Python Scripting for Dell Networking N-Series

Dell Networking Solutions Engineering
March 2016

A Dell EMC Deployment and Configuration Guide

Revisions

Date	Description	Authors
March 2016	Initial Release - Version 1.0.1	Victor Teeter Ravindra Kadiyala

Contents

Executive Summary

There are several ways to configure a Dell N-Series switch, each with its own advantages. While the Web User Interface allows new users to see all features displayed in logical tiers and can prompt them for settings, more experienced users often prefer the Command Line Interface (CLI) for entering commands to configure switches. When several switches require configuration changes or one or more switches require frequent changes, experienced users can now take advantage of the Dell Networking N-Series Python scripting interface.

Python is a popular high-level programming language with a vast standard library and supports multiple programming styles. Interpreters are available for many operating systems, allowing code execution on a wide range of systems. Dell Networking N-Series switches with firmware 6.3.x.x and later support installation and execution of Python applications to assist in the automation of configuring these switches. This provides a better alternative to the existing script feature in that it allows more control and therefore more robust scripts.

This paper provides instructions and examples on how to deploy Python scripts on Dell Networking N-Series switches.

1 Introduction

This document is a supplement to the *Dell Networking N-Series User Guide* and provides easy step-by-step instructions to help users configure Dell N-Series switches using Python scripts. This document primarily advises the reader on how to run Python scripts on N-Series switches. Explanation of Python syntax is beyond the scope of this document. Locate Python syntax explanations in numerous other resources both on the internet and in hardcopy material. Switch administrators need to develop and test scripts offline prior to executing a Python script on the switch since the switch does not offer interactive shell access for script development. Dell Networking N-Series supports Python version 2.7.10.

Examples in this document use Dell Networking N3xxx switches, however any N-Series switches running firmware version 6.3 or later accepts scripts and commands shown in this paper. The following N-Series models offer Python (v. 2.7.10) scripting support:

N1524, N1524P, N1548, N1548P, N2024, N2024P, N2048, N2048P, N3024, N3024F, N3024P, N3048, N3048P, N4032, N4032F, N4064, N4064F

Important Notes: Dell EMC strongly advises switch administrators to maintain Dell Networking N-Series switches with the latest version of the Dell Networking Operating System. Dell Networking continually improves the features and functions based on feedback from you, the customer.

For critical infrastructure, Dell EMC recommends prestaging new releases into a non-critical portion of the network to verify configurations and operations prior to installing into a production environment.

Figure 1 illustrates the network topology represented throughout this guide.

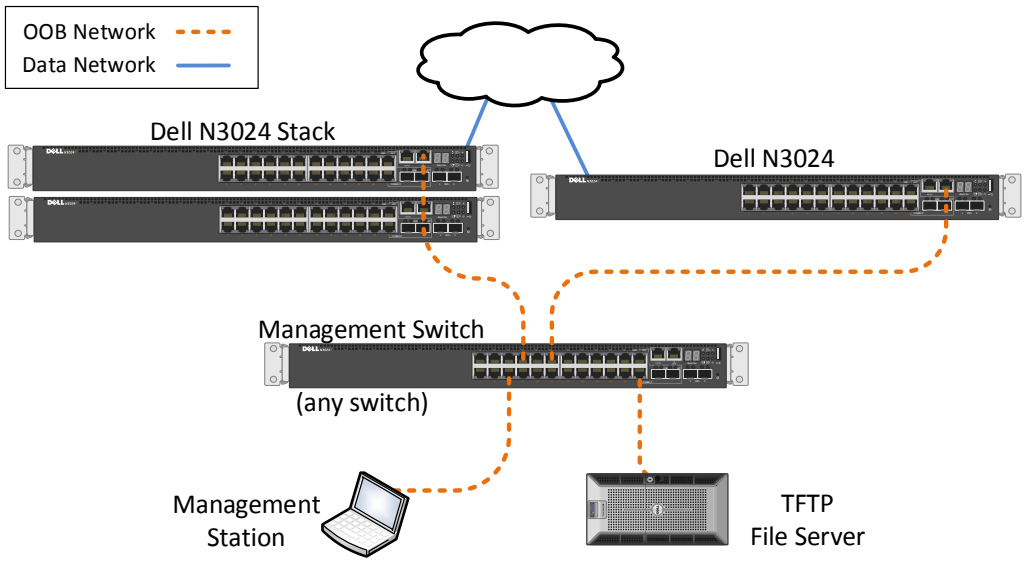

Figure 1 Python scripting example topology

The topology includes two networks. The Out-of-Band (OOB) network is used only for management of switches on the network, while the Data network handles production traffic such as applications and file sharing. The examples in this guide apply to both a single N-Series switch as well as multiple, stacked N-Series switches as shown in Figure 1. A console cable may also be used between the management station and each switch to view the output of Python scripts.

1.1 Console output

Python scripting output is displayed from the console. Administrators can use the console cable that comes with the Dell N-Series switch to view console output, as well as configure the switch using any of the commands in this guide. See the User Guide for your switch for more information on how to use the console port.

