NXOS device inventory that is exported into MySQL databases running on the VM.
    e. Exports the databases into xml format
    f. Combines all three xml files into one. (CDP, IOS and NXOS)
    g. This xml is then used to automatically generate diagrams in Microsoft Visio using the NDNA diagram generator.
4. The program builds Excel spreadsheets (CSVs) with hostname to IPs (One for IOS devices and one for NXOS devices.)
5. The program generates "stats" files which breakdown all the connection results, e.g. how many IOS, NXOS IPs were connected to, further broken down by L2/L3 counts. In addition, gives you a summary of how many hosts (and IPs provided) that were running incompatible SSH version, and how many hosts (and IPs provided) where authentication failed***. This gives you an easy reference to investigate these *problem* IOS/NXOS devices.***
6. Additional steps/considerations will be covered *after* the section "Running NDNA"
7. The program makes a full backup of the site/datacenter at the end of the program into the /usr/Backups directory. This is a dated zip file. There is also a restore program to restore from these zip files back into a DataCenter.
8. When the NDNA program is started, if that datacenter already exists and has custom configurations in the configs directory (which would be there after running custom automation scripts *post* discovery), then those files are zipped up and backed up to the /usr/Backups directory before the NDNA program starts.
9. The program uses a Company Code, Site/DC naming convention which builds a directory structure based on a Company/Site basis. This allows easy creation (via provided scripts) of enterprise wide IP lists (e.g. that span all sites of a company) and on a *per-company* basis or subset of a company

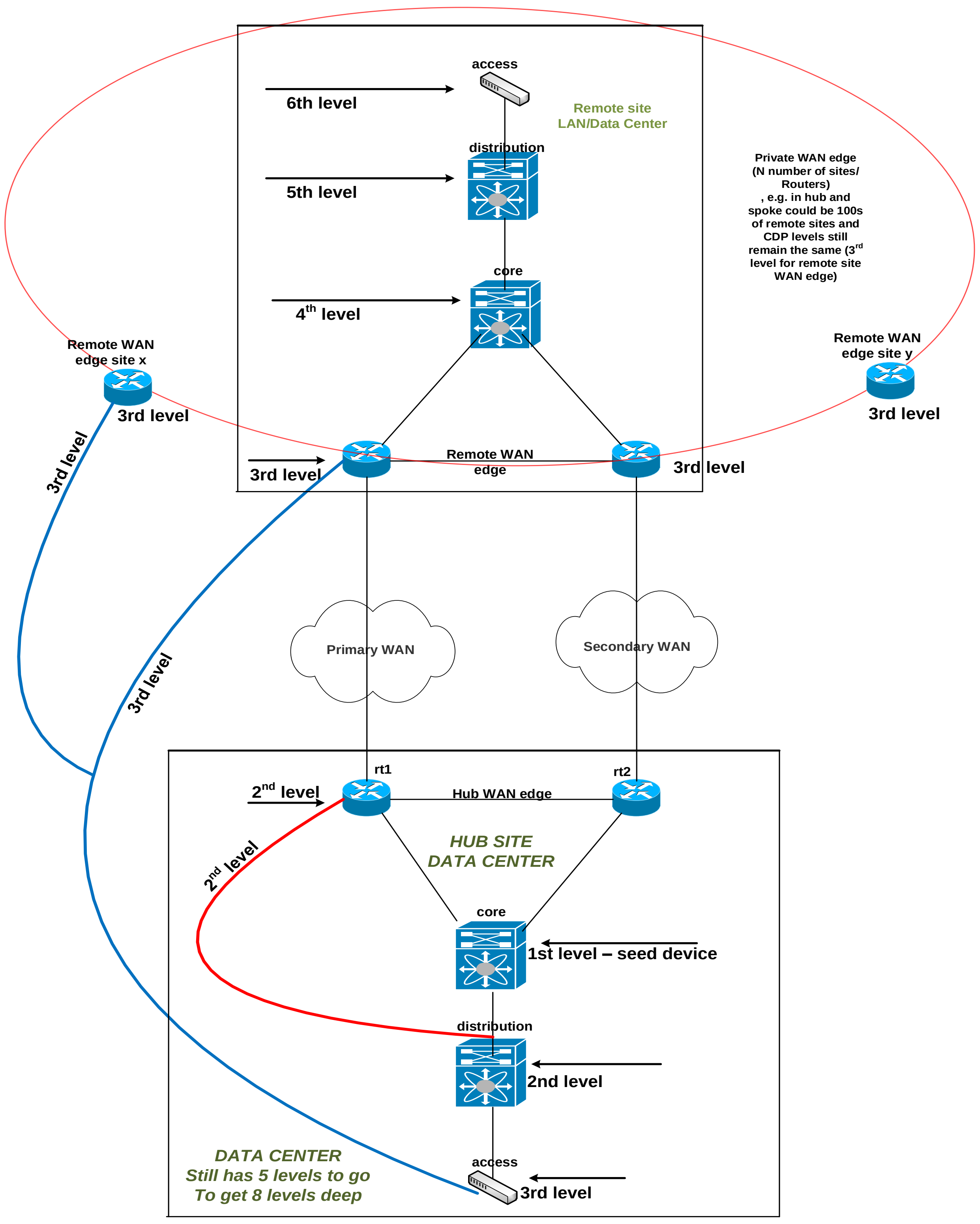
Folder Structure and Navigation

This is an important section which discusses *where* (folder location) you use the program at various times of operation.

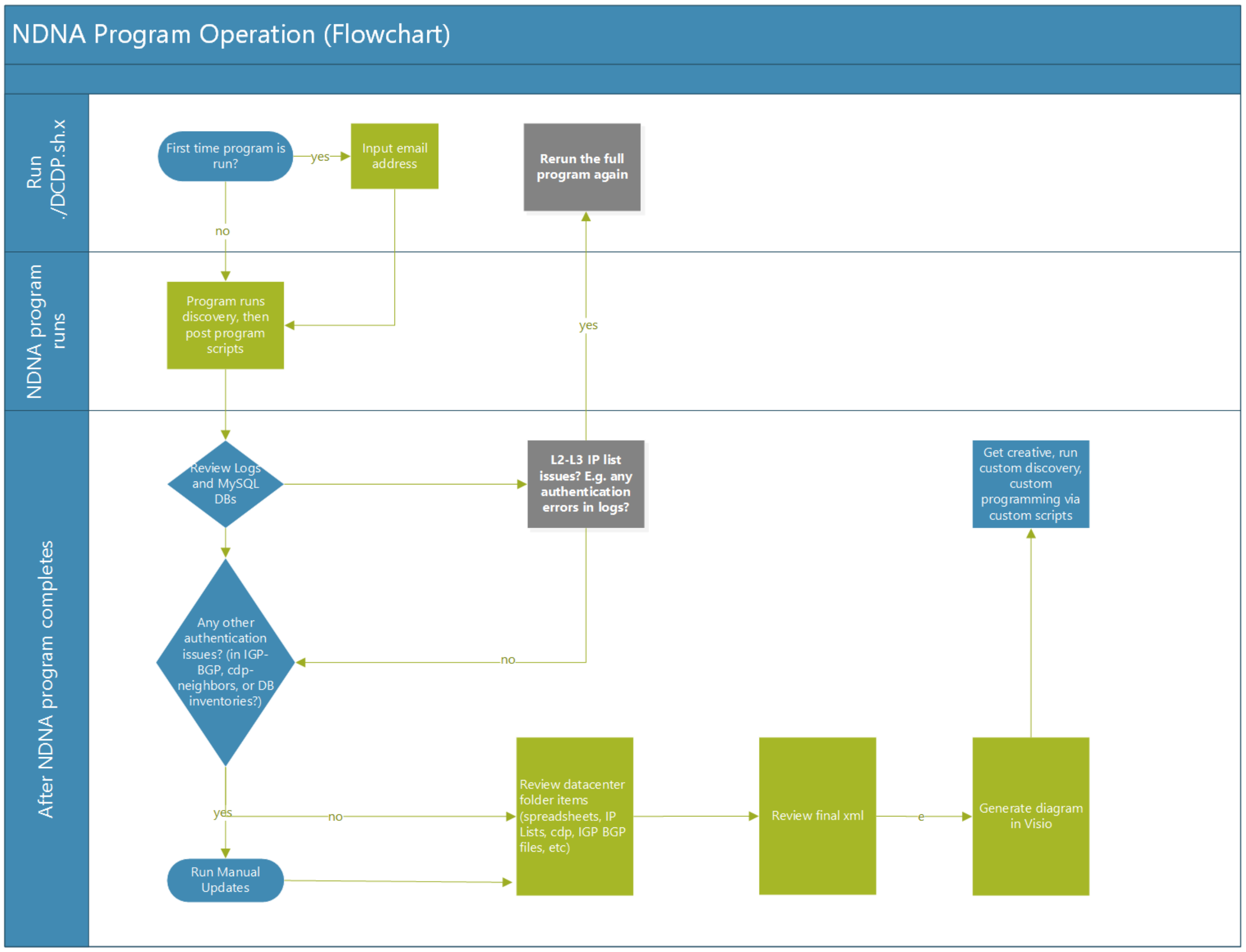
Task	Folder Location	Purpose	Notes
Run main program ./DCDP.sh	/usr/DCDP/	Discover the Network. Build Network Artifacts	This is the Main "NDNA" Discovery Program
Review Logs	/usr/DCDP/logs or /usr/DataCenters/<variable DC Name>/DCDP/logs	Review logs for issues to resolve after running the main program. These can be reviewed in either location noted in "Folder Location".	As you run discovery on new Data-Centers, the logs in /usr/DCDP will no longer point to a previous DC. At that point, you want to always use the DC path to review logs for that Data-Center
Review Site Data	/usr/DataCenters/<variable DC Name>/DCDP/*	Review all artifacts, e.g. IP lists, Inventories, spreadsheets, & cdp neighbors' files,etc.	See chapters on NDNA program operation for more info.
Run Custom Automation (all types, e.g. site level, Enterprise-wide, vendor-neutral)	*Always run from:* /usr/DCDP/bin/ python_custom_scripts/*	Automate discovery, troubleshooting and programming the network.	See chapters on automation for more info.
View configs After running custom automation At the "Site-Specific" level	/usr/DataCenters/<variable DC Name>/DCDP/configs	View site specific configurations pulled during automation for each node.	See chapters on automation for more info.
View configs After running custom automation At the "enterprise-wide-switches" level	/usr/enterprise-wide-switches/configs	View enterprise-wide-switch configurations pulled during automation for each node.	See chapters on automation for more info.
View configs After running custom automation At the "enterprise-wide-routers" level	/usr/enterprise-wide-routers/configs	View enterprise-wide-router configurations pulled during automation for each node.	See chapters on automation for more info.

Task	Folder Location	Purpose	Notes
View configs After running vendor neutral automation	/usr/DCDP/usr-level-folders/vendor_neutral/configs	View vendor neutral configurations pulled during automation for each node.	See chapters on automation for more info.
Restore a Data-Center	/usr/Restore	To restore a Data-Center from backup	Copy over backup from /usr/Backups dir to /usr/Restore first. See Restore Chapter
Diagram Generation	/usr/DataCenters/<variable DC Name>/DCDP/bin/diagram-generation	Xml file location used to generate diagrams in Microsoft Visio via the NDNA Diagram generator	Copy over xml file from this location to your windows desktop via WinSCP where you have Microsoft Visio installed
Run Manual Updates	/usr/DCDP/bin python_custom_scripts/ Manual-Updates/*	To re-run various scripts if any authentication errors caused nodes to be missed. This includes cdp-neighbors, DB inventories, etc.	See chapter on Manual Updates for more info.
View results of Manual Updates. e.g. new DB inventory, etc.	/usr/DataCenters/<variable DC Name>/DCDP/* and also http://<your-VM-IP>/phpmyadmin/index.php	View results of new artifacts created after re-running Manual Update.	To view log file results of Manual Updates see: /usr/DCDP/logs
View log files of Manual Updates Results	/usr/DCDP/logs	Identify if authentication errors went away and if manual update was successful or not.	See chapter on Manual Updates for more info.

CDP Level Visualization



NDNA Program Operation Flowchart


NDNA Program Operation (Flowchart)
Run ./DCDP.sh.x
NDNA program runs
After NDNA program completes
First time program is run?
yes
Input email address
no
Rerun the full program again
Program runs discovery, then post program scripts
yes
Review Logs and MySQL DBs
L2-L3 IP list issues? E.g. any authentication errors in logs?
Get creative, run custom discovery, custom programming via custom scripts
Any other authentication issues? (in IGP-BGP, cdp-neighbors, or DB inventories?)
no
yes
no
Review datacenter folder items (spreadsheets, IP Lists, cdp, IGP BGP files, etc)
Review final xml
e
Generate diagram in Visio
Run Manual Updates

Running NDNA (Network Discovery N Automation Program)

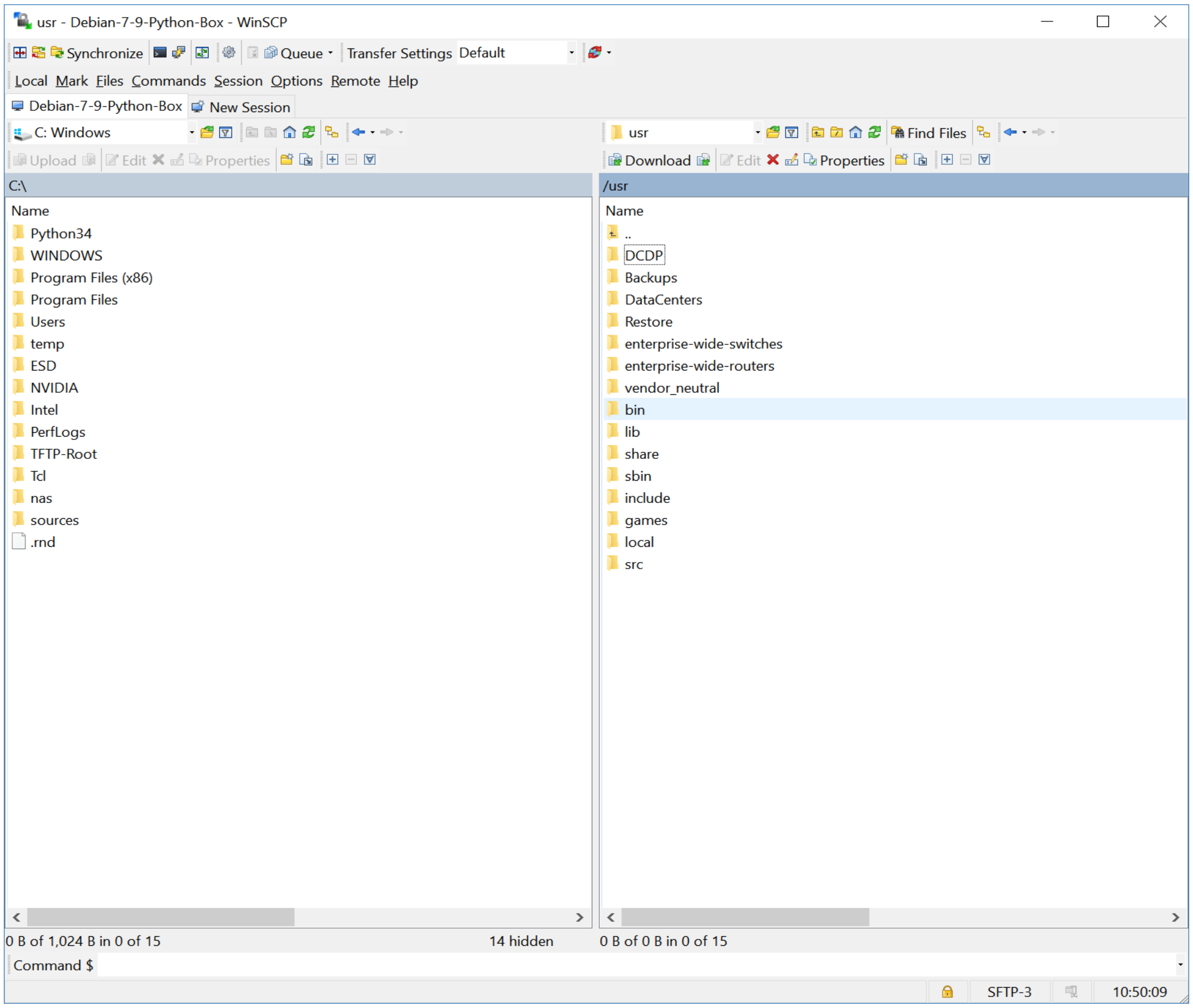
This section covers running the main program to discover the network.

