Net-Sense Automater
Users Manual

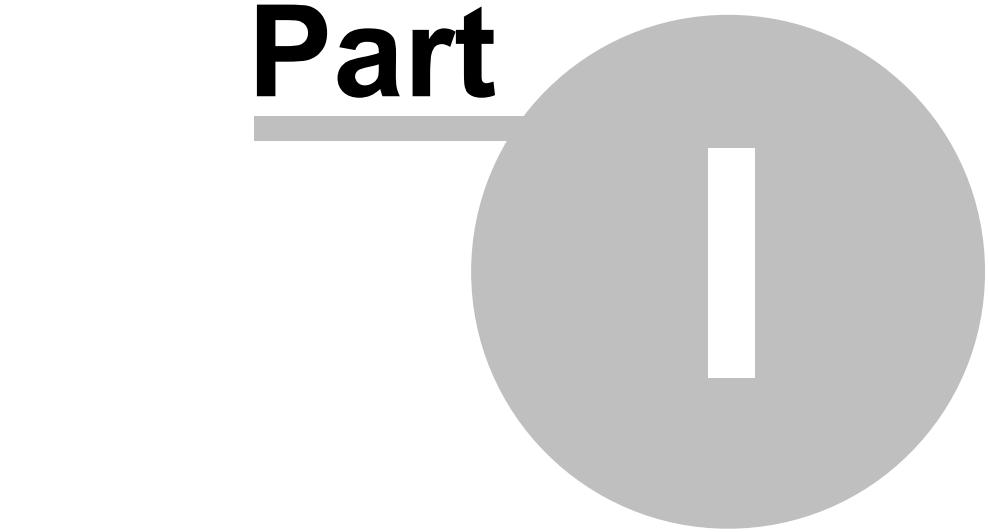
Version 5.3

Net-Sense Automater

Printed: April 2017 in (New Jersey, US)

Table of Contents

Introduction (Version 5.3.0)



1 Introduction (Version 5.3.0)

The Automater is a program which is comprised of a series of individual Scripts which automate many common tasks performed on Cisco Networking Equipment. These scripts provide engineers with a simple yet extremely powerful set of tools to perform their everyday tasks. A simple task such as clearing the counters on all the routers in a network can take hours or days depending on the size of the network. The Automater can do this in minutes.

Each of the Scripts included with the Automater was created from the "mind-set" of Engineers working on real networks or test-beds. It's not that you can't perform these tasks without the Automater, it just allows you to perform these tasks a 1000 times faster and more thoroughly than doing it manually.

Individual scripts can be run from the command line (i.e. UNIX shell prompt) or using the ***Net-Sense Automater GUI.*** In addition, multiple scripts can be combined into a Linux shell script or Windows batch file to run multiple scripts in succession.

The scripts work by telneting (or Secure Shell) to the device and issuing CLI based commands. Depending on the script, the commands issued are either pre-defined or user defined.

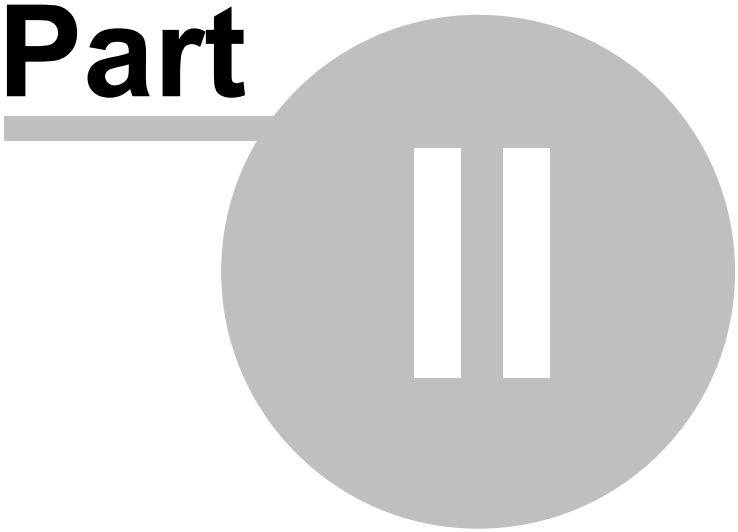
The following Cisco devices are supported:

- Cisco Routers (IOS)
- Cisco Switches (IOS)
- Cisco Switches (NX-OS)
- Cisco Switches (CatOS)
- PIX Firewalls

The Automater also contains a generic script that will work on any type of device that has telnet/ssh connectivity and a CLI interface. Examples include:

- UNIX Systems
- LINUX Systems
- Other Vendor Networking Equipment.