Note: See Appendix B for more information about console output.

1.2 OOB Network

Administrators can also use the OOB or management network (for example, VLAN 1) for switch configuration. This allows the administrator to SSH or telnet into each switch from a single management station. All commands in this guide may be entered through the OOB/management network.

1.3 SSH/Telnet

Telnet access must be allowed (default) on the switch since Python scripts use the telnetlib module to telnet internally into the switch console (localhost). From there, the scripts initiate CLI commands for both managing and configuring Dell N-Series switches. Where required, the example scripts in this document provide the basic framework for a local, internal Telnet session to the switch console.

Consequently, three distinct uses of Telnet/SSH may be used when deploying and running Python scripts:

- The telnetlib module mentioned above is used for internal commands.
- SSH and Telnet can be used to remotely access each switch from a management station.
- As with any CLI command, telnet may be embedded into a Python script for automating a process that requires remote access to another switch.

For those new to Python scripting, it is important to differentiate these three uses, since it is easy to confuse them when reading this guide.

1.4 Python script flow on Dell N-Series

Figure 2 depicts common flows of Python scripts on Dell N-Series switches.

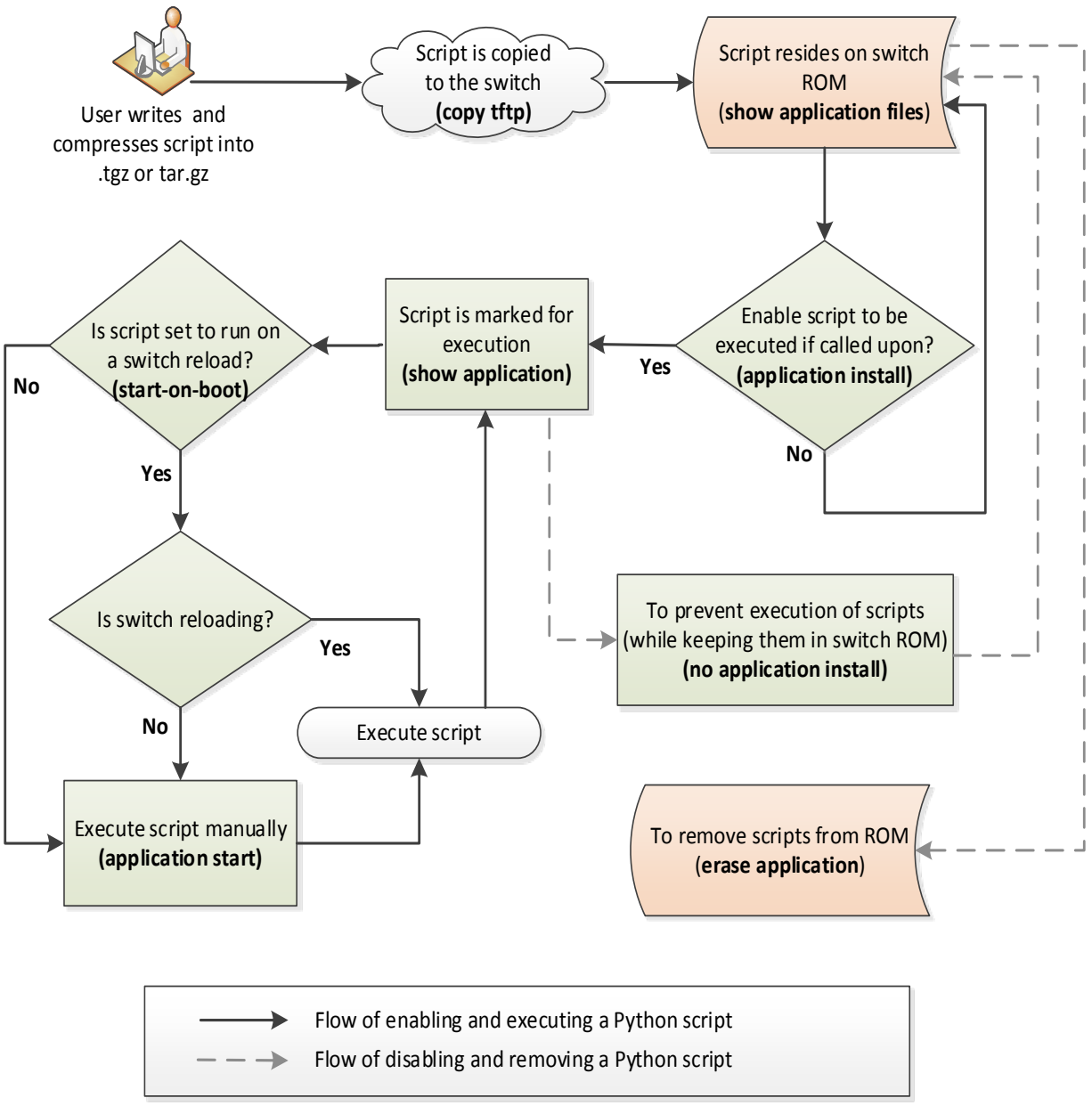

Figure 2 Common flows of Python Scripts on Dell Networking N-Series

The flowchart above illustrates a common example of creating, enabling, executing, disabling and removing a Python script on a Dell N-Series switch. There are a few other methods to start and stop these scripts as well.