Main Program Operation

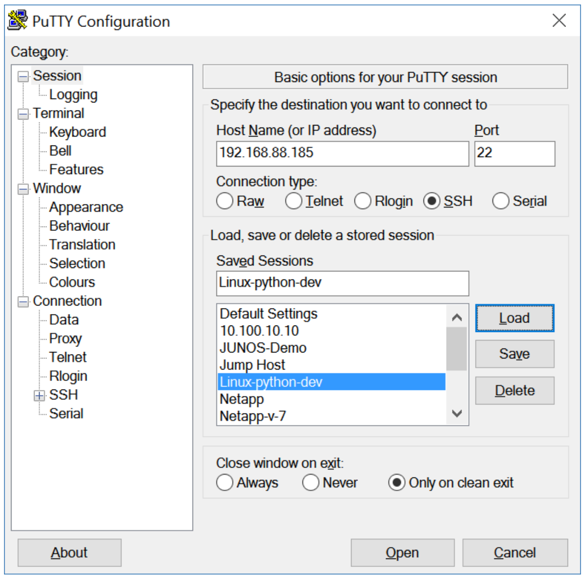
From Windows:

Launch WinSCP to create an SFTP connection to the Linux VM. Change to the /usr/DCDP directory. *You want to *always* have a WinSCP session going when using the application*. You will constantly use this to browse files, copy files between the host and guest machines *(Your windows machine and the Linux NDNA Virtual Machine).* If you are on a Mac, then launch CyberDuck SFTP client.

The WinSCP interface is shown below. This gives you *file explorer* type of access to view files/folders and copy between the host machine and the guest machine. You will need this flexibility to run the program efficiently. You can create bookmarks, copy paths (and then paste into your CLI) etc.



Launch Putty and create an SSH connection to the Linux VM.
Login as root.
Password is dcdp



Host Name (or IP address): 192.168.88.185
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Saved Sessions: Linux-python-dev
Close window on exit: Always Never [✓ Only on clean exit]

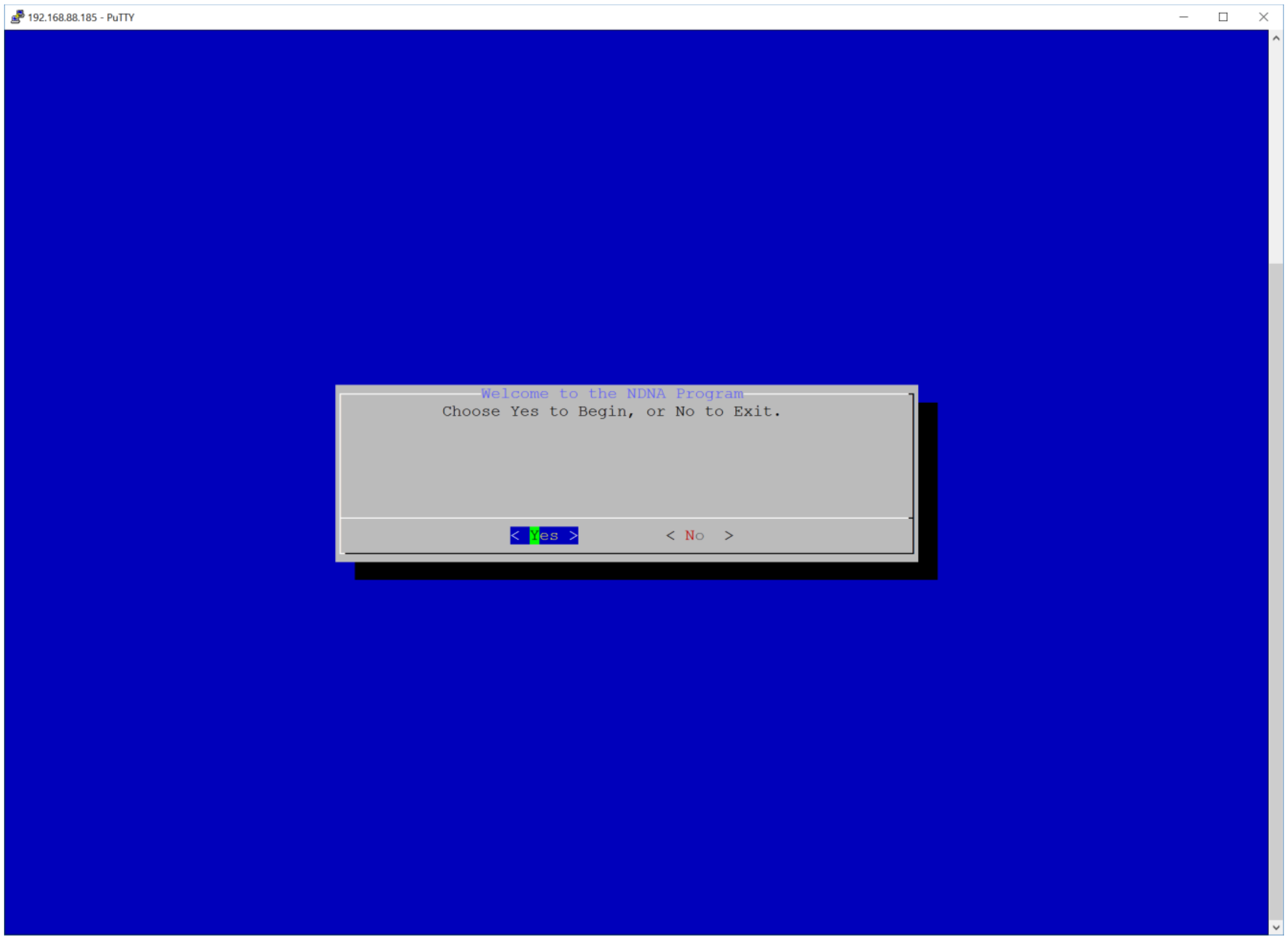
Change to the /usr/DCDP directory using the following command:

```
cd /usr/DCDP
```

Note: all Linux VM commands are always case sensitive.

To launch the program, type in:

```
./DCDP.sh
```



```
Choose Yes to Begin, or No to Exit.
```

Hit enter and follow the prompts

Note:

You can always hit “Tab” to finish off a command and “Tab” two times to see what matches there are in the current directory based on what you’ve already typed.

root@debian-python:/usr/DCDP# ./DCDP.sh

Enter a name as described: 3-character company code, followed by a colon, followed by your Data-Center/Site name (which can be any length)

Come up with a good naming scheme for each company you work with, and document it.

A few examples are shown below:

Example 1:
Coca Cola Company (Los Angeles and New York Offices):

- *CCC:LA-DC*
- *CCC:NYC-DC*

Example 2:
Disney Company (Los Angeles and New York Offices):

- *DIS:LA-DC*
- *DIS:NYC-DC*

Important Note:
You can also choose to break down the company into subsets. This is critical if you want to automatically generate enterprise wide IP lists later based on subsets of a company. ***Plan this out NOW!***

Examples of creating a naming convention based on subsets of a company, e.g. by region of the world:

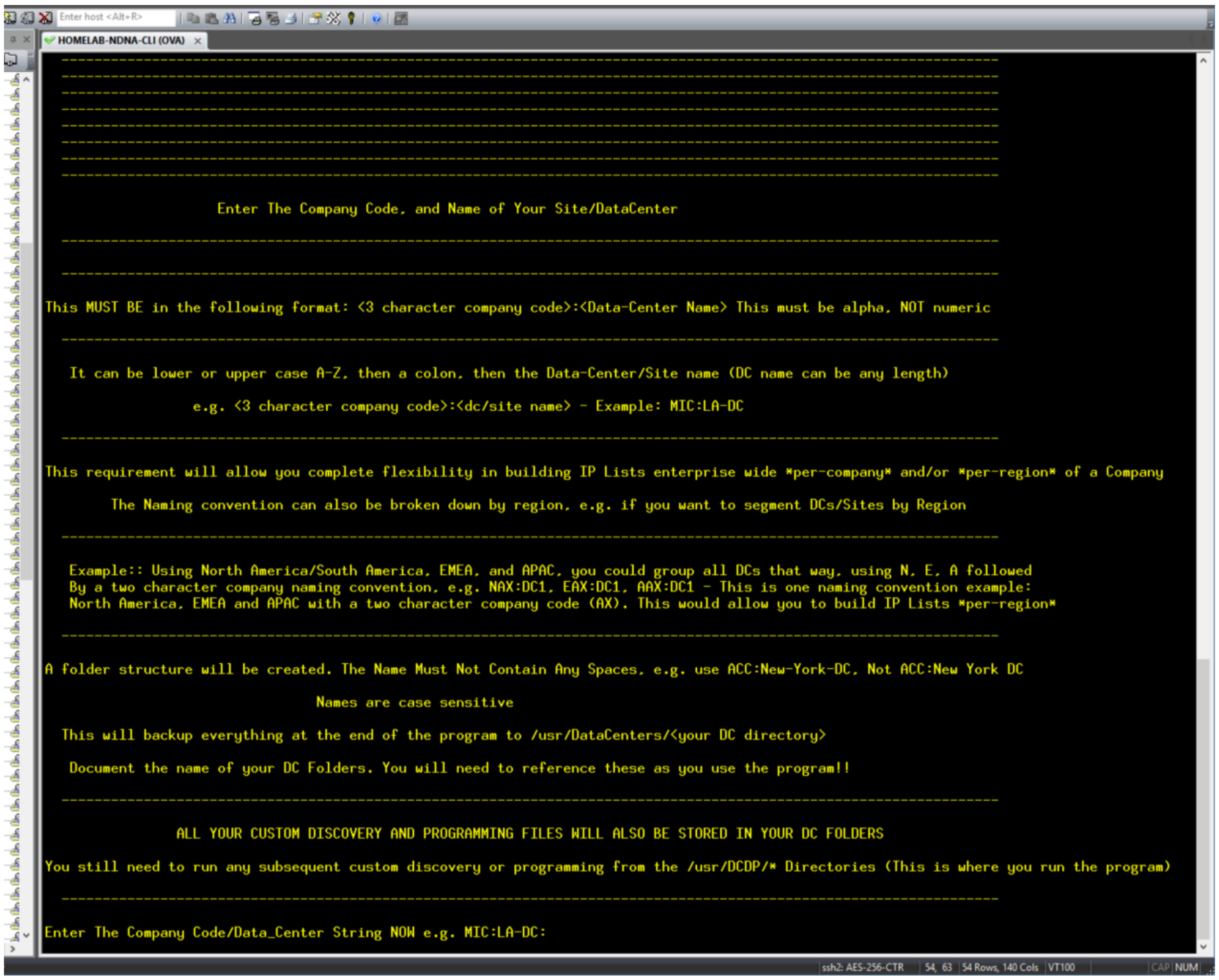
Example 1:
Coca Cola Company (EMEA, APAC and NASA regions):

- ***E**CC:LONDON-DC*
- ***A**CC:-SINGAPORE-DC*
- ***N**CC:-NEW-YORK-DC*

Example 2:
Disney Company (EMEA, APAC and NASA regions):

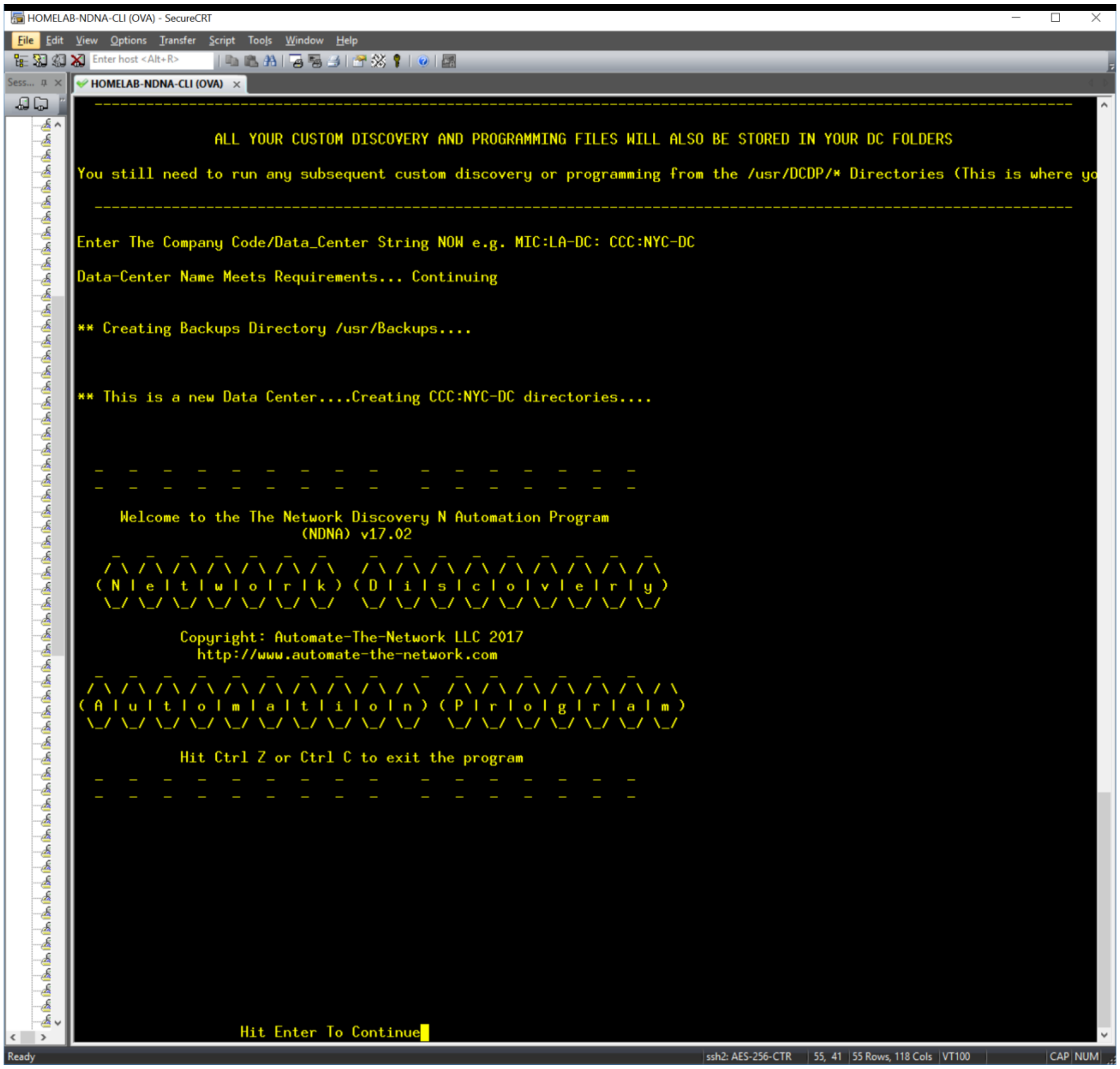
- ***E**DI:LONDON-DC*
- ***A**DI:-SINGAPORE-DC*
- ***N**DI:-NEW-YORK-DC*