Installation - System Requirements



2 Installation - System Requirements

The Automater is supported on the following operating systems:

- Red Hat Linux 7.2 and above.
- Fedora Core Linux
- Suse Linx 9.2 and above
- Windows NT, 2000, XP, 2003, Vista, 7, 8

2.1 Windows Installation

The Automater can be installed on the following Microsoft Operating systems:

- Windows 2000
- Windows XP
- Windows 2003
- Windows Vista
- Windows 7
- Windows 8

To install the program, just double click on the .exe file and follow the on screen instructions. The default installation directory is ***C:\Program Files (x86)\Net-Sense.***

The first time The Automater is run, it will create a "user" working directory (known as SCRIPT_HOME) and install some user specific files. Different from earlier versions of The Automater, now each user will have their own working directory (where all script log files are stored). The default location of the "working" directory varies depending on the Version of MS Windows. The table below shows the default location for the working directory as well as the directory location of the User Configuration file (setup.var)

OS	SCRIPT_HOME (Default for Script logs)	User Configuration File Location (setup.var)
Windows XP	C:\Documents and Settings\jdoe\My Documents \Net-Sense\net-scripts	C:\Documents and Settings\jdoe\Application Data\Net-Sense
Windows Vista	C:\Users\jdoe\Documents\Net-Sense\net-scripts	C:\Users\jdoe\AppData\Roaming\Net-Sense
Windows 7	C:\Users\jdoe\Documents\Net-Sense\net-scripts>	C:\Users\jdoe\AppData\Roaming\Net-Sense

Note: The working directory can be changed by opening the GUI and choosing Options->Settings and changing the SCRIPT_HOME variable.

2.1.1 Windows License Key

By default, the progam is installed with a Trial License Key. The Trial version is fully functional but is limited to being run against 3 routers/devices. If you purchased a copy of the Automater, a license-key should have been e-mailed to you. Install your license key using from the GUI menu (Help->License).

2.2 Unix/Linux Installation

There are two parts to the installation. First, the system administrator must run the installation script and install the license password file. Second, the end-user needs to set up his/her environment to run the scripts.

2.2.1 Program Installation for System Administrator

The programs are distributed as a UNIX tar file. The tar file should be extracted to a temporary directory and then the ***install.sh*** program should be run. In addition to the programs themselves, there are also several template input files that are required by some programs. Later, the end-user will copy the template file from $ ***INSTALL_HOME/template*** to their own directories and modify their copies.

1. Switch user to super-user (root). (Note, this is **NOT** mandatory but if you'd like to put the program in a central directory where multiple users can access it (such as ***/usr/local/net-sense***) then you will need root privileges during the install. If you would like to install it in another location such as $HOME/net-sense, that will work also. Just make sure you put that directory in your search path. [See steps below])

2. Copy/Move the tar file to a temporary directory (e.g. /tmp). (Note the actual name of the tar file may be different than automater_cb_Linux_4-5-5.tar.gz.)
```
cp automater_cb_Linux_4-5-5.tar.gz /tmp
```

3. Change directory to the temporary directory
```
cd /tmp
```

4. Extract the tar file:
```
tar xvfz automater_cb_Linux_4-5-5.tar.gz
```

5. Change directory into the sub-directory created from extraction of the tar file:
```
cd /tmp/build_051311
```

6. Run the installation program and answer the questions:
```
./install.sh
```

   Installation Note: The default installation directory is **/usr/local/net-sense**. The executables are placed in the ***bin*** subdirectory of the installation directory (i.e. /usr/local/net-sense/bin). This directory must be in the PATH environment variable for the users running the scripts. For example, if the executables are installed in ***"/usr/local/net-sense/bin"*** then the user's ***.profile*** (or similar startup file, .bash_profile, etc.) must be edited to contain the following (See Section 3.2 for another example on setting up your PATH):

```
PATH=$PATH:/usr/local/net-sense/bin

export PATH
```

```
[root@linux-1 build_051311]# ./install.sh

...< Licensing Info > ...
..
.

Do you accept the terms of the license [agree/no]? agree

 Enter the directory to install the scripts
 and other files [/usr/local/net-sense]?

 The directory </usr/local/net-sense> does not exist
 Would you like the install program
 to create it [yes]? yes
```

```
Installation Complete!!
Please Review the Documentation for
Setting up Users to Run the Scripts.

[root@linux-1 build_051311]#
```

7. Update the License key file. For Linux, the license key can be installed in one of two ways. One, through the GUI Help-License. Alternatively, you can edit the license key file (i.e. license_key) which is in the same directory that the executable program files were installed in (e.g. /usr/local/net-sense/bin). If you did not receive a license key with your purchase, please send an E-mail to support@net-sense.com.. Note, the programs will be installed with a TRIAL license by default.