The list below includes the techniques represented in the flow chart along with a few others. The commands mentioned are covered in detail in this guide.

Starting and Stopping a Python script

Methods of starting Python scripts include the following:

- Using the **application start** command
- Using the **start-on-boot** parameter and reloading the switch
- Using the **auto-restart** parameter (restarts/loops the script)
- Call the script from another script (Python or other)

Methods of stopping Python scripts include the following:

- Script completes (not in a loop)
- Using the **application stop** command
- Using the **no application install** command
- Using the **erase application** command
- A switch reload (and script is not set to **start-on-boot**)
- Script is halted by another script using an above command

1.5 Tarball (.tgz and tar.gz)

Before copying a Python script to a switch (as shown at the top of Figure 2), the administrator must set the appropriate permissions, then compress the file into a tarball:

1. Set the execute permission on the script file.
2. Compress the script into a zipped tar archive file (tarball), with either a .tgz or .tar.gz extension.

The second step is generally accomplished from a Linux command line using the command:

tar cfz <*script filename*>**.tgz** <*script filename.py*>

Other methods may also be used, such as right-clicking the file from the GUI desktop and selecting **Compress…**. The N-Series switch will automatically decompress the file after copying, placing the executable script file into the user-apps directory.

2 Switch commands for utilizing Python scripts

This section covers all Dell N-Series commands for installing and managing Python scripts on an N-series switch: *copy, dir, application install, show application, application start, application stop* and *erase.* The commands are presented here in the order typically used. Descriptions, syntax and examples for each are described in detail below.

2.1 copy

Use the **copy** command to copy Python scripts from the source to the switch. Scripts must first be compressed into a *.tgz* or *.tar.gz* file as discussed on page 8. Using the *application* option, the compressed files are automatically decompressed and placed into the *user-apps* directory.

Syntax: **copy** *source-url/filename* **application** *filename*

Example:
```
console#copy tftp://172.100.1.50/hello.tar.gz  application  hello.tar.gz
                                    ⋮
Application download completed successfully.
```

If the script must automatically run after a switch reload, use the **application install** command (below) to install a script. Then use the **copy** command again to save the *running-configuration* to the *startup-configuration.* This is especially necessary for the **start-on-boot** parameter to succeed.

Note: There is a limit of 16 characters for script filenames, including the extension. This limitation also applies to the compressed (**.tgz** or **.tar.gz**) file containing the script.

2.2 dir

Use the **dir** command to show all Python scripts that were copied to the *user-apps* directory on the switch. Scripts in this directory can be installed as user applications. See the **copy** command for directions on how to copy other scripts into this directory.

Syntax: **dir user-apps**

Example:
```
console#dir user-apps

Attr      Size(bytes)     Creation Time              Name
drwx             1088  Dec 30 2015 13:54:31  .
drwx             2824  Dec 30 2015 10:25:35  ..
-rwx               50  Dec 29 2015 21:18:18  hello
                                    ⋮
Total Size: 215265280
Bytes Used: 778487
Bytes Free: 214486793
```

Note: The **show application files** command presents the same output without attributes and time stamp

2.3 application install

Use the **application install** command (in global configuration mode) to load a script into switch memory. This does not start the script, but enables it to be started using one of the methods listed in section 1.4.

Note: Scripts must be in the *user-apps* directory before they can be installed. See the copy command above for more information.

After installing, scripts can be run manually using the application start command or can be scheduled to run automatically at bootup using the **start-at-boot** parameter. Other parameters listed below are also available to assist the administrator with CPU and memory management while scripts are running.

Syntax: **application install** <*filename*> **[start-on-boot] [auto-restart] [cpu-sharing** <*percent*>**]** **[max-memory** <*max-megabytes*>**]**

Syntax: **no application install** <*filename*>

Example: console(config)#**application install hello**

Optional parameters:

start-on-boot	automatically starts script at bootup
auto-restart	runs script in a loop once started manually or at bootup
cpu-sharing <*percent*>	maximum CPU (0%-99%) the app may use
max-megabytes <megabytes>	limits the amount of memory (0-200) the app may use

Be sure to save the *running-configuration* to the *startup-configuration* after using the **application install** command if the script needs to run after a switch reload. This is particularly essential for the **start-on-boot** parameter to succeed. Use the **no application install** command to uninstall the script.

Note: The **no application install** command uninstalls the script and stops execution if started. If the desire is only to stop the current execution of the script but leave it installed to run later, run the application stop command discussed on page 11 instead.

The **no application install** command does not remove the script from the switch. To remove a script completely, use the erase command discussed below.