Best viewed at 200% magnification:



```
  --------------------------------------------------------------------------------------------------------------------
  --------------------------------------------------------------------------------------------------------------------
  --------------------------------------------------------------------------------------------------------------------
  --------------------------------------------------------------------------------------------------------------------
  --------------------------------------------------------------------------------------------------------------------
  --------------------------------------------------------------------------------------------------------------------
  --------------------------------------------------------------------------------------------------------------------
  --------------------------------------------------------------------------------------------------------------------

                     Enter The Company Code, and Name of Your Site/DataCenter

  --------------------------------------------------------------------------------------------------------------------

  --------------------------------------------------------------------------------------------------------------------

This MUST BE in the following format: <3 character company code>:<Data-Center Name> This must be alpha, NOT numeric

  --------------------------------------------------------------------------------------------------------------------

   It can be lower or upper case A-Z, then a colon, then the Data-Center/Site name (DC name can be any length)

                  e.g. <3 character company code>:<dc/site name> - Example: MIC:LA-DC

  --------------------------------------------------------------------------------------------------------------------

This requirement will allow you complete flexibility in building IP Lists enterprise wide *per-company* and/or *per-region* of a Company

        The Naming convention can also be broken down by region, e.g. if you want to segment DCs/Sites by Region

  --------------------------------------------------------------------------------------------------------------------

   Example:: Using North America/South America, EMEA, and APAC, you could group all DCs that way, using N, E, A followed
   By a two character company naming convention, e.g. NAX:DC1, EAX:DC1, AAX:DC1 - This is one naming convention example:
   North America, EMEA and APAC with a two character company code (AX). This would allow you to build IP Lists *per-region*

  --------------------------------------------------------------------------------------------------------------------

A folder structure will be created. The Name Must Not Contain Any Spaces, e.g. use ACC:New-York-DC, Not ACC:New York DC

                                 Names are case sensitive

  This will backup everything at the end of the program to /usr/DataCenters/<your DC directory>

   Document the name of your DC Folders. You will need to reference these as you use the program!!

  --------------------------------------------------------------------------------------------------------------------

                ALL YOUR CUSTOM DISCOVERY AND PROGRAMMING FILES WILL ALSO BE STORED IN YOUR DC FOLDERS

You still need to run any subsequent custom discovery or programming from the /usr/DCDP/* Directories (This is where you run the program)

  --------------------------------------------------------------------------------------------------------------------

Enter The Company Code/Data_Center String NOW e.g. MIC:LA-DC:
```

Next, you'll see another welcome screen.
You can see I've entered a name of CCC:NYC-DC

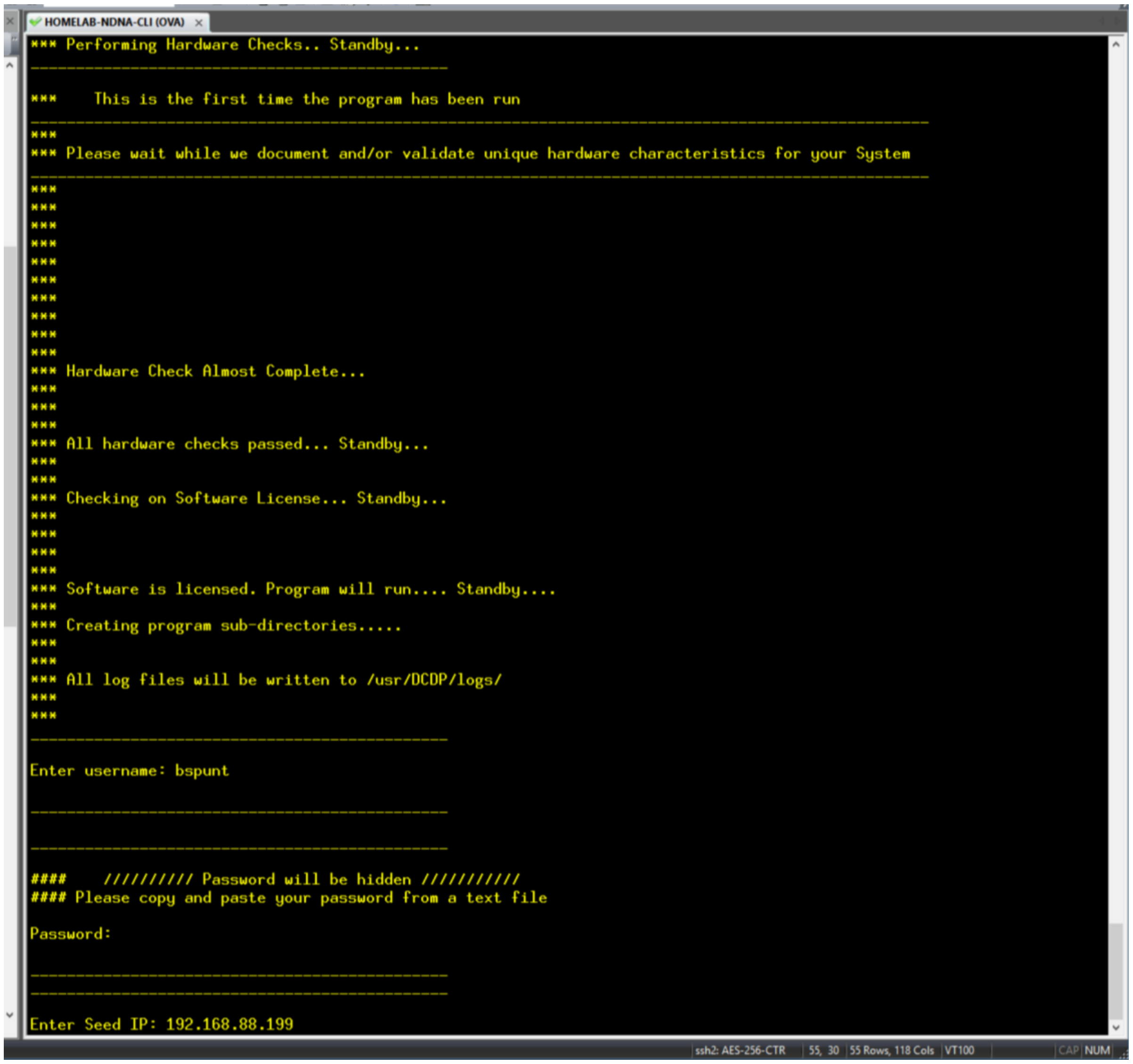


```
  -------------------------------------------------------------------------------------------------------------------

                 ALL YOUR CUSTOM DISCOVERY AND PROGRAMMING FILES WILL ALSO BE STORED IN YOUR DC FOLDERS

You still need to run any subsequent custom discovery or programming from the /usr/DCDP/* Directories (This is where yo

  -------------------------------------------------------------------------------------------------------------------

Enter The Company Code/Data_Center String NOW e.g. MIC:LA-DC: CCC:NYC-DC

Data-Center Name Meets Requirements... Continuing


** Creating Backups Directory /usr/Backups....



** This is a new Data Center....Creating CCC:NYC-DC directories....




  _   _   _   _   _   _   _   _   _    _   _   _   _   _   _   _
  _   _   _   _   _   _   _   _   _    _   _   _   _   _   _   _

     Welcome to the The Network Discovery N Automation Program
                          (NDNA) v17.02
   _   _   _   _   _   _   _     _   _   _   _   _   _   _   _   _
  / \ / \ / \ / \ / \ / \ / \   / \ / \ / \ / \ / \ / \ / \ / \ / \
 ( N | e | t | w | o | r | k ) ( D | i | s | c | o | v | e | r | y )
  \_/ \_/ \_/ \_/ \_/ \_/ \_/   \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/

            Copyright: Automate-The-Network LLC 2017
              http://www.automate-the-network.com
  _   _   _   _   _   _   _   _   _   _     _   _   _   _   _   _   _
 / \ / \ / \ / \ / \ / \ / \ / \ / \ / \   / \ / \ / \ / \ / \ / \ / \
( A | u | t | o | m | a | t | i | o | n ) ( P | r | o | g | r | a | m )
 \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/ \_/   \_/ \_/ \_/ \_/ \_/ \_/ \_/

            Hit Ctrl Z or Ctrl C to exit the program
  _   _   _   _   _   _   _   _   _    _   _   _   _   _   _   _
  _   _   _   _   _   _   _   _   _    _   _   _   _   _   _   _














                   Hit Enter To Continue
```

Hit enter again to continue.

Wait till it prompts you to enter:

- Username (You'll be prompted numerous times during discovery)
- Password (You'll be prompted numerous times during discovery)
- Seed IP



```
*** Performing Hardware Checks.. Standby...
----------------------------------------------

***    This is the first time the program has been run
-----------------------------------------------------------------------------------------------------
***
*** Please wait while we document and/or validate unique hardware characteristics for your System
-----------------------------------------------------------------------------------------------------
***
***
***
***
***
***
***
***
***
***
*** Hardware Check Almost Complete...
***
***
***
*** All hardware checks passed... Standby...
***
***
*** Checking on Software License... Standby...
***
***
***
***
*** Software is licensed. Program will run.... Standby....
***
*** Creating program sub-directories.....
***
***
*** All log files will be written to /usr/DCDP/logs/
***
***
----------------------------------------------

Enter username: bspunt

----------------------------------------------


----------------------------------------------

####    ////////// Password will be hidden ///////////
#### Please copy and paste your password from a text file

Password:

----------------------------------------------
----------------------------------------------

Enter Seed IP: 192.168.88.199
```

That's it! Then just wait for the program to complete. The entire process usually takes about 10 to 15 minutes for most Sites and/or Data Centers.

- You’ll get various progress updates through-out the process.
- When the program completes, you will see the following messages:

```
The total number of Good IP hosts (Total number of successful connections, e.g.
Authentication Success) is: 56
The total number of IOS IP hosts discovered is: 36
The total number of NXOS IP hosts discovered is: 20
The total number of L2-IOS IP hosts discovered is:26
The total number of L3-IOS IP hosts discovered is: 10
The total number of L2-NXOS IP hosts discovered is: 14
The total number of L3-NXOS IP hosts discovered is: 6
The total number of hosts with authentication errors is: 0
The total number of Incompatible SSH version hosts is: 0


##################################################################
  --------------------------------------------------------------
            Program has completed for CCC:NYC-DC

               Thank you for being patient

       You must now review all logs in /usr/DCDP/logs

           Follow instructions in the user guide
           To ensure the most complete discovery
  --------------------------------------------------------------
##################################################################
```

Post *Main-Program* Tasks

After the main program runs, you want to first go through all the logs.

The first two log files you want to look at are:

L2-L3-IOS.log
L2-L3-NXOS.log

If there are any authentication errors listed in the above log files, you will need to re-run the program again from the start.

If any other log files/part of the program had any SSH authentication issues on subsequent connections to nodes (e.g. on post-scripts that run after the main NDNA program), then you can incrementally update this information using a quick Manual update process. *This will be covered later.*

This is not the case with the L2-L3 IP lists process.

In fact, this point is a good Segway to go over some program architecture, so it's clear *why* any L2-L3 IP list authentication failure would necessitate the need to totally re-run the program on a site.

Then, we'll resume back to log file analysis.

L2-L3 IP Lists & Program Architecture Notes

The Main program uses SSH and Python automation on the network, and allot of Bash/Shell programming on the backend system to achieve its functionality.

When it connects to the network it uses CDP to discover Cisco Devices. It will attempt to connect to anything it can see via CDP. Due to this, it will attempt to connect to LWAPs, IP Phones, UCS FIs, WLCs, etc.

Due to the above fact, you WILL see authentication errors during the Main NDNA program discovery process, sometimes MANY authentication errors during discovery. You will see these messages in the log files and/or see this as you watch the program perform its discovery (E.g. see these messages printed back to the terminal).

This part of the program's main purpose is to document the Route/Switch environment.

It also uses the Paramiko module in Python which only supports SSHv2, so you will inevitably see "incompatible SSH version" messages in the log files and/or see this as you watch the program perform its discovery (E.g. see the messages printed back to the terminal).

Both are totally normal messages, and to be expected. We will go over recommended methodologies for filtering and getting through these issues and/or to correct these issues for any devices that are routers or switches.

The authentication errors might be due to numerous reasons:

1. Incorrect TACACS credentials on a device
2. Transient network conditions, e.g. issues with the Mgmt. plane of the device, congestion on the network, etc.
3. Cisco ACS server issues

Once the main discovery has run, it will report that the NDNA program has completed, *but the program continues to run*, starting up numerous python automation scripts to build L2-L3 IP lists, CDP neighbors files, MySQL DB inventories (Which are used to automatically generate diagrams), etc.

When the main discovery part of the program completes, it will go back out into the network and run commands that it uses to build L2-L3 IP lists for IOS/NXOS devices. We will go over all these files in a minute….