8. To confirm the programs are working correctly, run one of the programs using the ***-help*** option. The following error message should appear. If this error message does not appear please contact technical support at ***support@net-sense.com.*** To setup end-users to run the program, follow the instructions in the following section.

9. To view the Help Document through the GUI, a PDF viewer is required (e.g Adobe Reader). For RedHat Linux installations, "xpdf" is used by default. Please install a PDF viewer of your choice if one is not installed on the system.

```
$ /usr/local/net-sense/bin/program_name -help

*****************************************************
* For more information about Script Automation
* or support issues, contact NetSense Technical Support
* E-mail: support@net-sense.com
*****************************************************

ERROR!!!!
 The file <setup.var> does not exist!!!!
 The file <setup_template.var> should be edited
 for your environment and saved as <setup.var>
```

To confirm the programs are working correctly, run one of the programs using the ***-help*** option. The following error message should appear. If this error message

2.2.2 End User Installation/Setup Procedures

Before the scripts can be run, each user must perform the following, one time, installation steps.

1. Add the ***net-sense/bin*** directory, where the system administrator installed the scripts (e.g. **/usr/local/net-sense/bin**), to your path. This is done by modifying the PATH variable in each users ***$HOME/.profile*** file. Issue the command ***env | grep "^PATH"*** to confirm.

   Example (Existing entry in .profile)
   ```
   PATH=/usr/sbin
   export PATH
   ```

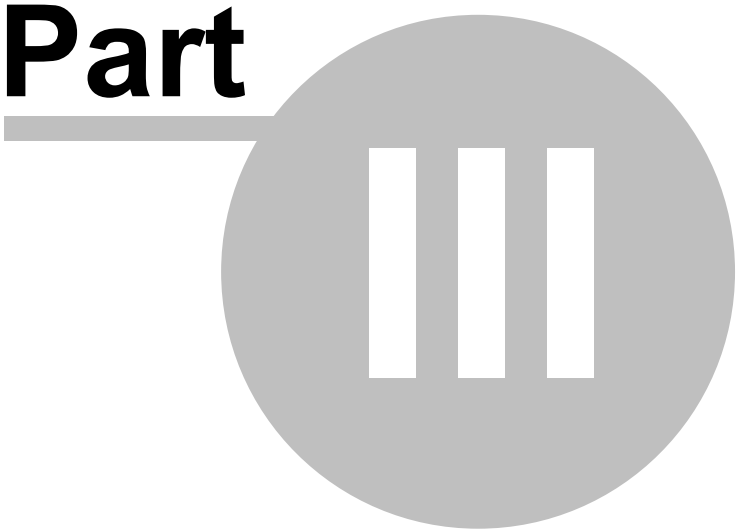
   Change to:
   ```
   PATH=/usr/sbin:/usr/local/net-sense/bin
   ```

```
export PATH
```

After the .profile file has been edited, you will need to log out and log back in again for your .profile to be re-read. Confirm the new directory has been added to your path by issuing the following:

```
linux-1:net-scripts> env | grep "^PATH"
PATH=/usr/local/bin:/usr/bin:/etc:/usr/sbin:/usr/ucb:/usr/bin/X11:/sbin:/common/b
in:/usr/local/net-sense/bin:.
```

Login/Password Managment



3 Login/Password Managment

There are two ways to provide router (or other device types) usernames/passwords for the scripts. Depending on your specific environment and requirements, you will use one of the options below.

- One, username/passwords can be stored in a file and that file is referenced as a script option (with the ***-pw*** option). This option should be used if any of the items below apply:
  - Scripts will be scheduled to run at a later time
  - Network Devices (routers, switches, etc.) all have different passwords
  - Multiple scripts will be packed into Linux shell wrappers scripts or Windows batch files