2.4 show application

Use this command to see the scripts that are loaded into memory. Each script listed in the **show applications** command also has an entry in the *running-configuration*.

Syntax: **show application [files]**

Example: console#**show application**

OpEN application table contains three entries.

Name	StartOnBoot	AutoRestart	CPU Sharing	Max Memory
SupportAssist	Yes	Yes	0	0
hello	No	No	0	0
hiveagent	Yes	Yes	0	0

Optional parameter:

files shows all application process directory contents available for install

2.5 application start

Use this command to execute a Python script. The script must be installed into memory using the **application install** command (explained above) before it can be started.

Syntax: **application start <***application-name***>**

Example: console#**application start hello**

Application started.

console#
Hello World!

2.6 application stop

Use this command to stop a running Python script. This command does not remove the script from memory. The script remains ready for manual execution at any time or automatic execution at bootup if configured.

Syntax: **application stop <***application-name***>**

Example: console#**application stop hello**

Application stopped.

Hint: Output of a script goes only to the switch console screen. It may be easier to SSH/Telnet into the switch to stop a running script if the console is continuously scrolling.

2.7 erase

Use the **erase** command to remove a Python script from the switch. Use the **dir user-apps** command discussed above to verify that the script has been removed from the user-apps directory.

Syntax: **erase application** *filename*

Example: console#**erase application hello**

Application file erased.

3 Examples of Python scripting on Dell N-Series

Three examples are provided below to demonstrate various ways a Python script can be used to configure and manage Dell N-Series switches. Example 1: Quick Steps is a short illustration showing the fundamental commands to express how quickly scripts can be created and applied. Example 2: Applying Python scripts for execution at reload and 3.3 provide more intricate details to build a better understanding of these features while covering the remaining commands.

3.1 Example 1: Quick Steps

The following is a list of steps to create and execute a simple Python script on an N-Series switch:

a. Use your favorite script editor (for example, gedit) to create the following script named "hello"

```
#!/usr/bin/env python
print "\n Hello World! \n"
```

b. Set execute permissions on the script file and compress the file into a .tgz or .tar.gz format (as discussed on page 8)

c. From the switch console screen, use the commands below (in bold) to:

   i. Copy the file to "user-apps" (be sure to use the "application" option)
   ii. Verify the file appears in the user-apps directory
   iii. Install the application
   iv. Start the application

console#**copy tftp://xx.xx.xx.xx/hello.tar.gz application hello.tar.gz** *< copies to user-apps*
console#**dir user-apps** *< shows file was copied*
console#**configure**
console(config)#**application install hello** *< readies file for execution*
console(config)#**show application** *< shows file is ready*
console(config)#**exit**
console#**application start hello** *< executes the script*

Application started.

console#

Hello World!

Note: Refer to the more elaborate examples below for details on what each of these commands do.

Note: Along with Python scripting, a second scripting method available for use with the Dell N-Series is to use the *copy tftp <…> script* command to load a .SCR script file. This offers a simple static script technique that may be accommodating in certain situations, but does not offer the robustness and control of Python scripting such as parsing, looping, if-then-else, etc. For more information on .SCR scripting, see the Images and File Management chapter of the User Guide.

3.2 Example 2: Applying Python scripts for execution at reload

This example shows how to configure a Python script to execute each time an N-Series switch reloads. This particular script copies the startup configuration to a TFTP server upon each reboot of the switch.

For this script to work the switch must have access to a TFTP server and the switch's configuration must include a username/password and OOB IP address.

This example demonstrates the following three areas:

i. Applying a Python script to be run upon reloading of the switch
ii. Performing management tasks, such as backing up the switch configuration or updating firmware using Python scripts
iii. Provides the framework for applying any CLI command in a Python script

Save the commands below into a file using a script editor, then compress the file into a **.tgz** or **.tar.gz** format (as discussed on page 8).

Note: There is a limit of 16 characters for script filenames, including the extension. This limitation also applies to the compressed (**.tgz** or **.tar.gz**) file containing the script.

```
#!/usr/bin/env python
#load this script using the start-on-boot parameter in
#order to back up the switch each time the switch reloads

import telnetlib
import os
import re
import time
import string
import sys

HOST = '127.0.0.1'
PORT = 23
LOGIN_STRING = "Login:"
PASSWORD_STRING = "Password:"
TERMINAL_LEN_ZERO = "terminal length 0\n"
TERMINAL_MONITOR = "terminal monitor\n"
ENABLE_STRING = "enable\n"
CONFIG_STRING = "configure\n"

USERNAME = 'admin'
PASSWORD = 'password'
ENABLE_PASSWORD = ''
TIMEOUT = 3

def do_terminal_settings(tn):
```

```
    tn.write(TERMINAL_MONITOR)
    tn.read_until("#")
    tn.write(TERMINAL_LEN_ZERO)
    tn.read_until("#")

def do_login(tn):
    tn.read_until(LOGIN_STRING, TIMEOUT)
    tn.write(USERNAME + "\n")
    tn.read_until(PASSWORD_STRING, TIMEOUT)
    tn.write(PASSWORD + "\n")
    tn.read_until(">", TIMEOUT)
    tn.write(ENABLE_STRING)
    tn.read_until("#", TIMEOUT)

#Replace the “xx.xx.xx.xx” below with the TFTP server IP address
def do_config(tn):
    tn.read_until("#", TIMEOUT)
    tn.write("copy running-config tftp://xx.xx.xx.xx/running-configuration\n");
    tn.read_until("(y/n)", TIMEOUT)
    tn.write("y");
    tn.read_until("#", TIMEOUT)

def main():
    telnet =  telnetlib.Telnet(HOST,PORT)
    do_login(telnet)
    do_terminal_settings(telnet)
    do_config(telnet)
    telnet.close()
    sys.exit(0)

main()
```