If you notice any authentication errors while it builds the L2-L3 IP lists (for either IOS or NXOS) then it will report to you both **a)** the IP, and **b)** it'll report that it will make a 2nd attempt to connect to that device (or devices). ***Wait for it to complete***. If, during the second attempt, it is not successful, we recommend at this point, hit CTRL Z to exit the program, wait 5 to 10 minutes and try to run the program again from the start. *This should be a rare occurrence, and/or should only happen due to transient network conditions.* First, let's describe what all the IP files are, ***then we'll get back to why the L2-L3 files must be accurate before continuing.***

Once the program has discovered the network, it builds the following IP files in the following folders (These files are moved to your DataCenter directory at the end of the program and deleted from /usr/DCDP/)

Directories
/usr/DataCenters/variable-DC-Name/DCDP/good-IPs
/usr/DataCenters/variable-DC-Name/DCDP/bad-IPs
/usr/DataCenters/variable-DC-Name/DCDP/Full-IP-List

See below for descriptions of what's in the folder

Files

Full-IP-List folder:
/usr/DataCenters/variable-DC-Name/DCDP/Full-IP-List/DCDP-ip-file.txt
These are all the IPs it could see via CDP

----
bad-IPs folder:
/usr/DataCenters/variable-DC-Name/DCDP/bad-IPs/Bad-IPs.txt

These are IPs it could not connect to and/or could not authenticate to. These IPs are “bad” from the program’s perspective. This could be due to invalid credentials, IP Phones, APs, SSHv1 devices, etc. (As stated earlier) – Later, we will go over ways to deduce what’s going on with the bad IPs efficiently, e.g. to be able to move any bad IPs into Good IPs (if you can get them set with proper TACACS, SSHv2 and they belong to a router or switch)

The Bad-IPs.txt file is the Full-IP-List minus the Good-IPs

----

good-IPs folder:
/usr/DataCenters/variable-DC-Name/DCDP/good-IPs/Good-IPs.txt
These are IPs it successfully connected to and authenticated to. These IPs are “good” from the program’s perspective. These are the sum-total of all IOS and NX-OS routers and switches it connected to.

/usr/DataCenters/variable-DC-Name/DCDP/good-IPs/IOS-IPs.txt
These are the IOS IPs it successfully connected to and authenticated to. These are a sub-set of the Good-IPs.txt

/usr/DataCenters/variable-DC-Name/DCDP/good-IPs/NX-OS-IPs.txt
These are the NX-OS IPs it successfully connected to and authenticated to. These are a sub-set of the Good-IPs.txt

The Good-IPs.txt are the sum-total of the NX-OS-IPs.txt and the IOS IPs.txt

Also, inside the good-IPs folder, you have the L2-L3 IP lists (Further broken down by IOS and NXOS)

/usr/DataCenters/variable-DC-Name/DCDP/good-IPs/L2-IOS-IPs.txt
These are the L2-IOS IPs. These are a sub-set of the IOS-IPs.txt

/usr/DataCenters/variable-DC-Name/DCDP/good-IPs/L3-IOS-IPs.txt These are the L3-IOS IPs. These are a sub-set of the IOS-IPs.txt

The sum-total of the L2 and L3 IOS IPs is = to the IOS-IPs.txt

/usr/DataCenters/variable-DC-Name/DCDP/good-IPs/L2-NX-OS-IPs.txt
These are the L2-NX-OS IPs. These are a sub-set of the NX-OS-IPs.txt

/usr/DataCenters/variable-DC-Name/DCDP/good-IPs/L3-NX-OS-IPs.txt
These are the L3-NX-OS IPs. These are a sub-set of the NX-OS-IPs.txt

The sum-total of the L2 and L3 NX-OS IPs is = to the NX-OS-IPs.txt

*OK, so now back to why the L2-L3 files *must* be accurate before continuing:*

If the L2-L3 IP lists are not accurate, then numerous subsequent scripts (for example, CDP neighbors, building enterprise wide L2-L3 IP lists, running custom discovery/custom programming at the site-specific level, etc.) all rely/assume the L2-L3 IP lists are correct.

If these are not correct (and/or have not been corrected), then all the previously mentioned program functions will not be accurate. *Therefore, it's critical to make sure this is correct or corrected before moving on.*

This is a rare occurrence, but none-the-less, you want to always pick up on it while the program runs, or catch it right away after the program runs by reviewing the log files (which we'll resume back to in the next section)

If there are any other authentication errors on any other subsequent scripts (CDP neighbors, All MySQL database inventory scripts), these can be incrementally updated using manual updates (taking just a minute or two after running the NDNA program – and covered in the chapter covering custom Python scripts), but the L2-L3 IP lists have to be 100% accurate first, so again, to be super-redundant, *if you get any authentication errors when these run, stop the program and start again from the beginning. If you don't check on this until after the program finishes and find while reviewing the log files that this didn't complete successfully, then just re-run that site/data-center again 100% e.g. just run the NDNA program (./DCDP.sh)*

As a last note, the reason this should be a rare occurrence is because the IP lists it uses to build the L2-L3 lists are the IOS and NX-OS IP lists, which are built from the Good-IPs.txt (Devices the program *just* discovered and successfully authenticated to a few minutes prior, e.g. the *Good-IPs*....)

Ok, sorry for the long interlude, now back to reviewing the log files….

Reviewing the Log Files

Once the program completes, the first thing to do is review the log files and the CDP neighbors' files. *<u>This is to make sure no "post-main-NDNA-program" scripts need to be re-run</u>*, and make sure you don't need to completely re-run the NDNA program *(If the L2-L3 IP Lists didn't complete error free)*

First, go into the log directory:
cd /usr/DCDP/logs or into /usr/DataCenters/variable-DC-Name/DCDP/logs

Tidbit of information: the significance of the acronym DCDP in the directory structure, program names, and log files has to do with the legacy name of the program, which was "The Data Center Discovery Program", hence all the references to "dcdp"

Review the files in the following order:

Log Files	Location	What to look for	Action to take
L2-L3-IOS.log	/usr/DCDP/logs	Authentication errors	Rerun the whole NDNA program if there are any
L2-L3-NX-OS.log	/usr/DCDP/logs	Authentication errors	Rerun the whole NDNA program if there are any
L2-L3-IOS-ERR.log	/usr/DCDP/logs	This file should be empty	If not empty, review errors. Rerun the whole NDNA program
L2-L3-NX-OS-ERR.log	/usr/DCDP/logs	This file should be empty	If not empty, review errors. Rerun the whole NDNA program
dcdp.log (Main NDNA program log)	/usr/DCDP/logs	Authentication errors	*See additional steps below*
		Incompatible SSH version	*See additional steps below*

Additional Steps: (for dcdp.log main NDNA program log issues)

You'll likely see authentication errors and SSHv1, e.g. messages in this log file that look like the following:

* Authentication Error for 10.2.2.2
* Incompatible SSH version. Paramiko requires SSHv2 on device 10.2.2.23

You can also view of summary of these authentication errors and Incompatible SSH version issues for The main NDNA program in the following files and folder:

/usr/DataCenters/<YOUR-SITE-NAME>/DCDP/stats/ auth-errors-IPs.txt
/usr/DataCenters/<YOUR-SITE-NAME>/DCDP/stats/ssh-version-errors-IPs.txt
/usr/DataCenters/<YOUR-SITE-NAME>/DCDP/stats/stats.txt

You can use the summary lists of IPs in the above files to quickly research these issues, e.g. manually try to log into these devices/get TACACS setup with your credentials on these devices, upgrade them to SSHv2 if possible, etc.

Log Files (Continued)	Location	What to look for	Action to take
sh-cdp-nei.log	/usr/DCDP/logs	Authentication errors (if any are seen, see "action to take") →	Run Manual Update script. See "*Manual-Updates" Section*
sh-cdp-nei.log-ERR.log	/usr/DCDP/logs	This file should be empty – If not, see "action to take" →	Run Manual Update script. See "*Manual-Updates" Section*
sh-cdp-nei-nxos.log	/usr/DCDP/logs	Authentication errors (if any are seen, see "action to take") →	Run Manual Update script. See "*Manual-Updates" Section*
sh-cdp-nei-nxos-ERR.log	/usr/DCDP/logs	This file should be empty – If not, see "action to take" →	Run Manual Update script. See "*Manual-Updates" Section*

Database Inventory Log Files	Location	What to look for	Action to take
CDP_INVENTORY.log	/usr/DCDP/logs	Authentication errors (if any are seen, see “action to take”) →	Run Manual Update script. See “*Manual-Updates” Section*
CDP_INVENTORY-ERR.log	/usr/DCDP/logs	This file should be empty – If not, “see action to take” →	Run Manual Update script. See “*Manual-Updates” Section*
IOS_INVENTORY.log	/usr/DCDP/logs	Authentication errors (if any are seen, see “action to take”) →	Run Manual Update script. See “*Manual-Updates” Section*
IOS_INVENTORY-ERR.log	/usr/DCDP/logs	This file should be empty – If not, see “action to take” →	Run Manual Update script. See “*Manual-Updates” Section*
NXOS_INVENTORY.log	/usr/DCDP/logs	Authentication errors (if any are seen, see “action to take”) →	Run Manual Update script. See “*Manual-Updates” Section*
NXOS_INVENTORY-ERR.log	/usr/DCDP/logs	This file should be empty – If not, see “action to take” →	Run Manual Update script. See “*Manual-Updates” Section*

For all above “Database Inventory Log Files”, another great place to always check/cross-reference is to go into:

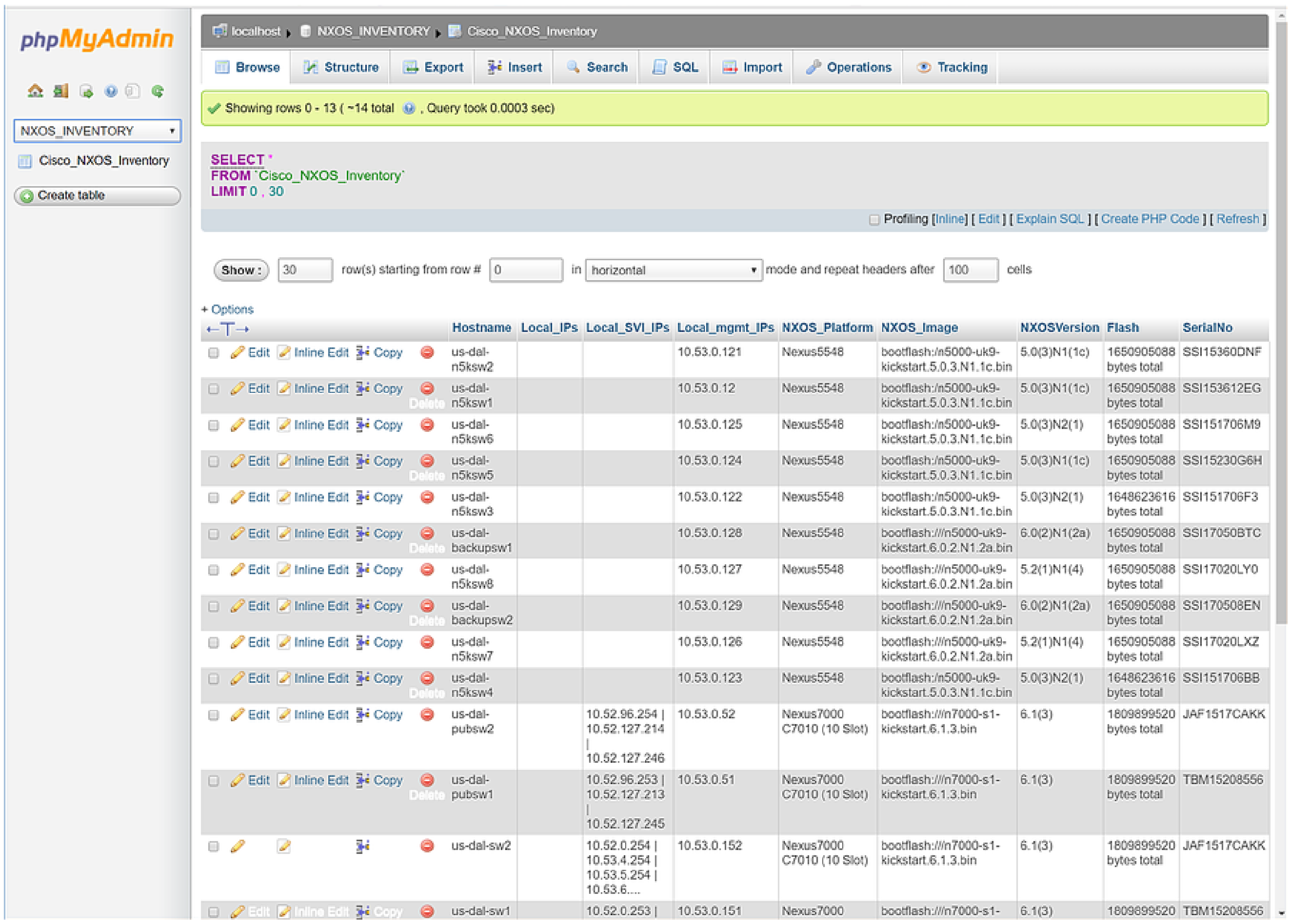
And view the MySQL Database inventories @

http://<YOUR-VM-IP>/phpmyadmin/index.php
Login username = root   password = dcdp

In phpMyAdmin, you can view the following database TABLES:

- ☐ CDP → CDP_Inventory Table
- ☐ IOS_INVENTORY → Cisco_IOS_Inventory Table
- ☐ NXOS_INVENTORY → Cisco_NXOS_Inventory Table

PhpMyAdmin screenshot below:



✓ Showing rows 0 - 13 (~14 total , Query took 0.0003 sec)

```
SELECT *
FROM `Cisco_NXOS_Inventory`
LIMIT 0 , 30
```

☐ Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from row # 0 in horizontal mode and repeat headers after 100 cells

+ Options

←T→	Hostname	Local_IPs	Local_SVI_IPs	Local_mgmt_IPs	NXOS_Platform	NXOS_Image	NXOSVersion	Flash	SerialNo
☐ Edit Inline Edit Copy	us-dal-n5ksw2			10.53.0.121	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N1(1c)	1650905088 bytes total	SSI15360DNF
☐ Edit Inline Edit Copy Delete	us-dal-n5ksw1			10.53.0.12	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N1(1c)	1650905088 bytes total	SSI153612EG
☐ Edit Inline Edit Copy	us-dal-n5ksw6			10.53.0.125	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N2(1)	1650905088 bytes total	SSI151706M9
☐ Edit Inline Edit Copy Delete	us-dal-n5ksw5			10.53.0.124	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N1(1c)	1650905088 bytes total	SSI15230G6H
☐ Edit Inline Edit Copy	us-dal-n5ksw3			10.53.0.122	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N2(1)	1648623616 bytes total	SSI151706F3
☐ Edit Inline Edit Copy Delete	us-dal-backupsw1			10.53.0.128	Nexus5548	bootflash:///n5000-uk9-kickstart.6.0.2.N1.2a.bin	6.0(2)N1(2a)	1650905088 bytes total	SSI17050BTC
☐ Edit Inline Edit Copy	us-dal-n5ksw8			10.53.0.127	Nexus5548	bootflash:///n5000-uk9-kickstart.6.0.2.N1.2a.bin	5.2(1)N1(4)	1650905088 bytes total	SSI17020LY0
☐ Edit Inline Edit Copy Delete	us-dal-backupsw2			10.53.0.129	Nexus5548	bootflash:///n5000-uk9-kickstart.6.0.2.N1.2a.bin	6.0(2)N1(2a)	1650905088 bytes total	SSI170508EN
☐ Edit Inline Edit Copy	us-dal-n5ksw7			10.53.0.126	Nexus5548	bootflash:///n5000-uk9-kickstart.6.0.2.N1.2a.bin	5.2(1)N1(4)	1650905088 bytes total	SSI17020LXZ
☐ Edit Inline Edit Copy Delete	us-dal-n5ksw4			10.53.0.123	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N2(1)	1648623616 bytes total	SSI151706BB
☐ Edit Inline Edit Copy	us-dal-pubsw2		10.52.96.254 | 10.52.127.214 | 10.52.127.246	10.53.0.52	Nexus7000 C7010 (10 Slot)	bootflash:///n7000-s1-kickstart.6.1.3.bin	6.1(3)	1809899520 bytes total	JAF1517CAKK
☐ Edit Inline Edit Copy Delete	us-dal-pubsw1		10.52.96.253 | 10.52.127.213 | 10.52.127.245	10.53.0.51	Nexus7000 C7010 (10 Slot)	bootflash:///n7000-s1-kickstart.6.1.3.bin	6.1(3)	1809899520 bytes total	TBM15208556
☐	us-dal-sw2		10.52.0.254 | 10.53.4.254 | 10.53.5.254 | 10.53.6....	10.53.0.152	Nexus7000 C7010 (10 Slot)	bootflash:///n7000-s1-kickstart.6.1.3.bin	6.1(3)	1809899520 bytes total	JAF1517CAKK
☐ Edit Inline Edit Copy	us-dal-sw1		10.52.0.253 |	10.53.0.151	Nexus7000	bootflash:///n7000-s1-	6.1(3)	1809899520	TBM15208556