OR

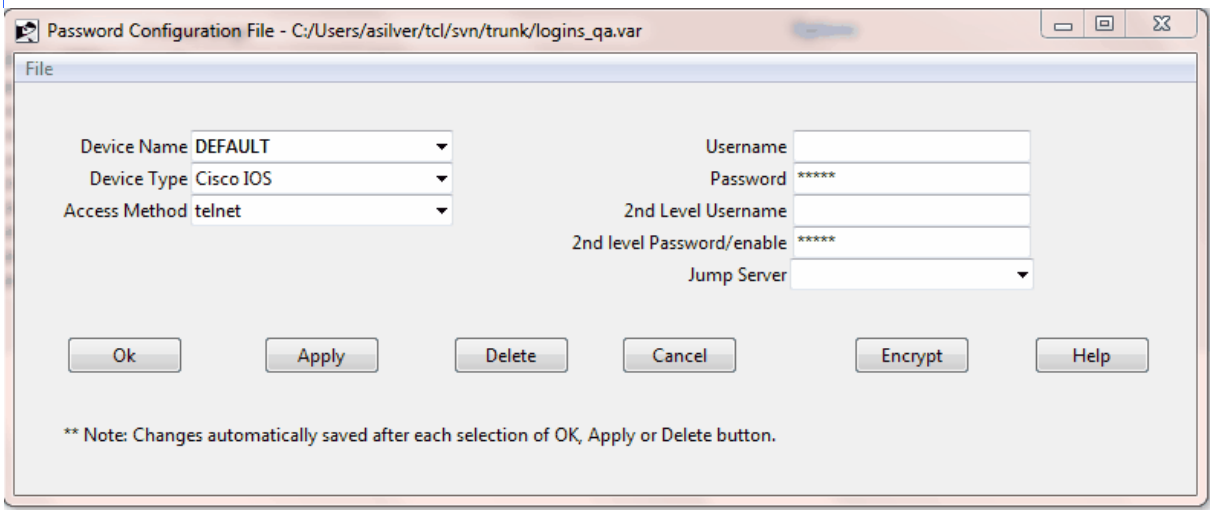
- Two, when running a script the script will interactively prompt the user to enter in the username/password information; in this case there is NO password file. This second method will automatically happen if the ***-pw*** option is NOT used for a script run.

3.1 Device Passwords

The login/password file is used to store Usernames and Passwords for your network devices. This feature is beneficial when you have different device types and passwords for your network devices. If all of your devices of are the same type and use the same username/password, then you can omit a password file and each time a script is run, the script will prompt you for a password. If a password file is needed, it should be encrypted using using the GUI (Options->Device Passwords).

Note, if you'd like you can create as many password files as you'd like.


Starting with version 4.5.6 the password file can be created, maintained, and encrypted all through the GUI. If encryption of the password file is not required, then you can also create and maintain the password file through a basic text editor (e.g., vi or windows notepad).



Each script can access the password file by using the -***pw <filename>*** option on the command line when running the script. When used, the script will search the ***Device Name*** field in this file (***case sensitive search***) for the IP Address/Name of the router the script is telneting/ssh into. The exact "IP Address/Name" it searches for is the "IP Address/Name" that was defined in the list of routers (-rf

<filename>) or on the command line (-ipaddr <name or ip address>). Thus, it is important that these two values match or the login information will not be found for a particular router (***case sensitive***).

One useful feature is the DEFAULT password option. This is the login/password information used for a device if the device name is not explicitly defined in the password file. Suppose you have 100 routers in your network and they all use the same username/password except one. In this case you would only need to define two devices in your password file; the DEFAULT entry and the one router with a unique username/password. The name **DEFAULT** is entered in the "Device Name" field and **MUST BE ALL CAPITAL LETTERS.**

As stated above, you can create multiple password files or you can have one password file with all of your devices. Here is an example where it makes sense to have two password files. Suppose you have 50 IOS routers and 25 PIX/ASA devices and all of the routers use a single username/password and all of the FWs use a different single username/password. You would create two separate password files. Each file would only need to contain a single DEFAULT entry. Note, even if the FWs and routers used the same username/password, you would still need 2 separate files because they are different "Device Types". If you wanted to use one password file for this scenario, then the most efficient method would be to create a DEFAULT entry for the 50 routers and then create 25 individual entries for each of the FWs (26 entries in all).

If trying to create, and maintain, your own password file using a text editor, the program comes with a sample password file (logins_template.txt) that should be used as a template. Below are two sample entries for a password/login file.

```
lappend ALL_ROUTERS [list C "10.10.1.1" "" "foobar" "" "foobar" "telnet" "" ] ;# rtrnj1
lappend ALL_ROUTERS [list C "nyrtr" "allan" "mypasswd" "" "my2ndpasswd" "ssh" "" ] ;# nyc
router
```

Each entry takes up one line in the file and must have the following format:

lappend ALL_ROUTERS [list *F1 F2 F3 F4 F5 F6 F7 F8*] **;# comment**

The following table explains fields 1 through 7. An informational comment can follow the **;#** at the end of each line.