Switch commands

Enter the following commands from the switch command line, replacing xx.xx.xx.xx with the TFTP server’s IP address. Replace <filename1> with the name of the compressed file. Replace <filename2> with the name of the decompressed file (which automatically unzips during the copy):

console#**copy tftp://xx.xx.xx.xx/<filename1> application <filename1>**
console#**configure**
console(config)#**application install <filename2> start-on-boot**
console(config)#**exit**
console#**write**

The last command, **write**, copies the running configuration to the startup configuration. The running configuration must be saved to the startup configuration for the **start-on-boot** feature to work. The script will run the next time the switch boots and every time thereafter. There are several ways a script can terminate, most of which are listed in Section 1.4, Python script flow on Dell N-Series.

Though the script was installed using the **start-on-boot** parameter, administrators can start it at any time using the application start command as long as the script allows for it (e.g. command prompts are properly detected). To turn off the **start-on-boot** feature while keeping the script installed to be run manually, enter the following command to overwrite the existing install:

console(config)#**application install <filename2>**

Framework example

The example script above provides the framework for any set of CLI commands. For instance, administrators can use the script to update firmware on a switch or switch stack simply by modifying the **do_config** subroutine with the necessary commands. To update the firmware upon script execution, simply replace **do_config** with the following:

```
def do_config(tn):
    tn.read_until("#", TIMEOUT)
    tn.write("copy tftp://xx.xx.xx.xx/N3000_N2000vNEW.stk backup\n")
    tn.read_until("(y/n)", TIMEOUT)
    tn.write("y")
    tn.read_until("#", TIMEOUT)
    tn.write("boot system backup\n")
    tn.read_until("#", TIMEOUT)
    tn.write("reload\n")
    tn.read_until("(y/n)", TIMEOUT)
    tn.write("y")
```

3.3 Example 3: Running Python scripts on an active switch

This example shows how to configure a Python script to be executed by the administrator at any time without reloading the switch (unless the script itself is programmed to execute a reload). This particular script performs the configuration tasks of creating and then verifying the creation of 1000 VLANs.

This example demonstrates the following areas:

i. Running a script on an active switch in production (without reloading)
ii. Using logic in scripts (e.g. loops, parsing, if-else)
iii. Modifying switch configurations (e.g. creating VLANs) using Python scripts

Save the commands below into a file using a script editor, then compress into a **.tgz** or **.tar.gz** format (as discussed on page 8).

Note: There is a limit of 16 characters for a script filename, including the extension.

```
#!/usr/bin/env python
#script creates 1000 VLANS (2-1002), then verifies 1000 were created
import sys, telnetlib, re

#variable declaration
#Replace the “xx.xx.xx.xx” below with the local host’s IP address
hostname = 'xx.xx.xx.xx'
username = 'admin'
password = 'password'
enPrompt = '>'
confPrompt = '#'

#open a telnet session to the device
print ("Opening telnet session to the device")
t = telnetlib.Telnet(hostname)
expect = t.read_until
send = t.write
expect('User:')
send(username + '\r')
expect('Password:')
send(password + '\r')
expect(enPrompt)
send('enable\r')
expect(confPrompt)
send('terminal length 0\r')
expect(confPrompt)
send('configure terminal\r')
expect(confPrompt)

#loop to create vlans
```

```
print ("Creating 1000 VLANs on the switch")
for x in range(2, 1002):
    cmd = "vlan %d" %(x)
    send (cmd + '\r')
    expect(confPrompt)
    send ('exit\r')
    expect(confPrompt)

#parse to count vlans created
print ("Verifying the VLAN database")
send('show vlan\r')
vlan = expect(confPrompt)
totalVlan = re.findall(r"VLAN(\d+){3}", vlan)
exp = len(range(2, 1002))

#if-else to determine output
if len(totalVlan) == exp:
    print exp,("VLANs are created on the switch")
else:
    print ("Failed to create required number of VLANs on the switch")

#close the telnet session
print ("Closing the telnet session")
t.close()
```