Review Data-Center Folder

This is where all your data is kept after you've completed discovery, including once you start to run "site-specific" python automation (e.g. configurations are all written into the configs directory of the Data-Center)

Folder Name	Location	What's in it	Notes
bad-IPs	/usr/DataCenters/<variable DC Name>/DCDP/	IPs that could be seen via CDP but are either not routers or switches, or had issues connecting (AUTH or SSH issues)	See log files and log file section of this manual
bin	/usr/DataCenters/<variable DC Name>/DCDP/	Diagram-generation folder and xml file	This is where your final xml file used in generating diagrams is found.
cdp_files	/usr/DataCenters/<variable DC Name>/DCDP/	This folder contains IPs seen at each "level" during discovery.	This folder can be ignored and is just informational, and there for tech support (if needed)
CDP-INVENTORY	/usr/DataCenters/<variable DC Name>/DCDP/	This folder contains the CDP inventory MySQL backup and the xml of the CDP_Inventory	This xml is combined with the IOS/NXOS inventories to create the final xml file used to automatically generate diagrams
cdp-neighbors	/usr/DataCenters/<variable DC Name>/DCDP/	This folder contains the CDP neighbors' files to view in sublime text for both IOS and NXOS.	Use these files to assist with tracking down bad-IPs to determine whether it's a device you are interested in or not, e.g. if it's an IP Phone, AP, WLC, etc. or a router or switch.
configs	/usr/DataCenters/<variable DC Name>/DCDP/	This is where all configs are written, (dated by the minute) when you run python automation against a datacenter.	Separate files are written for each node, whether you are performing discovery or programming the network.

Folder Name (Continued)	Location	What's in it	Notes
Full-IP-List	/usr/DataCenters/<variable DC Name>/DCDP/	All IPs that could be seen via CDP (Sum-total of the good and bad IPs)	
Good-IPs	/usr/DataCenters/<variable DC Name>/DCDP/	All the various good IPs – see this previous section for info on all files in this directory	
Hostname-to-IPs	/usr/DataCenters/<variable DC Name>/DCDP/	This folder contains IOS and NXOS csv spreadsheets with hostname to IPs of all the good IPs	
Inventory	/usr/DataCenters/<variable DC Name>/DCDP/	This folder contains the IOS and NXOS inventory MySQL backups and the xmls of the IOS and NXOS Inventories	These xmls are combined with the CDP inventory to create the final xml file used to automatically generate diagrams
logs	/usr/DataCenters/<variable DC Name>/DCDP/	This folder contains a copy of the /usr/DCDP/logs directory	Can view logs from the Data-Center folder or from /usr/DCDP
snmpwalk	/usr/DataCenters/<variable DC Name>/DCDP/	This folder is empty, but will populate if you use the custom snmpwalk on the Data-Center	It will store hostname to IP spreadsheets of any SNMP walk's run on bad-IPs

Manual Updates

If your log files show you need to run any manual updates, you can run them using the instructions below:

From a command line, go into the Manual Updates folder:

```
cd /usr/DCDP/bin/python_custom_scripts/Manual-Updates
```

list the contents, using the ls command.

you'll see the following folders:

```
cdp-neighbors  db_inventory  L2-L3-IP-Lists
```

Updating CDP-Neighbors

Change into the following directory:

```
cd /usr/DCDP/bin/python_custom_scripts/Manual-Updates/cdp-neighbors
```

list the contents, using the ls command.

you'll see the following:

```
cdp-neighbors-ios-manual.sh  cdp-neighbors-nxos-manual.sh
```

If you need to update the regular cdp-neighbors file, issue the following command:

```
./cdp-neighbors-ios-manual.sh
```

This will launch the Python script on the backend, SSH to the device using the correct IPs from that Datacenter, run the correct commands, then launch another shell script to build the file for you in the Datacenter that you specify. *Pay attention to the prompts.*

If you need to update the nxos cdp-neighbors file, issue the following command:

```
./cdp-neighbors-nxos-manual.sh
```

This will launch the Python script on the backend, SSH to the device using the correct IPs from that Datacenter, run the correct commands, then launch another shell script to build the file for you in the Data-Center that you specify. *Pay attention to the prompts.*

After either of these run (or if you run both of them), go into the Data-Center folder and check to make sure there were no authentication errors (the file will show this)

You can also review the regular logs in the /usr/DCDP/logs folder too

Updating DB-inventories

Change into the following directory:

cd /usr/DCDP/bin/python_custom_scripts/Manual-Updates/db_inventory/

list the contents, using the ls command.

you'll see the following directories:
CDP
NXOS
IOS

And the following files:
cdp_db_inventory_update.sh
generate-xml-diagram-file.sh
ios_db_inventory_update.sh
nxos_db_inventory_update.sh
README.txt

If you need to update the CDP DB Inventory, issue the following command:
./cdp_db_inventory_update.sh

If you need to update the IOS DB Inventory, issue the following command:
./ios_db_inventory_update.sh

If you need to update the NXOS DB Inventory, issue the following command:
./nxos_db_inventory_update.sh

These scripts will launch the Python script on the backend, SSH to the device using the correct IPs from that Datacenter, run the correct commands, then launch another shell script to build a new xml file for you in the Datacenter that you specify. *Pay attention to the prompts.*

READ THE README.txt to properly re-generate the final xml file, which regenerates an xml that combines all three xmls into one (*Which the NDNA diagram generator looks for*)

After any of these run, go into the Data-Center folder to check on the date and time of the new .SQL and xml files to make sure you can see the new files. Also, go into the MySQL DB to review it. You will also get feedback from the terminal as you run the program.

<Review Databases>
http://<YOUR-VM-IP>/phpmyadmin/index.php
Login username = root   password = dcdp

Lastly, go into the */usr/DCDP/logs folder to review the logs to see if there were any authentication errors. If there were, rerun it again until it's successful*

Updating L2-L3 IP Lists

While we recommend just re-running the entire NDNA program if you run into any authentication issues with L2-L3 IP lists, you do always have the option of manually updating these lists.

You might or might not need to update the CDP neighbors file, and all the DB inventories, but these are not L2-L3 dependent.

Review Databases

Always check/cross-reference the actual databases to make sure it hashes with your discovery and IP lists, e.g. the numbers are correct. For example, if you show 20 NXOS hosts and 30 IOS hosts were discovered in a Data-Center, then you should show 20 hosts in the NXOS_Inventory table, and 30 hosts in the IOS_Inventory table.

Lastly, you should show 50 hosts in the CDP_Inventory table (This Database table encompasses both IOS and NXOS, e.g. all hosts from a site)

To look at the tables in the DBs, go into:

phpMyAdmin

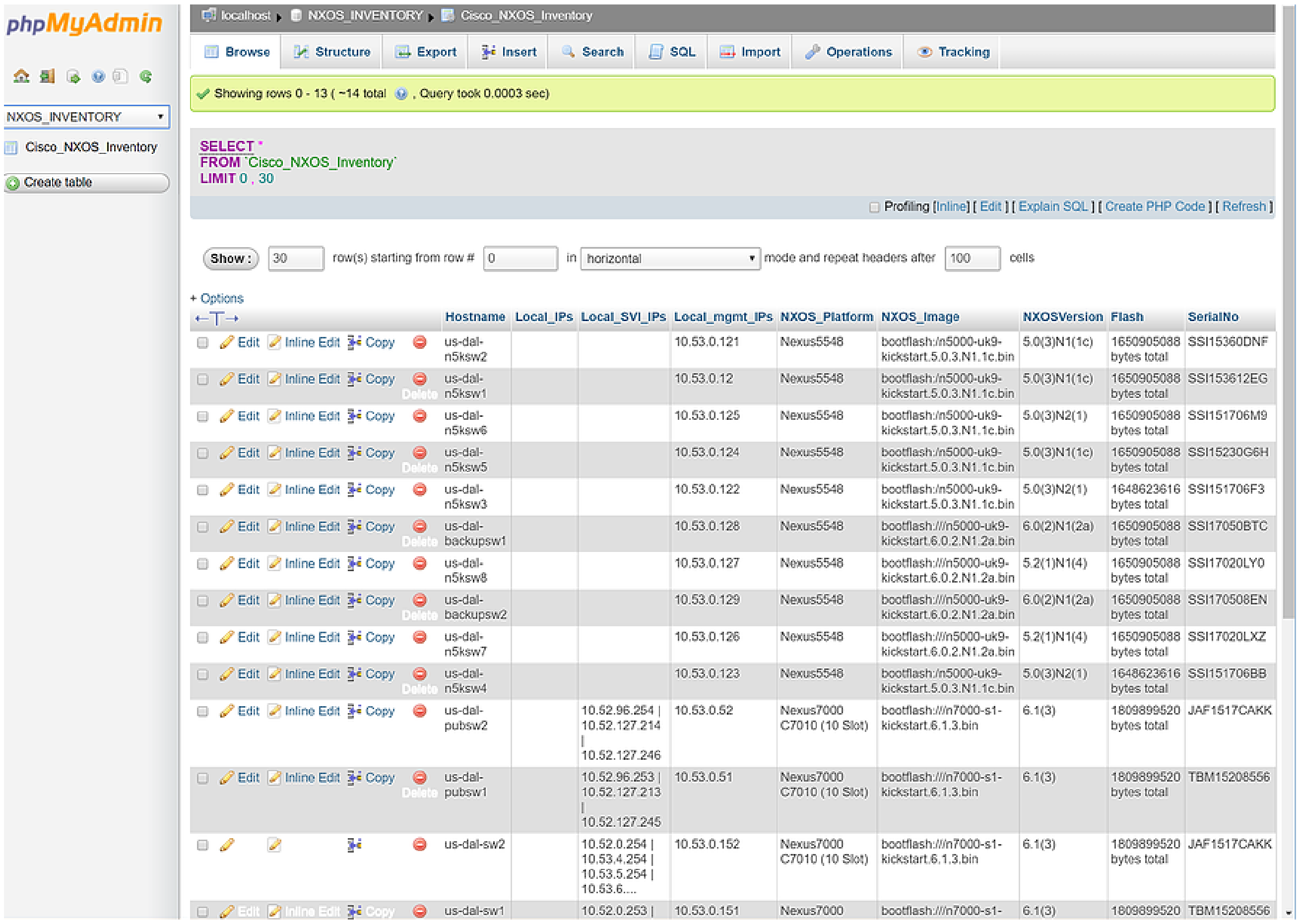
And view the MySQL Database inventories @

http://<YOUR-VM-IP>/phpmyadmin/index.php
Login username = root   password = dcdp

In phpMyAdmin, you can view the following database TABLES:

- ☐ CDP → CDP_Inventory Table
- ☐ IOS_INVENTORY → Cisco_IOS_Inventory Table
- ☐ NXOS_INVENTORY → Cisco_NXOS_Inventory Table

DB Inventories will look like what's shown on the next page:



Showing rows 0 - 13 (~14 total , Query took 0.0003 sec)

```
SELECT *
FROM `Cisco_NXOS_Inventory`
LIMIT 0 , 30
```

Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from row # 0 in horizontal mode and repeat headers after 100 cells

	Hostname	Local_IPs	Local_SVI_IPs	Local_mgmt_IPs	NXOS_Platform	NXOS_Image	NXOSVersion	Flash	SerialNo
Edit Inline Edit Copy	us-dal-n5ksw2			10.53.0.121	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N1(1c)	1650905088 bytes total	SSI15360DNF
Edit Inline Edit Copy Delete	us-dal-n5ksw1			10.53.0.12	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N1(1c)	1650905088 bytes total	SSI153612EG
Edit Inline Edit Copy	us-dal-n5ksw6			10.53.0.125	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N2(1)	1650905088 bytes total	SSI151706M9
Edit Inline Edit Copy Delete	us-dal-n5ksw5			10.53.0.124	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N1(1c)	1650905088 bytes total	SSI15230G6H
Edit Inline Edit Copy	us-dal-n5ksw3			10.53.0.122	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N2(1)	1648623616 bytes total	SSI151706F3
Edit Inline Edit Copy Delete	us-dal-backupsw1			10.53.0.128	Nexus5548	bootflash:///n5000-uk9-kickstart.6.0.2.N1.2a.bin	6.0(2)N1(2a)	1650905088 bytes total	SSI17050BTC
Edit Inline Edit Copy	us-dal-n5ksw8			10.53.0.127	Nexus5548	bootflash:///n5000-uk9-kickstart.6.0.2.N1.2a.bin	5.2(1)N1(4)	1650905088 bytes total	SSI17020LY0
Edit Inline Edit Copy Delete	us-dal-backupsw2			10.53.0.129	Nexus5548	bootflash:///n5000-uk9-kickstart.6.0.2.N1.2a.bin	6.0(2)N1(2a)	1650905088 bytes total	SSI170508EN
Edit Inline Edit Copy	us-dal-n5ksw7			10.53.0.126	Nexus5548	bootflash:///n5000-uk9-kickstart.6.0.2.N1.2a.bin	5.2(1)N1(4)	1650905088 bytes total	SSI17020LXZ
Edit Inline Edit Copy Delete	us-dal-n5ksw4			10.53.0.123	Nexus5548	bootflash:/n5000-uk9-kickstart.5.0.3.N1.1c.bin	5.0(3)N2(1)	1648623616 bytes total	SSI151706BB
Edit Inline Edit Copy	us-dal-pubsw2		10.52.96.254 | 10.52.127.214 | 10.52.127.246	10.53.0.52	Nexus7000 C7010 (10 Slot)	bootflash:///n7000-s1-kickstart.6.1.3.bin	6.1(3)	1809899520 bytes total	JAF1517CAKK
Edit Inline Edit Copy Delete	us-dal-pubsw1		10.52.96.253 | 10.52.127.213 | 10.52.127.245	10.53.0.51	Nexus7000 C7010 (10 Slot)	bootflash:///n7000-s1-kickstart.6.1.3.bin	6.1(3)	1809899520 bytes total	TBM15208556
	us-dal-sw2		10.52.0.254 | 10.53.4.254 | 10.53.5.254 | 10.53.6....	10.53.0.152	Nexus7000 C7010 (10 Slot)	bootflash:///n7000-s1-kickstart.6.1.3.bin	6.1(3)	1809899520 bytes total	JAF1517CAKK
Edit Inline Edit Copy	us-dal-sw1		10.52.0.253 |	10.53.0.151	Nexus7000	bootflash:///n7000-s1-	6.1(3)	1809899520	TBM15208556

Review CDP-neighbors-files

This Chapter covers reviewing the CDP neighbors file. These are both flat files (.txt) that are created when you run the main NDNA program. There's one for IOS and one for NXOS.