Field	Description
F1	**Router Type**: Can be one of the following: **C** : Cisco Router or Switch **CAT** : CatOS based Switch **G1** or **G2** for Generic Type 1 and Type 2 (See Section 6.3) **P** : PIX Firewall **N** : Nortel Router (Limited Support) **E :** Efficient DSL Router (Limited Support)
F2	IP Address or Router Name. This is the IP address used to telnet to the router. This field can either be an IP Address or a name that can be resolved through DNS (/etc/hosts, etc.)
F3	1st level Username. This is used if TACACS or local Usernames are setup on the router. If this does not apply, enter two double quotes (e.g. “”) for a place holder. (Note, does not apply for Efficient Routers)
F4	1st level password. If TACACS is enabled, this is the Username password. If TACACS is not enabled, this is the “typical” 1st level password for the router.

	This is the password used for Efficient Routers.
F5	2nd level Username. This would not typically apply, even when TACACS is being used. This would only apply if the router responded with a **Username** prompt when trying to enter enable mode on the router. If this does not apply, enter two double quotes (e.g. “”) for a place holder. (Note, does not apply for Efficient Routers)
F6	2nd level password. This is the Cisco password that would be entered after typing in the exec command **enable**. This field can be left blank (i.e. “”) if 2nd level access is not required for the scripts you plan on running (e.g. ios_report). (Note, does not apply for Efficient Routers)
F7	Specifies whether **telnet** or **Secure Shell** should be used to access the routers. For telnet access, this field should be “***telnet***”. For secure shell access, this field should be “***ssh***”. (Note, does not apply for Efficient Routers or Nortel Devices)
F8	Jumpserver Device. By default this field should blank (i.e, ""). When specified, access to the device specified in field F2 will go through the jump server. More specifically, the script will first log in to the jumpserver device and then from there it will ssh/telnet to the device in field F2.

If creating a DEFAULT password entry, the default entry MUST contain the word DEFAULT for F2 and **this MUST BE the last entry in the file (MUST BE ALL CAPITAL LETTERS).** If this is not the last entry in the file, it will not be considered the default password. It will be considered a router with a name of DEFAULT. Since there can only be one last entry, you cannot have multiple default entries, only the last one in the file will be considered the default.

Below is a copy of the sample template file (***logins_template.txt***) that is included with the program. For Unix/Linux installations, this file should have been copied into your directory where you run the scripts from; Step **2** (End User Installation/Setup Procedures). For MS Windows, the file is in c:/Program Files/net-sense/userdata (assuming the default installation directory was accepted). The actual template file contains detailed informational comments.

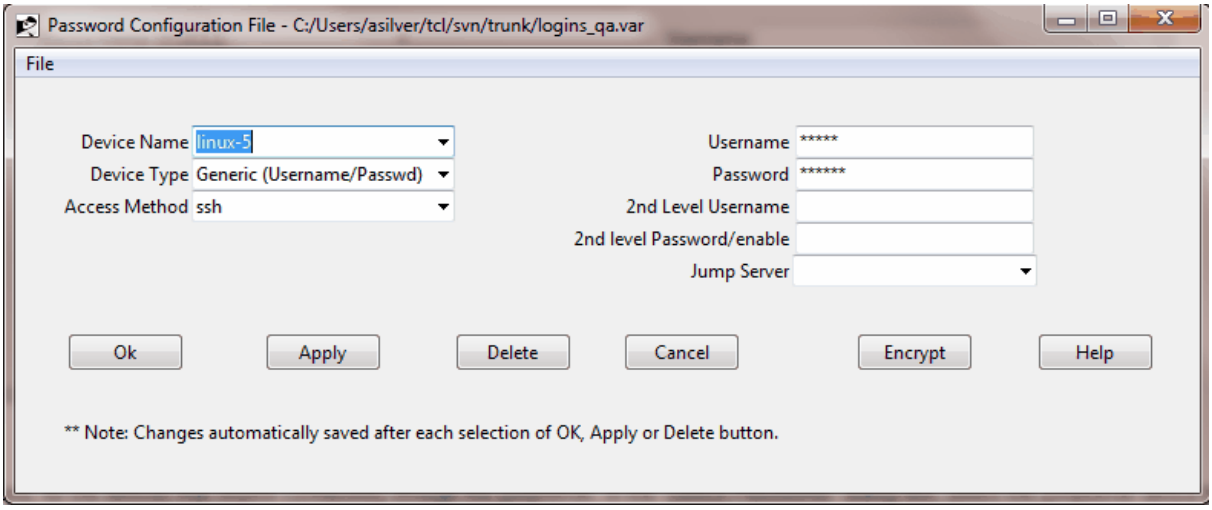
```
set ALL_ROUTERS ""
lappend ALL_ROUTERS [list C "10.10.1.1" "" "foobar" "" "foobar" "telnet"
"" ] ;# SanFran 1720
lappend ALL_ROUTERS [list C "nyrtr" "allan" "mypasswd" "" "my2ndpasswd"
"telnet" "" ] ;# NY router
lappend ALL_ROUTERS [list C "10.10.2.1" "" "foobar" "" "foobar" "ssh" ""
] ;# LA 7500
lappend ALL_ROUTERS [list C "M2" "" "foobar" "" "foobar" "ssh" "" ] ;# MI
3660
lappend ALL_ROUTERS [list C "10.10.9.1" "" "foobar" "" "foobar" "telnet"
"" ] ;# FL 4500
lappend ALL_ROUTERS [list C "10.10.17.1" "" "foobar" "" "foobar" "telnet"
"" ] ;# NJ 12000
lappend ALL_ROUTERS [list C "DEFAULT" "" "abcde" "" "ddffgg" "telnet" ""
] ;# Default password
```