Switch commands

Enter the following commands from the switch command line. Replace xx.xx.xx.xx with the TFTP server's IP address. Replace <filename1> with the name of the compressed file. Replace <filename2> with the name of the decompressed file (which automatically unzips during the copy):

console#**copy tftp://xx.xx.xx.xx/<filename1> application <filename1>**
console#**configure**
console(config)#**application install <filename2>**
console(config)#**exit**

These commands install the script and mark it for execution. Administrators may then enter the *start* command below when ready to run the script. The resulting output is shown:

```
console#application start <filename>

Opening telnet session to the device
Creating 1000 VLANs on the switch
Verifying the VLAN database
VLANs are created on the switch
Closing the telnet session
```

Run the appropriate show commands to verify the results.

```
console#show vlan

VLAN   Name                 Ports          Type
-----  ---------------      -----------    ------------
1      default              Po1-128,       Default
                            Gi1/0/1-48,
                            Te1/0/1-2
2      VLAN0002                            Static
3      VLAN0003                            Static
4      VLAN0004                            Static
                    ⋮
1001   VLAN1001                            Static
1002   VLAN1002                            Static
```

3.4 Testing offline (Best Practice)

Administrators can test Python scripts without having to compress and then copy the compressed Python script to the switch each time between edits. The recommended method is to use a Linux system to run the Python scripts, while establishing a telnet connection to the Dell N-Series switch under test. For instance, Dell EMC tested the script shown in 3.3Examples of Python scripting on Dell N-Series several times from a remote Ubuntu system until the script was fully validated and working. Doing this prior to performing the tftp copy onto the switch can save valuable time, drastically cutting down on compression (tar.gz and tgz) and copying (tftp) tasks between debugging sessions.

Dell EMC validated the Example 3 script using Ubuntu v. 14.04, with Python v. 2.7.6 installed (default). As written, this script requires no changes since the `"hostname = 'xx.xx.xx.xx'"` line uses the actual OOB IP address of the switch. The loopback IP address would also work but only after being copied to the switch. If using the loopback address in the script, it will need to be temporarily changed to the OOB/ management IP address for remote testing in order to access the switch over the network.

From the Ubuntu terminal command line, go to the folder where the script resides, or set the path accordingly, in order to execute the script. Run the script by typing **python <script.py>** and **Enter**. The script will create 1000 VLANs on the switch just as if running it directly from the switch.

4 Troubleshooting

This section provides tips on how to help alleviate problems that may be encountered when working with Python scripts.

Problem	Possible Solution(s)
error “ImportError: no module” is received when importing a module into Python	The module you are trying to use is not available. Double-check the spelling of the module. It is also possible that the module is valid for Python but not supported in the Dell N-Series switch. For a complete list of supported modules for N-Series, see Appendix A: Supported Python modules.
error “execl: Exec format error” when running the “application start” command	Make sure both the compressed file (.tar.gz or .tgz) as well as the uncompressed file permissions are set to “execute” before copying the file onto the switch. Make sure the script is created and saved with a scripting editor that uses plain text character encoding (UTF-8 or ASCII). This ensures that no special (hidden) characters are embedded within the file preventing it from executing properly. Common text editors such as those used for word processing typically add special characters that restrict scripts from properly executing.
error when attempting to start script using the “application start” command	Check to make sure the first line in the script is #!/user/bin/env python.
No output on the screen	If after starting the application script, expected output does not display in the CLI session, be sure to view the serial console for output and not an SSH or Telnet session. Python scripts only send output to the serial console.
script files are missing from the “user-apps” directory or when typing “show application files,” even though the TFTP copy was successful	Check to see if the script filename has exceeded 16 characters (including extensions). There is a limitation of 16 characters for Python script filenames. The compressed file (with a .tgz or .tar.gz extension) also has this limitation.

"tar: no gzip magic" message is seen while copying the compressed file to the switch or "Unable to unpack archive…" message is seen while copying the compressed file to the switch	The method used to compress the file is not acceptable, and the switch is not able to unpack it. Be sure to compress the script into a gzipped tar archive, with either a .tgz or .tar.gz extension as discussed on page 8 of this guide. To verify whether or not the file was properly decompressed after copying to the switch, run the "show application files" command. Any script files that still show a compress extension (e.g. .tgz or .tar.gz) were not properly compressed and therefore could not be decompressed. Be sure to use the instructions on page 8 of this guide to properly compress the file. Note: As discussed in the copy command section, the switch unpacks the file during copying and strips any "compress" extensions off of the script file.
Script is not running properly using the "start-on-boot" parameter. For example, the script is not able to contact the TFTP/FTP server or Telnet outside of itself in the first few seconds after a reload, when using "start-on-boot" to initiate the script.	The script may be starting too soon, not allowing for the network to be fully discovered by the switch. After a switch reload, ports need time to initiate and connect to the network. Protocols like spanning tree, DHCP, and LLDP require a short period of time to discover the network. Stacked switches can take addition time in completing a discovery. Scripts should be written to allow time for the switch ports to be fully functional. This can be done in multiple ways. Below are two examples. 1. In the beginning of the script, include a "ping" loop to the desired remote server (e.g. TFTP server) and exit the loop once the destination is reachable. This is the recommended method since it allows the script to initiate and complete in the most efficient manner. It also requires the least amount of debugging and maintenance. 2. In the beginning of the script, import the *time* Python module and use time.sleep() to hard-code a delay. This is less efficient than method 1 above, since it is difficult to determine how long of a delay is needed. Selecting a duration that is too long adds unnecessary delays in completing the script. Any duration selected today may need to be changed at a later date as other influences change on the network, switch or switch stack.