It can be used for numerous things:

1. Help add additional detail to diagrams you create using the diagram generator, e.g. add in anything else you want to add in that can be seen via CDP, e.g. IP Phones, APs, etc.
2. Provide documentation to a customer on an engagement.
3. Troubleshooting aid
4. Your own documentation for the site.
5. Identify BAD IPs – what they are. See the "Analyzing BAD IPs" section for more information.

You should only view it in a real text editor, e.g. Sublime Text or Notepad++ etc.

Analyzing BAD IPs

This Chapter covers analyzing the Bad-IPs.txt file.

There are a few reasons you want to do this, some of which are listed below:

1. Determine what these devices are, e.g. are they Routers or Switches? And if so, you can see about correcting the issues with them (Either Authentication issues, or SSH connection issues), and moving the IPs into the Good-IPs.txt for the site. This could be a manual process, or you could just re-run NDNA again. If it's a large number of devices, you'd just rerun NDNA again, since you'll also want to update the IP/Hostname CSV files, L2/L3 Lists, etc. If it's one or two devices, you can manually update items needed, but if it's more than that, if you now convert bad IPs to good IPs (by configuring them with TACACS access for example or configuring them with SSHv2), it's recommended to just run NDNA Discovery on the site/Data-Center again. This will automate the process, so all your data is accurate (L2-L3 IP lists, csv files, etc.)
2. You want to determine what the other devices are, e.g. you might be interested in placing these devices on the diagram and/or document these devices in spreadsheets, etc.

Recommended approach for identifying devices that are in the Bad-IPs.txt file:

1. Review the CDP neighbors' files, and do a "find" on these IPs.

Doing a find on IPs that are in the Bad-IPs.txt file.

2. *Review your Stats file.* This will tell you the IPs of all devices that had authentication errors, and all the devices that did not meet the SSHv2 requirements. This way you do ***not*** have to manually track these IPs down in your log files.

   /usr/DataCenters/<your variable DC name>/DCDP/stats

3. If you have SNMP read access to the network, you can run the automation program from this directory:

   /usr/DCDP/bin/snmpwalk_add_on_app-for-bad-IPs

   Run the following command:
   ./ snmpwalk-on-BAD-IPs.sh

This will perform an SNMP walk on the Bad-IPs and create a hostname-to-IP csv spreadsheet for you on all devices it gets SNMP read access to.

The output file will be saved to:
/usr/DataCenters/*<your variable DC name>*/DCDP/snmpwalk/snmpwalk-BAD-IPs-Hostname-Platform.csv

This can assist identifying the devices quickly and provides you with additional data on the site to provide for the customer and/or have for your own inventory. This can speed up the process of identifying hosts you're interested in or not interested in, saving you time manually searching for these IPs in the CDP Neighbors files.

Validate Final XML File

Go into the following directory:

/usr/DataCenters/<variable DC Name>/DCDP/bin/diagram-generation

And make sure you see a file with the following filename:

<Your-DC-Variable-Name>-diagram-generation.xml

This is the xml file you'll use to generate diagrams in Microsoft Visio.

Copy this file down to your Windows machine where you have the NDNA Diagram generator installed.

Remember to properly shutdown the VM (Virtual Machine) properly, e.g. don't just shutdown from the VMware or VirtualBox interface. You must shutdown from the actual Linux CLI using the following command:

shutdown -h now

You can leave it up and running always if you'd like, even when putting your computer to sleep. Just make sure to properly shutdown if you are going to shutdown your host computer/laptop, or Mac

The NDNA Diagram Generator

The NDNA diagram generator is provided by:

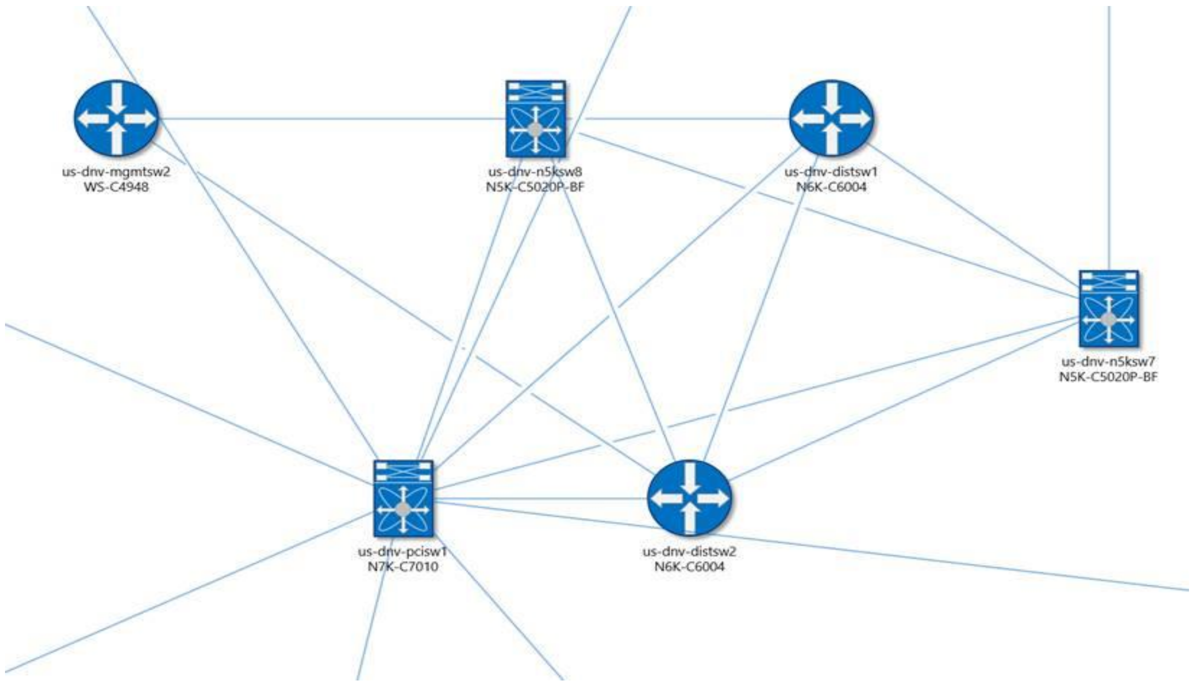
Automate The Network LLC
Get on the road to automation

It was jointly developed by Chris Roth – AKA "The Visio Guy" http://www.visguy.com/ and Brett M. Spunt

Credits:
Brett M. Spunt: Design & Overall App Framework, Meta-Data & App Data that gets fed into the Diagram Generator –
CCIE #12745
Lead Software Developer @ Automate-the-Network LLC (Author of the Main NDNA program)

Chris Roth: NDNA Diagram Generator Coding and additional Design - All Windows side programming (e.g. Author: The NDNA Diagram Generator Visio Plugin)

VISIO GUY



Installing the NDNA Diagram Generator

This is self-explanatory, for example, if you have installed a windows application, that’s all there is to it.

Using the NDNA Diagram Generator

See the following instructional videos:

https://youtu.be/wMfcm0gUm00

https://youtu.be/v38aHgkzvL8

https://youtu.be/sGhG3Ozy3F0

https://youtu.be/XOAdLYDDfi8

https://youtu.be/bd8ny2VwszM

NDNA to Cacti Integration (NTC)

After you've discovered Data-Centers/Sites using NDNA, you can now easily build all of these sites devices into Cacti (IOS devices only) using automation, giving you an almost instant NMS solution (Network Monitoring System). This includes using automation to both build in devices, AND build bandwidth graphs for devices. Lastly, the automation also builds an organized heading on the Cacti tree with the Company/Site name and places all graphs onto that heading of the Cacti graph tree.

NTC Only supports importing AND graphing IOS devices, not NXOS devices.

The VM OVA installation, login credentials (root login, and PhpMyadmin) are the same as the NDNA VM. Refer to that documentation for any setup/installation details

For more info and a full instructional video/demo video, see the direct link to the video below:
https://youtu.be/uZXgzJjyRPg

Virtual Machine | Application Details

A ready built VM OVA is provided by automate the network that has the following installed:

- Cacti Version 0.8.8a
- Thold plugin (For email alerts)
- Settings plugin
- Spine Poller
- PhpMyadmin is installed too.

Using Your Own Cacti System

While we provide you with everything you need, you also have the option of using your own existing Cacti System, just follow these instructions to prep it for use with the NTC (NDNA to Cacti Integration):

1. Create the following directories on your Cacti Server:

```
mkdir /usr/share/cacti/cli/tmp
mkdir /usr/share/cacti/cli/NDNA
```

2. Copy the following file from the NDNA Cacti NMS server into the same path on your Cacti Server, and make it executable: /usr/share/cacti/cli/NDNA/NDNA-build-graphs-into-cacti.bash

```
chmod 755 /usr/share/cacti/cli/NDNA/NDNA-build-graphs-into-cacti.bash
```

3. Copy the following file from the NDNA Cacti NMS server into the same path on your Cacti Server, and make executable: /usr/share/cacti/cli/cacti-hosts-before-import.bash
```
chmod 755 /usr/share/cacti/cli/cacti-hosts-before-import.bash
```

4. You must have SSH enabled on your Cacti Server and allow root log in via SSH

5. Create the following Heading in Cacti under your top-level Tree:(case sensitive)
NDNA-Internetwork
(sorting type Alphabetic)

6. Make sure your "Cisco Router" Host template has the following:

Associated Graph Templates
1) Cisco - CPU Usage Delete
2) Interface - Errors/Discards Delete
3) Interface - Non-Unicast Packets Delete
4) Interface - Traffic (bits/sec, Total Bandwidth)
5) Interface - Unicast Packets

Associated Data Queries
1) SNMP - Get Processor Information
2) SNMP - Interface Statistics

Please enter your Cacti user name and password below:

User Name: admin
Password: •••••

NDNA Automation

This Chapter covers all the custom automation features of the NDNA program, including the following sections:

1. Site Specific Automation
2. Enterprise Wide Automation
3. Vendor Neutral Automation
4. BYOI (Bring Your Own IP List) Automation
5. Creative Use of NDNA – Use it Your Way

All NDNA automation programs are run from subfolders @:
/usr/DCDP/bin/python_custom_scripts

Design & Architecture information

1. *Uses the Python Threading Module*
2. *Supports a maximum of 300 threads at a time. So, if you have more than 300 IPs in an IP list (highly unlikely in site specific/DC related IP lists, but VERY likely in enterprise wide IP lists) – then, you must edit your IP file to run 300 IPs at a time. For example if your enterprise switch IP list has 1000 IPs in it, run 300, 300, 300, then 100, e.g. four times to encompass all 1000 IPs. This still only takes minutes to do.*
3. *While the program runs, it throttles threads from going out into the network to 25 at a time, with a 20 second interval before running the next 25 threads. This helps to assist with TACACS being overloaded*
4. *No 64K limit in function return buffer in Paramiko for NDNA code!*
5. *In addition to threading, all output is written to separate text files for each node, with the name of the IP address in the name of the file. This assists with creative use of the NDNA program, as you will see in the section* "Creative Use of NDNA"*. Files are written for each node, regardless if you are sending show commands (e.g. pulling information from the network) or programming the network (sending information to the network). This represents a big differentiator for the NDNA program, e.g threading PLUS writing the output to separate text files for each node.*

Site-Specific Automation (AKA Data-Centers)

This Chapter covers Site Specific Automation using the IP Lists created during the NDNA program network discovery.

All "Site Specific" automation will prompt you for the Data-Center you want to run the automation on.

It will then run the automation on the IP lists from that Data-Center. For example, if you run it on the IOS IPs, it will look at the IOS IP list inside the Data-Center you specify and run the automation on those nodes.

The only thing you need to change and/or configure before you run the program is:
Input the commands you want to send to the devices via the "ssh_custom_<variable>_commands.txt" file.

You'll see there's already the command "terminal len 0" – DO NOT REMOVE THIS. This is needed to ensure correct terminal length, regardless of the output.

All "Site Specific" automation programs are run from:
/usr/DCDP/bin/python_custom_scripts/site-specific

You can always hit the Tab key to finish a command in Linux!
You can always copy and paste commands from this user guide into the CLI

File list from this directory:
custom-ios-nxos-python-script.py
custom-ios-python-script.py
custom-IOS-Routers-python-script.py
custom-IOS-Switches-python-script.py
custom-nxos-python-script.py
custom-NXOS-Routers-python-script.py
custom-NXOS-Switches-python-script.py
README.txt
ssh_custom_ios_commands.txt
ssh_custom_ios_nxos_commands.txt
ssh_custom_IOS_Routers_commands.txt
ssh_custom_IOS_Switches_commands.txt
ssh_custom_nxos_commands.txt
ssh_custom_NXOS_Routers_commands.txt
ssh_custom_NXOS_Switches_commands.txt

You can always refer to the README.txt in this directory as well….

IOS

To run automation on all IOS IPs for a Site/Data-Center:

1. Open up the following file via WinSCP (recommended), CyberDuck, or Nano if you are comfortable with Linux.
   ssh_custom_ios_commands.txt

2. Again, leave the command "terminal length 0" in the file.

3. Input the commands you want to send to your devices. That's it!

4. *Run the program*
   cd /usr/DCDP/bin/python_custom_scripts/site-specific
   ./custom-ios-python-script.py
   Follow the prompts….

5. Go into the following directory to view all your files after automation completes:
   /usr/DataCenters/<your variable-DC-Name>/DCDP/configs

All files will be dated by the minute, so as long as you wait at least one minute between automation runs, you won't overwrite previous files.

NXOS

To run automation on all NXOS IPs for a Site/Data-Center:

1. Open up the following file via WinSCP (recommended), CyberDuck, or Nano if you are comfortable with Linux.
   ssh_custom_nxos_commands.txt

1. Again, leave the command "terminal length 0" in the file.

2. Input the commands you want to send to your devices. That's it!

3. *Run the program*
   cd /usr/DCDP/bin/python_custom_scripts/site-specific
   ./custom-nxos-python-script.py
   Follow the prompts….

4. Go into the following directory to view all your files after automation completes:
   /usr/DataCenters/<your variable-DC-Name>/DCDP/configs

All files will be dated by the minute, so as long as you wait at least one minute between automation runs, you won't overwrite previous files.

IOS-NXOS (All IPs in a DC)

To run automation on all IPs for a Site/Data-Center:

1. Open up the following file via WinSCP (recommended), CyberDuck, or Nano if you are comfortable with Linux.
   ssh_custom_ios_nxos_commands.txt

1. Again, leave the command "terminal length 0" in the file.

2. Input the commands you want to send to your devices. That's it!

3. *Run the program*
   cd /usr/DCDP/bin/python_custom_scripts/site-specific
   ./custom-ios-nxos-python-script.py
   Follow the prompts….