3.2 Jump Servers

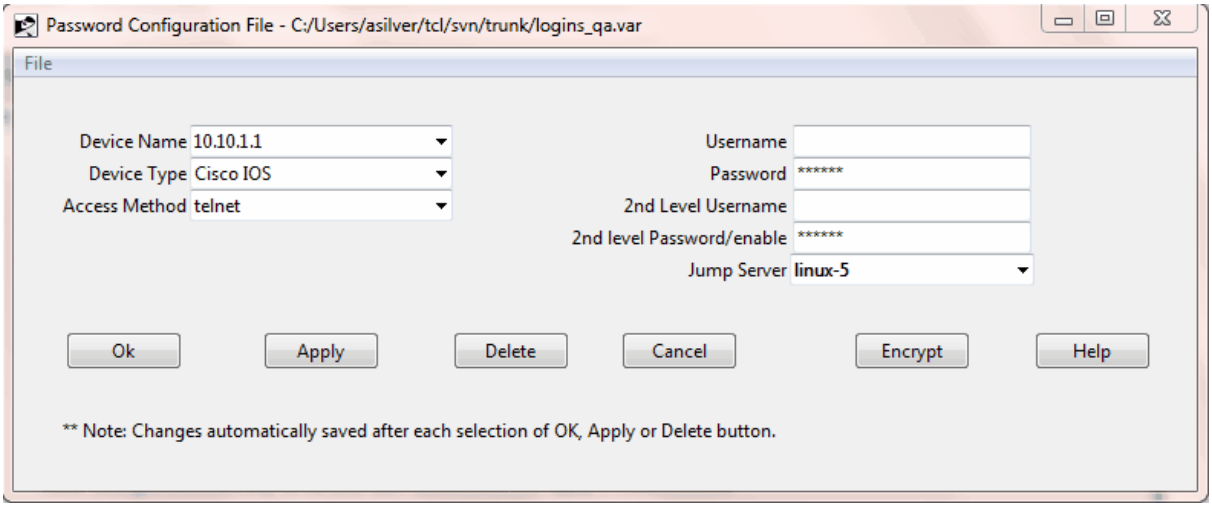
In some cases, routers/switches or other devices you are trying to run scripts against are not directly reachable from the system that The Automater application is installed on. Typically, when manually trying to reach these types of devices, you would first need to telnet/ssh to a jumpserver and then from

there you would telnet/ssh (i.e. "jump") to the router/switch/device you are trying to get to. Starting with version 4.5.6, The Automater supports reaching networking devices that are behind a jumpserver. Jumpservers are typcially Linux/Unix systems or possibly other routers. When using this feature, it is recommended to configure the setting using the GUI.

To have a device use a jumpserver, several configuration steps must first be followed. First, the jumpserver itself must be defined in the password file. This entails entering login and password information for the jumpserver device. The figure below shows login information being entered for the linux server "linux-5" which will be used as a jumpserver.