A Supported Python modules

Table 1 shows the complete list of Python version 2.7 modules included with Dell N-Series switches. Each module is defined in the Python Standard Library.

Table 1 Python modules supported with Dell N-Series switches

BaseHTTPServer	audioop	io	rfc822
Bastion	base64	itertools	rlcompleter
CGIHTTPServer	bdb	json	robotparser
ConfigParser	binascii	keyword	runpy
Cookie	binhex	libopenclt	sched
DocXMLRPCServer	bisect	libospf	select
HTMLParser	bsddb	libpam	sets
MimeWriter	cPickle	libping	sgmllib
OpEN	cProfile	libproc_libs	sha
OpENUtil	cStringIO	librpcclt	shelve
OpEN_py	calendar	libsock_agent	shlex
Queue	cgi	libsshcompat	shutil
SimpleHTTPServer	cgitb	libsshpam	signal
SimpleXMLRPCServer	chunk	libtraceroute	site
SocketServer	cmath	libvr_agent	smtpd
StringIO	cmd	libvrf_init	smtplib
UserDict	code	libz	sndhdr
UserList	codecs	linecache	socket
UserString	codeop	locale	spwd
_LWPCookieJar	collections	logging	sre
_MozillaCookieJar	colorsys	macpath	sre_compile
_OpEN	commands	macurl2path	sre_constants
__builtin__	compileall	mailbox	sre_parse
__future__	compiler	mailcap	ssl
_abcoll	contextlib	markupbase	stat
_ast	cookielib	marshal	statvfs
_bisect	copy	math	string
_codecs	copy_reg	md5	stringold
_codecs_cn	crypt	mhlib	stringprep
_codecs_hk	csv	mimetools	strop
_codecs_iso2022	curses	mimetypes	struct
_codecs_jp	datetime	mimify	subprocess
_codecs_kr	dbhash	mmap	sunau
_codecs_tw	decimal	modulefinder	sunaudio
_collections	difflib	multifile	symbol
_csv	dircache	multiprocessing	symtable
_ctypes	dis	mutex	sys
_ctypes_test	distutils	netrc	sysconfig

_elementtree
_functools
_heapq
_hotshot
_io
_json
_locale
_lsprof
_md5
_multibytecodec
_multiprocessing
_osx_support
_pyio
_random
_sha
_sha256
_sha512
_socket
_sre
_ssl
_strptime
_struct
_symtable
_sysconfigdata
_testcapi
_threading_local
_warnings
_weakref
_weakrefset
abc
aifc
antigravity
anydbm
argparse
array
ast
asynchat
asyncore
atexit
audiodev
doctest
dumbdbm
dummy_thread
dummy_threading
email
encodings
errno
exceptions
fcntl
filecmp
fileinput
fnmatch
formatter
fpformat
fractions
ftplib
functools
future_builtins
gc
genericpath
getopt
getpass
gettext
glob
grp
gzip
hashlib
heapq
hmac
hotshot
htmlentitydefs
htmllib
httplib
ihooks
imaplib
imghdr
imp
importlib
imputil
inspect
new
nntplib
ntpath
nturl2path
numbers
opcode
operator
optparse
os
os2emxpath
parser
pdb
pickle
pickletools
pipes
pkgutil
platform
plistlib
popen2
poplib
posix
posixfile
posixpath
pprint
profile
pstats
pty
pwd
py_compile
pyclbr
pydoc
pydoc_data
pyexpat
quopri
random
re
repr
requests
resource
rexec
syslog
tabnanny
tarfile
telnetlib
tempfile
termios
textwrap
this
thread
threading
time
timeit
toaiff
token
tokenize
trace
traceback
tty
types
unicodedata
urllib
urllib2
urlparse
user
uu
uuid
warnings
wave
weakref
webbrowser
whichdb
wsgiref
xdrlib
xml
xmllib
xmlrpclib
xxsubtype
zipfile
zipimport
zlib

B Console output options

Python scripting output is displayed from the console. For a single switch, the console cable that comes with the Dell N-Series switch can be used to view the console output. This is the preferred method for those situations where there are only a few switches involved. See the User Guide for your switch for more information on how to use the console port.

A *console switch* may also be used to allow the administrator to view all switches' console outputs from a single management station. Using a console switch on the network, Figure 3 shows the same network topology as shown in Figure 1 with an added console switch to help manage the network.