4. Go into the following directory to view all your files after automation completes:
   /usr/DataCenters/<your variable-DC-Name>/DCDP/configs

All files will be dated by the minute, so as long as you wait at least one minute between automation runs, you won't overwrite previous files.

L2 IOS

To run automation on all L2-IOS IPs for a Site/Data-Center:

1. Open up the following file via WinSCP (recommended), CyberDuck, or Nano if you are comfortable with Linux.
   ssh_custom_IOS_Switches_commands.txt

1. Again, leave the command "terminal length 0" in the file.

2. Input the commands you want to send to your devices. That's it!

3. *Run the program*
   cd /usr/DCDP/bin/python_custom_scripts/site-specific
   ./custom-IOS-Switches-python-script.py
   Follow the prompts….

4. Go into the following directory to view all your files after automation completes:
   /usr/DataCenters/<your variable-DC-Name>/DCDP/configs

All files will be dated by the minute, so as long as you wait at least one minute between automation runs, you won't overwrite previous files.

L3 IOS

To run automation on all L3-IOS IPs for a Site/Data-Center:

1. Open up the following file via WinSCP (recommended), CyberDuck, or Nano if you are comfortable with Linux.
   ssh_custom_IOS_Routers_commands.txt

1. Again, leave the command "terminal length 0" in the file.

2. Input the commands you want to send to your devices. That's it!

3. *Run the program*
   cd /usr/DCDP/bin/python_custom_scripts/site-specific
   ./custom-IOS-Routers-python-script.py
   Follow the prompts….

4. Go into the following directory to view all your files after automation completes:
   /usr/DataCenters/<your variable-DC-Name>/DCDP/configs

All files will be dated by the minute, so as long as you wait at least one minute between automation runs, you won't overwrite previous files.

L2 NXOS

To run automation on all L2-NXOS IPs for a Site/Data-Center:

1. Open up the following file via WinSCP (recommended), CyberDuck, or Nano if you are comfortable with Linux.
   ssh_custom_NXOS_Switches_commands.txt

1. Again, leave the command "terminal length 0" in the file.

2. Input the commands you want to send to your devices. That's it!

3. *Run the program*
   cd /usr/DCDP/bin/python_custom_scripts/site-specific
   ./custom-NXOS-Switches-python-script.py
   Follow the prompts….

4. Go into the following directory to view all your files after automation completes:
   /usr/DataCenters/<your variable-DC-Name>/DCDP/configs

All files will be dated by the minute, so as long as you wait at least one minute between automation runs, you won't overwrite previous files.

L3 NXOS

To run automation on all L3-NXOS IPs for a Site/Data-Center:

1. Open up the following file via WinSCP (recommended), CyberDuck, or Nano if you are comfortable with Linux.
   ssh_custom_NXOS_Routers_commands.txt

1. Again, leave the command “terminal length 0” in the file.

2. Input the commands you want to send to your devices. That’s it!

3. *Run the program*
   cd /usr/DCDP/bin/python_custom_scripts/site-specific
   ./custom-NXOS-Routers-python-script.py
   Follow the prompts….

4. Go into the following directory to view all your files after automation completes:
   /usr/DataCenters/<your variable-DC-Name>/DCDP/configs

All files will be dated by the minute, so as long as you wait at least one minute between automation runs, you won’t overwrite previous files.

Enterprise Wide Automation

This Chapter covers Enterprise-Wide Automation using IP lists you create. These lists are built from all the sites/Data-Centers you have discovered using the NDNA program.

- These are built on a *per company* basis.

- They can also be built on a *subset-of-a-company* too (depending on your naming convention)

You first create the IP lists – *This is a one-time process*, or as often as you want to recreate/update these IP lists, e.g. if you discover additional sites/DCs - *(Procedure is covered in the next section)*

Then, you move the IPs into the proper file. *(Procedure is covered in the next section)*

Finally, you input commands you want to send to the devices.

Input the commands you want to send to the devices via the "ssh_custom_<variable>_commands.txt" file. *(Procedure is covered in the next section)*

You'll see there's already the command "terminal len 0" – DO NOT REMOVE THIS. This is needed to ensure correct terminal length, regardless of the output length.

It will then run the automation on the IP list covering all devices enterprise wide, either for NXOS, IOS, or NXOS/IOS combined (Depends on the IP list, e.g. list of IPs you move into IP file.)

Enterprise automation is broken down by Routers or Switches (L2 or L3 devices). L3 devices (from the perspective of the program) are devices found to be running a dynamic routing protocol, so a device configured ONLY with static routing is not considered an L3 device (even though it is, just not from the perspective of the program).

The L2 "enterprise-wide" automation program is run from:
/usr/DCDP/bin/python_custom_scripts/enterprise-wide-switches

The L3 "enterprise-wide" automation program is run from:
/usr/DCDP/bin/python_custom_scripts/enterprise-wide-routers

Information about the NDNA program automation (Provided again)

1. *Uses the Python Threading Module*
2. *Maximum of 300 threads at a time. So, if you have more than 300 IPs in an IP list (highly unlikely in site specific/DC related IP lists, but VERY likely in enterprise wide IP lists) – then, you must edit your IP file to run 300 IPs at a time. For example, if your enterprise switch IP list has 1000 IPs in it, run 300, 300, 300, then 100, e.g. four times to encompass all 1000 IPs. This still only takes minutes to do.*
3. *While the program runs, it throttles threads from going out into the network to 25 at a time, with a 20 second interval before running the next 25 threads. This helps to assist with TACACS being overloaded*
4. *In addition to threading, all output is written to separate text files for each node, with the name of the IP address in the name of the file. This assists with creative use of the NDNA program, as you will see in the section "*Creative Use of NDNA*". Files are written for each node, regardless if*

you are sending show commands (e.g. pulling information from the network) or programming the network (sending information to the network). This represents a big differentiator for the NDNA program, e.g threading PLUS writing the output to separate text files for each node.

Generating L2 Enterprise Wide IP Lists

To generate your L2 IP list enterprise wide for a company (or subset of a company), take the following steps:

Enter the following command:

```
cd /usr/enterprise-wide-switches/Create-Enterprise-IP-Lists-Scripts/
```

list the directory using the ls command:

```
create-enterprise-ios-nxos-switches-list.sh
create-enterprise-ios-switches-list.sh
create-enterprise-nxos-switches-list.sh
```

Run the script to create the three lists above. One is for IOS/NXOS combined. One is NXOS only, and one is IOS only.

Examples:

```
./create-enterprise-ios-nxos-switches-list.sh
./create-enterprise-ios-switches-list.sh
./create-enterprise-nxos-switches-list.sh
```

You'll be prompted for your three-character company code now. This is how you'll build IP lists per company or subset of a company.

Once you've generated your list, go into the following directory to view them:

```
cd /usr/enterprise-wide-switches/Created-IP-Lists/
```

List the directory using the ls command:

```
ls
CCC-enterprise-wide-ios-nxos-switches-Sep-06-17.txt
CCC-enterprise-wide-ios-switches-Sep-06-17.txt
CCC-enterprise-wide-nxos-switches-Sep-06-17.txt
```

When we get to the sections on running the automation, we'll go over how we use these IP lists, so the automation program knows about them.

Generating L3 Enterprise Wide P Lists

To generate your L3 IP list enterprise wide for a company (or subset of a company), take the following steps:

Enter the following command:

```
cd /usr/enterprise-wide-routers/Create-Enterprise-IP-Lists-Scripts/
```

list the directory using the ls command:

```
create-enterprise-ios-nxos-routers-list.sh
create-enterprise-ios-routers-list.sh
create-enterprise-nxos-routers-list.sh
```

Run the script to create the three lists above. One is for IOS/NXOS combined. One is NXOS only, and one is IOS only.

Examples:

```
./create-enterprise-ios-nxos-routers-list.sh
./create-enterprise-ios-routers-list.sh
./create-enterprise-nxos-routers-list.sh
```

You'll be prompted for your three-character company code at this time. This is how you'll build IP lists per company or subset of a company.

Once you've generated your list, go into the following directory to view them:

```
cd /usr/enterprise-wide-routers/Created-IP-Lists/
```

List the directory using the ls command:

```
ls
CCC-enterprise-wide-ios-nxos-routers-Sep-06-17.txt
CCC-enterprise-wide-ios-routers-Sep-06-17.txt
CCC-enterprise-wide-nxos-routers-Sep-06-17.txt
```

When we get to the sections on running the automation, we'll go over how we use these IP lists, so the automation program knows about them.

Running Enterprise-Wide Automation on Routers

To run automation on an enterprise wide IP list of routers:

Copy IPs from the "Created-IP-Lists " folder located @:
/usr/enterprise-wide-routers/Created-IP-Lists/<which-ever IP list you want to use.txt>

To the following file:
/usr/enterprise-wide-routers/enterprise-wide-routers-IPs.txt

Change into the automation program directory
```
cd /usr/DCDP/bin/python_custom_scripts/enterprise-wide-routers/
```

Input the commands you want to send to the device in the "ssh_custom_enterprise-routers_commands.txt" file.

You enter them just as if you would if were at the CLI, e.g. if you want to grab information, you could enter:
```
Show ip interface brief
Show ip bgp summary
```

As always, leave the command "terminal len 0" at the top of the file. Never remove this command.

Run the Program:
```
./enterprise-wide-routers.py
```

Follow the prompts

You can always refer to the README.txt in the directory:
/usr/enterprise-wide-routers too.

Once the automation run completes, you can view all of your files in the following directory:
/usr/enterprise-wide-routers/configs/

Enter the following command:
```
cd /usr/enterprise-wide-routers/configs/
```

All files will be dated (by the minute, so as long as you wait at least one minute between automation runs, you'll never overwrite the previous files)

Running Enterprise-Wide Automation on Switches

To run automation on an enterprise wide IP list of routers:

Copy IPs from the "Created-IP-Lists " folder located @:
/usr/enterprise-wide-switches/Created-IP-Lists/<which-ever IP list you want to use.txt>

To the following file:
/usr/enterprise-wide-switches/enterprise-wide-switches-IPs.txt

Change into the automation program directory
cd /usr/DCDP/bin/python_custom_scripts/enterprise-wide-switches/

Input the commands you want to send to the device in the "ssh_custom_enterprise-switches_commands.txt" file.

You enter them just as if you would if were at the CLI, e.g. if you want to grab information, you could enter:
Show vlan

As always, leave the command "terminal len 0" at the top of the file. Never remove this command.

Run the Program:
./enterprise-wide-switches.py

Follow the prompts

You can always refer to the README.txt in the directory:
/usr/enterprise-wide-switches too.

Once the automation run completes, you can view all of your files in the following directory:
/usr/enterprise-wide-switches/configs/

Enter the following command:
cd /usr/enterprise-wide-switches/configs/

All files will be dated (by the minute, so as long as you wait at least one minute between automation runs, you'll never overwrite the previous files)

Vendor Neutral Automation

This Chapter covers Vendor Neutral Automation using IP lists you already have (This will also be covered in the next section *BYOI*). This program can be used to program anything you have SSHv2 access to, including:

- Juniper
- ESXi
- F5
- Palo Alto
- Cisco

These IP lists might come from a variety of sources:

- Could be pre-existing IP lists you have for your environment, e.g. for non-Cisco devices

- For Cisco Devices, e.g. a list of MPLS Routers, or Edge-I-NET-Routers

- Could be IP lists you create using NDNA automation and the GRASP toolset. There are many ways you can get creative in building your own IP lists this way. See our blog for *value-add-ready-to-go* procedures on creative ways to build IP lists. *Here's a few example articles from the blog:*

  - Using NDNA automation and GRASP to find Palo Alto hosts in an NDNA Data Center (Build a Palo Alto IP list)
  - Using NDNA automation and GRASP to find Juniper Hosts in an NDNA Data Center (Build a Juniper IP list)
  - Building an Enterprise Wide WAN MPLS Router IP list with NDNA

Once you have your IPs, change into the proper directory:
cd /usr/DCDP/bin/python_custom_scripts/vendor-neutral/

Move the IPs into the proper file.
(Into the *vendor-neutral-IPs.txt*)

Input commands you want to send to the devices.
(Into the *ssh_custom_vendor_neutral_commands.txt*)

Run the program:
./custom-vendor-neutral-python-script.py

That's it!

All your output files will be written to the following location:
/usr/vendor_neutral/configs

BYOI (Bring Your Own IP List) Automation

This Chapter covers BYOI (Bring Your Own IP List) Automation. This is different than Vendor Neutral Automation.

BYOI can be used with the vendor neutral.

It can also be used with enterprise wide automation programs.

It just depends on how you want to use it, or the nature of the IP lists.

For example, does your IP list relate to something that is enterprise wide or specific to a site?

It can also just be a personal preference.

It can be from IP lists your already have, or from IP lists you create using the NDNA program and the GRASP toolset (*using your own creative process*), or using procedures from our blog

A few examples:

Enterprise-Wide-Example

You used a procedure in our blog to create an MPLS WAN routers IP list (Or you already had your own IP list).

You'd just put this IP list into a file (which you'd create yourself) into the following pre-existing folder:
/usr/enterprise-wide-routers/Created-IP-Lists

Name it accordingly:
e.g. if it's for a Company called xyz, name it something like "xyz-MPLS-Routers.txt"

You can now copy this IP list (as needed when you want to run automation on these nodes) into the
/usr/enterprise-wide-routers/enterprise-wide-routers-IPs.txt

Finally, Run automation following the instructions provided in the Enterprise-wide-routers automation section.

Vendor-Neutral-Example

You used a procedure in our blog to build a list of IPs for all Juniper devices that exist in a Data Center. (Or you already had your own IP list) that was specific to a certain site or data-center (so putting the list in enterprise wide really wouldn't make sense, since it's only devices at a single site)

Create a file, say, called <site/dc name>-juniper-devices.txt
Move the file into the following directory
/usr/DCDP/bin/python_custom_scripts/vendor-neutral/

Move the IPs from the above file into the proper file (as needed) when running automation
/usr/DCDP/bin/python_custom_scripts/vendor-neutral/vendor-neutral-IPs.txt

Input commands you want to send to the devices.
(Into the *ssh_custom_vendor_neutral_commands.txt*)

Run the program:
./custom-vendor-neutral-python-script.py

All your output files will be written to the following location:
/usr/vendor_neutral/configs

If these IPs were specific to a Data Center/Site, then you could copy them over to the configs directory of that Data Center too. – e.g. cp /usr/vendor_neutral/configs/* /usr/DataCenters/<your variable-DC-Name>/DCDP/configs

Creative Use of NDNA – Use it Your Way

See our blog for **value-add-ready-to-go** procedures on creative ways to use NDNA.

These should get you started with your own creative ways to use the NDNA program.

NDNA becomes *very* powerful as a troubleshooting and discovery tool when you use a combination of automation, followed by data parsing using the GRASP tool-set. (grep, regex, awk, sed and python)

If you'd like to share information and/or an article you've written on creative ways you've used it, please send us an email (From our support page). We'd love to publish any creative use of the NDNA program on our blog.

VNAD

VNAD is the *Vendor Neutral Automation *n* Discovery Program*. It is provided as a free *value-add* to the main NDNA program, and provided *as-is*. The plan is to make it more full-featured and add features over time, although as it stands, VNAD is still incredibly powerful in what it can do.

VNAD Overview

It's based on ICMP echo requests and SNMP polling for discovery (Using SNMP v2c), and Python and SSH for automation after discovery. After it discovers devices via ICMP and SNMP, you can run automation on those devices using Python and SSH; just like the NDNA program. It is also tied into the NDNA Company/DC-Sites folder structure for automation runs after discovery. (Or into a VNAD LE folder structure.)

*VNAD introduces a new concept called the *VNAD-LE* -- The VNAD logical entity.*

When you run VNAD, it prompts you for:

1. Data Center/Site Name (This is your three-character company code: DC-Name) or VNAD logical entity (*See* VNAD Logical Entities section *in this chapter for more info*)
2. SNMP Community string
3. Subnet to scan in CIDR format, which must be a /23 or smaller, e.g. /23, /24, /25, etc. up to a /32

If you need to scan a /22, you'd just run it twice, scanning two /23s

VNAD creates the following during each discovery scan:

1. IP/Hostname CSV files (Which will contain any node is can both ping using ICMP and can poll using SNMP) – Note: You can potentially see devices in the CSV file that you are able to ping and have SNMP read access to, which are not in any IP lists. This would be because they are not one of the vendors listed below. If it's ping-able, running SNMP and you use a good community string, any device will end up in the CSV file. *This is to be expected.*
2. IP lists *per vendor* . The current supported vendor list is shown below:
   a. Arista
   b. Juniper
   c. Cisco (Broke down by IOS, NXOS, ASA)
   d. Palo Alto
3. Log file (minimalistic in this version, limited to STDERR - exceptions only)
4. These items are saved to the VNAD LE or Data-Center folder structure under a folder named "VNAD"

VNAD has an automation folder, which is like the /usr/DCDP/bin folder for NDNA. This is where all your custom Python Programs are. This is where you run automation against your discovered IPs *per-vendor*/*per-NDNA-site* or *per-vendor*/*per VNAD LE*

You will also “potentially” see IPs in your IP lists or CSV files that are **not** in the /CIDR subnet you specified.

This is due to the way the program extracts hosts and removes duplicate Hosts.

The host you see with an IP that doesn’t match a subnet you scanned, in fact, has IPs in the range you specified, but during the process of removing duplicate hosts, it might choose another IP from the same host.

You will still have all the hosts in the final “IP lists” and “Hostname to IP” CSV spreadsheet.

Important Notes:

1. If you run VNAD on an *NDNA Data-Center*, and you re-run NDNA on the DC (which creates a fresh discovery) it will also remove the VNAD folder and all subdirectories. Not to worry, there is always a backup of your latest VNAD data (for all DC/sites and VNAD LEs) in the usr/Backups folder. Just run a restore of VNAD after re-running an NDNA DC discovery on a site. (See restore chapter.) Or you can just copy the VNAD folder via WinSCP before you run discovery and copy back over to the VM afterwards.
2. *VNAD is dynamically updating*, e.g. when you run a new VNAD scan, it will append any new hosts found to the IP-to-Hostname CSV file, and update IP lists for the VNAD LE or Data-Center/VNAD folders…*so you can incrementally keep updating it*. Duplicate hosts are automatically removed from IP Lists and CSV files.
    a. The dated CSV file in the Data-Center or VNAD LE folder is the one to reference for latest information.
    b. The IP file in the automation folders that ends with *-py.txt* is the file to reference – this will be the final, non-duplicate IP file (*This is also the IP-list file that the automation code looks for too.*)

VNAD Logical Entities

There's no better way to describe what a VNAD LE is than to just *paste* in what you'll see when you run the program:

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

                 _    ___    _____    ____
                | |  / / |/ /   | / __ \
                | | / /  |/ / /| | / / / /
                | |/ / /|  / ___ |/ / /_/ /
                |___/_/ |_/_/  |_/_____/


          Vendor-Neutral-Automation and Discovery program

                 Add-on to the NDNA Program
                         _  _  _  _  _  _  _

                Hit Ctrl Z to Escape the Program
################################################

################################################

Enter The Name of an Existing NDNA Site/DataCenter, VNAD LE (Logical Entity) or new VNAD LE specific
name

VNAD LEs (Logical Entities), would be something that SPANs Multiple Sites/Be logical in Nature.....
 e.g. something like MPLS-WAN-ROUTERS, EDGE-ROUTERS, etc.


                              Names are case sensitive
    This will place all IP Lists that are created in /usr/DataCenters/<your DC Name>
                 or /usr/VNAD-LEs/<your VNAD Logical Entity Name>

         Document the name. You will need to reference it as you use the program!!

###################################################################

ALL YOUR CUSTOM DISCOVERY AND PROGRAMMING FILES WILL ALSO BE STORED IN THESE FOLDERS!!

You still need to run any subsequent custom discovery or programming from the /usr/VNAD/* Directories
(This is where you run the program)

Enter Existing Data_Center, Existing VNAD LE (Logical Entity), or new VNAD LE Here:
```

Running VNAD Discovery

To run VNAD, simply launch the program and follow the prompts. Read and pay attention to the instructions provided in the CLI terminal:

First, make the files executable. (Only required one time):

```
cd /usr/VNAD
```

```
chmod 755 *.bash
```

Run the program:

```
./VNAD.bash
```

Again, follow the prompts. Read and pay attention to the instructions provided in the CLI terminal:

When you run VNAD, it prompts you for:

1. Data Center/Site Name (This is your three-character Company Code:DC-Name) or VNAD logical entity (*See VNAD Logical Entities section in this chapter for more info*)
2. SNMP Community
3. Subnet to scan in CIDR format, which must be a /23 or smaller, e.g. /23, /24, /25, etc. up to /30

Then, it runs the scan. After the scan, you can go into your VNAD LE or DataCenter VNAD directory and view:

1. IP/Hostname files
2. Automation directories
3. IP lists.
4. Log file (for STDERR only)

VNAD Folder Structure and Navigation

This is an important section which discusses *where* (folder location) you use the program at various times of operation.

Task	Folder Location	Purpose	Notes
Run main program ./VNAD.bash	/usr/VNAD/	Discover the Network. Build Network Artifacts	This is the "VNAD" Discovery Program
Review Log file	/usr/DataCenters/<DC name>/ VNAD/logs Or /usr/VNAD-LEs/<VNAD LE Name>/VNAD/logs Or /usr/VNAD/logs/<VNAD-LE-NAME or DC-Name>	Review log for issues to resolve after running the program.	Path depends on if you run it against an NDNA Data-Center or a VNAD LE
Review hostname to IP csv files	/usr/DataCenters/<DC Name>/VNAD/hostname-to-IPs /usr/VNAD-LEs/<VNAD LE Name>/VNAD/hostname-to-IPs	Self-explanatory	Review the dated CSV file
Review hostname to IP lists	/usr/DataCenters/<DC name>/VNAD/IP-Lists /usr/VNAD-LEs/<VNAD LE-Name>/VNAD/hostname-to-IPs /IP-Lists	Self-explanatory	Review the IP file that ends with -py.txt
Run VNAD automation on discovered devices	/usr/VNAD/Automation	Self-explanatory	All supported vendors are located in separate folders
View configs after running automation	/usr/DataCenters/<DC name/VNAD/configs or /usr/VNAD-LEs/<VNAD LE Name>/VNAD/configs	Self-explanatory	

VNAD Automation

This Chapter covers the custom automation features of the VNAD program. It's almost identical to the NDNA automation programs.

All VNAD automation programs are run from subfolders of:
/usr/VNAD/Automation/   with different folders for each vendor supported.*

Design & Architecture information

1. *Uses the Python Threading Module*
2. *Supports a maximum of 300 threads at a time. So, if you have more than 300 IPs in an IP list then, you must edit your IP file to run 300 IPs at a time.*
3. *While the program runs, it throttles threads from going out into the network to 25 at a time, with a 20 second interval before running the next 25 threads. This helps to assist with TACACS being overloaded*

In addition to threading, all output is written to separate text files for each node, with the name of the IP address in the name of the file. This assists with creative use of the program. Files are written for each node, regardless if you are sending show commands (e.g. pulling information from the network) or programming the network.

VNAD Automation uses the IP Lists created during the VNAD program network discovery.

All "Vendor-Specific" automation programs will prompt you for the Data-Center you want to run the automation on or the VNAD LE you want to run the automation on.

It will then run the automation on the IP lists from that Data-Center (Or VNAD LE) – Just like the NDNA program does.

Log files for both standard output and exceptions are written into the following location:
/usr/VNAD/logs/<DC-name or VNAD-LE name>

The only thing you need to change and/or configure before you run the program is:
Input the commands you want to send to the devices via the "ssh_<vendor-name>_commands.txt" file.

Config files are written to the configs directory as stated in previous section.

/usr/DataCenters/<DC name>/VNAD/configs

or

usr/VNAD-LEs/VNAD LE name>/VNAD/configs)

Backup

This Chapter covers the backups which are performed for the various programs, including:

1. NDNA
2. VNAD
3. VNAD/NDNA custom configs directories (From automation runs on DCs/Sites/VNAD LEs)

All backups are stored in the following directory:
*/usr/Backups/**

When the main NDNA program (or enable pass version of the NDNA program) runs, it creates a backup of the DC/Site to a zip file, and moves it into the backups directory. <u>This can be used with the Restore program to restore your data from a previous discovery (If you rediscover a DC and overwrite your data and/or if you accidentally delete your data)</u>

When the VNAD program runs, it creates a backup of the DC/Site or VNAD LE to a zip file, and moves it into the backups directory. <u>This can be used with the Restore program to restore your data from a previous discovery (If you accidentally delete your data)</u>

When the VNAD program runs or the NDNA program (or enablepass version) runs, if any custom automation has been run on the DC/Site or VNAD LE, and any config files exist in the configs directory, it creates a backup of the DC/Site or VNAD LE configs directory to a zip file, and moves it into the backups directory. <u>This can be used with the Restore program to restore your custom configurations, (If you rediscover a DC or VNAD LE and overwrite your data and/or if you accidentally delete your data)</u>

Backup file name formats:
<u>NDNA backups will have the following name format:</u>
<variable name>-DCDP-Backup-<date>.zip

<u>NDNA enable pass version backups will have the following name format:</u>
<variable name>-DCDP-enablepass-Backup-<date>.zip

<u>NDNA custom configs directory backups will have the following name format:</u>
<variable name>-custom-configs-dir-<date>.zip

<u>VNAD backups will have the following name format:</u>
<variable name>-VNAD-Backup-<date>.zip

<u>VNAD custom configs directory backups will have the following name format:</u>
<variable name>-VNAD-custom-configs-dir-<date>.zip

The only thing you will need to manually backup is the enterprise wide automation and vendor neutral automation output files. Be sure to back these up regularly as needed down to your Windows or Mac host machine.

Also, be sure to back-up your backups often, e.g. manually copy all zip files down to your Windows or Mac host machine

Restore

This Chapter covers the Restore programs which can be used to automatically restore backups for the following:

1. NDNA (Data-Centers)
2. VNAD (Data-Centers or VNAD LEs)
3. VNAD/NDNA custom configs directories (Automation Run configs)

Detailed steps to restore any of the above:

1. First, make sure either one of the following conditions are met:
    a. The Data-Center folder structure is completely deleted for the DC/Site under /usr/DataCenters, ***or***
    b. You can create/or leave the DataCenter folder, but delete everything below it, e.g. you can have /usr/DataCenters/<your DC name> but delete the DCDP directory and everything below it (if this directory structure exists)
2. Copy the .zip file from the /usr/Backups directory of the backup you want to restore to the /usr/Restore directory using the cp command, e.g. cp /usr/Backups/<filename>.zip /usr/Restore
3. ***Do not unzip the file. The Restore program will do it for you***. If you unzip the file manually, the program will prompt you to overwrite it again. Just choose *A* for all.
4. Change into the Restore directory, e.g. cd /usr/Restore
5. Run the restore program, either for NDNA (Restore-DataCenter.sh) or for VNAD (Restore-VNAD.sh)
6. Follow the prompts. It'll ask you for the Data-Center name and/or VNAD LE (If running a VNAD restore)
7. Input Your DC or VNAD LE name, and that's it.
8. To restore custom configs, just unzip it directly in the usr/Backups directory, e.g.
    a. cd /usr/Backups
    b. unzip <filename>.zip
    c. You'll have a new directory structure <u>under</u> /usr/Backups called /usr/DataCenters/<your-site-name>/DCDP/configs
    d. Just copy your configs from here over to the real Data-Center path, e.g. cp /usr/Backups/usr/DataCenters/<your-site-name>/DCDP/configs/*.txt /usr/DataCenters/<your-site-name>/DCDP/configs
    e. You can now delete this directory from the Backups directory. We recommend doing this from the WinSCP interface to make sure you are in the /usr/Backups directory. Do not delete the path /usr, e.g. DO NOT run the command rm -r /usr – make sure you are in the backups directory from the WinSCP interface and delete the usr directory that is inside of the /usr/Backups ONLY, or you will destroy your system. You would then have to redeploy it from the OVA!

Links

Automate-the-network Blog
https://www.automate-the-network.com/blog

Automate-the-network
https://www.automate-the-network.com

OUI Lookup
https://www.wireshark.org/tools/oui-lookup.html

Subnet Calculator
http://www.subnet-calculator.com/subnet.php?net_class=A

IPv6 subnet calculator
https://subnettingpractice.com/ipv6_subnetting.html

Diff Checker
https://www.diffchecker.com/diff

TOS Mappings
https://www.tucny.com/Home/dscp-tos

Cacti
https://cacti.net/

Glossary Of Terms

NDNA – Network Discovery "*N*" Automation Program

DCDP - Data Center Discovery Program (NDNA Legacy program term)

VNAD – Vendor Neutral Automation and Discovery Program

VNAD LE – VNAD Logical Entity – used for creating folder structures for devices spanning multiple Data-Centers and/or Multiple Sites. *Examples:* Juniper-SRX-FWs, MPLS-WAN-ROUTERS, Palo-Alto-Firewalls, etc.

Seed Device – IP Address. Usually associated with the core of a Data Center to run the NDNA program.

NTC – NDNA-*to*-Cacti Integration

TACACS – Terminal Access Controller Access Control System.

VM – Virtual Machine

CDP – Cisco Discovery Protocol

IOS – Cisco Internetwork Operating System

NXOS – Cisco Nexus Operating System

JUNOS – Juniper Operating System

L2 - Layer 2 in the OSI Model, associated with Switches

L3 - Layer 3 in the OSI Model, associated with Routers

Site-Specific – Associated with a Data-Center/Site in relation to automation features

Enterprise-Wide – Associated with all Data-Centers/Sites relating to automation features

Vendor Neutral – Not associated with any specific vendor.

GRASP - – An acronym for Grep, Regular Expressions, AWK, SED, and Python