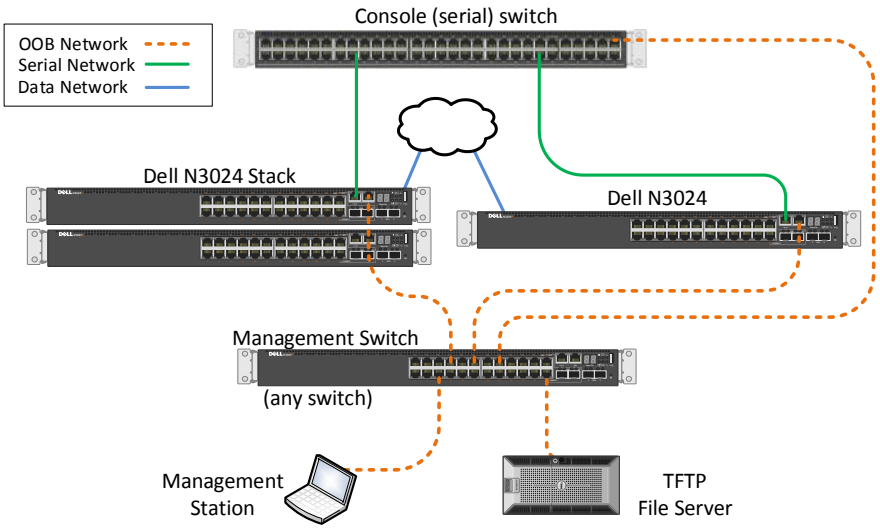

Figure 3 Using a console switch on the network

Here, the console (serial) network and the OOB network are used for management of the network, while the Data network processes the production traffic such as applications and file sharing.

A console switch is not required to run Python scripts on a Dell N-Series switch, but may be helpful when there are several switches requiring the administrator to regularly test or manage Python scripts.

C Glossary of Terms

console – the interface that uses the serial port of a switch for management purposes.

console switch – a device that allows management of several switches from a single management station, via the console port on each switch.

CLI (command line interface) – a text-based interface for issuing commands to a switch (typically uses Telnet, SSH, or a serial console).

module (Python) – a pre-written Python script (from the Python library) available for programmers to import and use in their own programs to save time.

OOB port – the Out-of-Band port, the port used to connect to the management OOB network.

Out-of-band – a separate management network that takes management traffic off of the production network.

Python – a popular, high-level programming language with a vast standard library.

package – similar to a module and often used interchangeably, part of the Python library of pre-written code for users.

serial port - a serial interface through which users may communicate with the switch requiring a single cable, a "console cable," between the switch and a computer.

SSH (Secure Shell)– a network protocol that allows users to securely log into a switch or other system on the same network.

tarball – a single compressed file that contains one or more other files to be transported as one, unpacked, then used in another area or application.

Telnet - a network protocol that allows users to log into a switch's command line interface. Telnet is less secure than SSH.

TFTP – Trivial File Transfer Protocol, a common protocol used to transfer files between local and remote hosts.

VLAN (virtual local area network, or virtual LAN) - logical subgroups that are partitioned off a physical network in order to create separate broadcast domains.

D Versions

This document was compiled using the following components and versions:

Component	Version
Dell Networking N-Series firmware	6.3.0.3
Python (Dell N-Series supported modules)	2.7.10
Python	2.7.6
Ubuntu	14.04.3
gedit	3.10.4
tar (GNU tar)	1.27.1

Additional Resources

Support.dell.com is focused on meeting your needs with proven services and support.

DellTechCenter.com is an IT Community where you can connect with Dell EMC customers and Dell EMC employees to share knowledge, best practices and information about Dell EMC products and installations.

Referenced or recommended Dell EMC publications:

- Dell Networking Support
  - http://www.dell.com/support
- Dell TechCenter (community forums and blogs for Dell EMC customers)
  - http://delltechcenter.com
- Dell Networking Whitepapers
  - http://en.community.dell.com/techcenter/networking/p/guides
- Dell Networking N15xx User Guides and Firmware downloads
  - http://www.dell.com/support/home/us/en/19/product-support/product/networking-n1500-series
- Dell Networking N2xxx User Guides and Firmware downloads
  - http://www.dell.com/support/home/us/en/19/product-support/product/networking-n2000-series
- Dell Networking N3xxx User Guides and Firmware downloads
  - http://www.dell.com/support/home/us/en/19/product-support/product/networking-n3000-series
- Dell Networking N4xxx User Guides and Firmware downloads
  - http://www.dell.com/support/home/us/en/19/product-support/product/networking-n4000-series

Support and Feedback

Contacting Technical Support

Support Contact Information	Web: http://Support.Dell.com/ Telephone: USA: 1-800-945-3355

Feedback for this document

We encourage readers of this publication to provide feedback on the quality and usefulness of this deployment guide by sending an email to Dell_Networking_Solutions@Dell.com

About Dell EMC

Dell EMC is a worldwide leader in data center and campus solutions, which includes the manufacturing and distribution of servers, network switches, storage devices, personal computers, and related hardware and software. For more information on these and other products, please visit the Dell EMC website at http://www.dell.com.