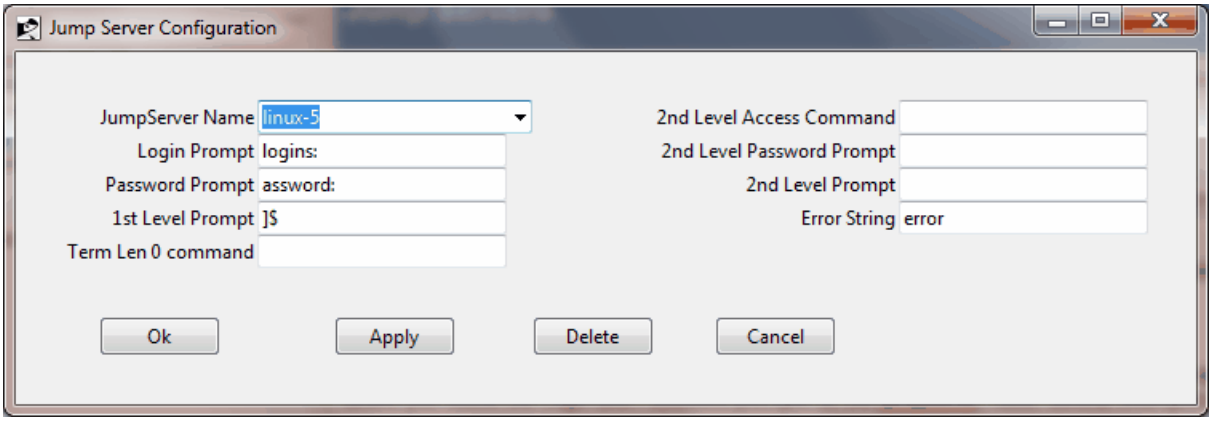


Second, for the devices that require connectivity through the jumpserver, in the "Device Passwords" dialog box, select the jumpserver device from the pull down box. Below shows the device 10.10.1.1 using the jumpserver linux-5



Lastly, the characteristics of logging into the Jumpserver must be defined. This is done through the Jump Server Configuration window (Options->JumpServer Setup). Here is where you define the Login and Password prompts for the jumpserver. Note, several examples are provided in the GUI.

Importnat: The JumpServer Name field in the window below must exactly match the name of the jumpserver field defined in the Password window above.



3.3 Login/Password Encryption Utility

The encrypt_logins utility provides a method to encrypt the password file **if the GUI is not available** (note it is recommended to use the GUI if available**)**. A password file must be used in environments where passwords are not the same for all routers. If the passwords are the same for all routers, the scripts can be run without the "-pw" option and the script will then prompt the user to enter the password. A password file is also required if the scripts will be launched automatically through Linux cron or through MS Windows' scheduler.

Program Name**: encrypt_logins**

Script Argument	Description
-if <filename>	Input File. Un-encrypted Password File (**REQUIRED**)
-nokey	Using this option creates the encrypted password file in a way that the key does not need to entered when another script uses this encrypted password file. Only use this option if you wish to start any of the scripts in batch mode using the cron utility.
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)

The script takes the un-encrypted password file and encrypts it. When the script is run, the user must enter an encryption key that must be at least 6 characters long.

This same encryption key will need to be entered when running any of the other scripts and using an encrypted file with the -pw <filename> option. Below shows an example of running the script:

```
[allan]$ ./encrypt_logins -if logins.txt -of encrypted_logins.txt

*****************************************************
* For more information about Script Automation
* or support issues, contact Technical Support
* E-mail: support@net-sense.com
*****************************************************

Please enter encryption key. You have 90 seconds



[asilver@localhost tcl]$
```

Running the Scripts

Part IV

4 Running the Scripts

Running the Scripts

All of The Automater scripts are designed to have a similar “look-and-feel” and can be run from the either the GUI or command line. The options used for the individual scripts are identical whether they are run from the command line or the GUI. Most users may want to use the GUI until they become familiar with the scripts and their options. In addition to being more intuitive, the GUI also offers additional pop-up help windows. The command line interface is valuable when the scripts are installed on a central Unix/Linux Server and are remotely accessed. The command line is also used when combining multiple scripts into a shell script or batch file.

In many cases the scripts require input files. The table below shows file locations for the template files along with the directory used by the GUI to store files. Note, the location of the GUI Program Files is included for completeness. There should be no need to edit these files.

OS	Template File Location	GUI Program Files
Windows XP	C:\Documents and Settings\jdoe\My Documents \Net-Sense\templates	C:\Documents and Settings\jdoe\Application Data\Net-Sense
Windows Vista	C:\Users\jdoe\Documents\Net-Sense\templates	C:\Users\jdoe\AppData\Roaming\Net-Sense
Windows 7	C:\Users\jdoe\Documents\Net-Sense\templates	C:\Users\jdoe\AppData\Roaming\Net-Sense
Linux	$HOME/.net-sense/templates	$HOME/.net-sense

4.1 From the GUI

The Net-Sense Automater GUI is started as follows:

MS Windows:

Double Click on the Automater Icon on the Desktop

Unix/Linux:

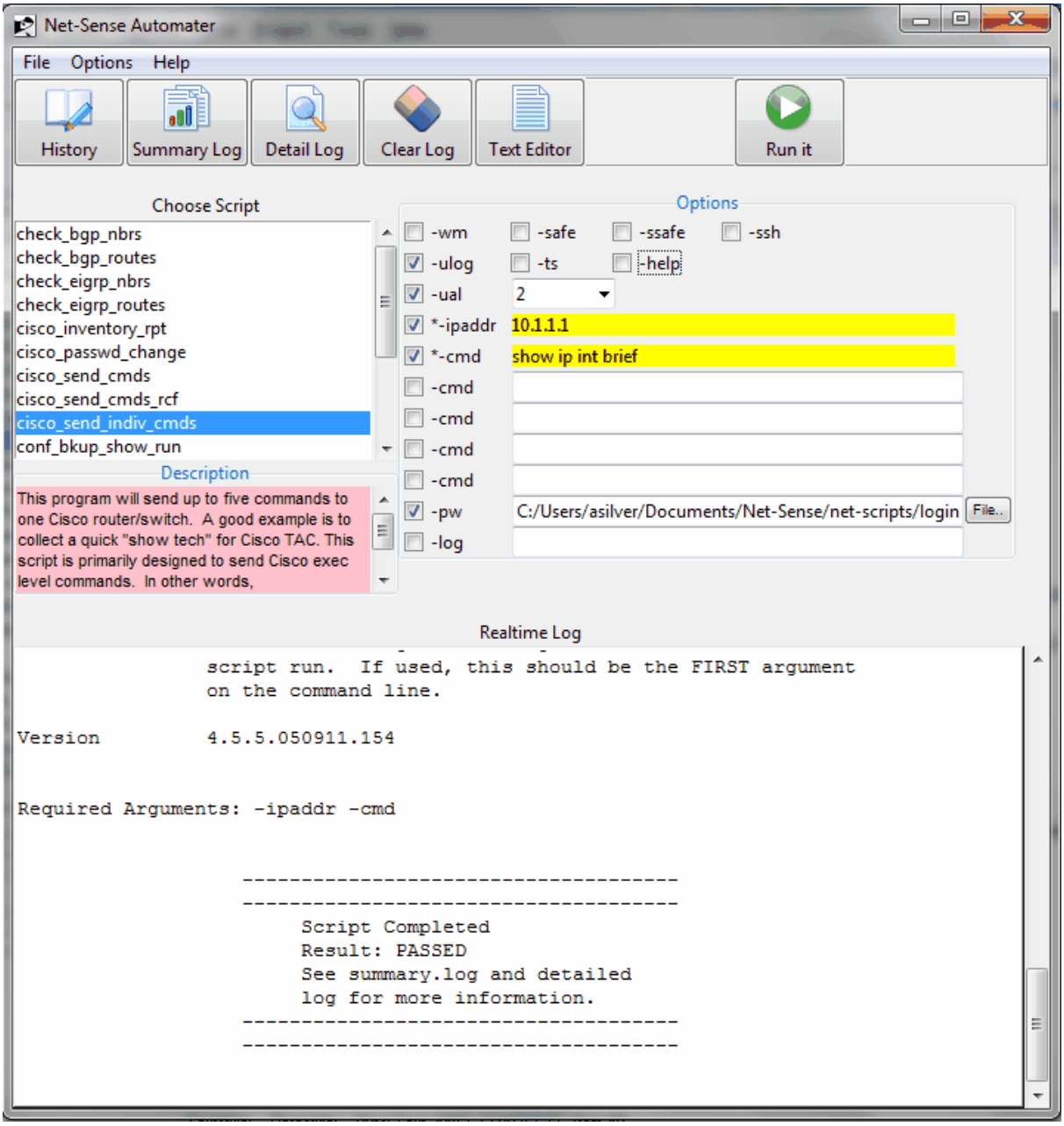
1. Make sure you've performed the "End User Installation/Setup Procedures" (one time effort)
2. Type in **automater** at the command prompt.

```
[asilver@localhost net-scripts]$ automater
```

GUI Script Run Steps

1. Choose a script
2. Select Script Options (Fields in yellow are **required**)
3. Run the Script
4. View the Summary Log (Information at end of file)
5. If necessary, view the Detailed Log

Figure 1 shows the ***Net-Sense Automater*** GUI. The scripts are run by choosing a script and selecting the relevant options and then clicking the ***Run it*** button. The output of the script is written to the Log Window.



4.2 From the Command Line

Running Scripts from the Command Line

All of the scripts can be run from the command line for both Unix/Linux and MS Windows. They are run by typing in the individual program name and options. There are slight differences in the setup

between running the scripts from Unix/Linux and MS Windows. Note, it is much more common to run the scripts from the command line in Linux, than it is for Windows. In Linux, shell scripts can be built which may contain many scripts all in one shell wrapper.

When running the scripts from the command line, the ***setup.var*** file is read from the current directory you are running the script from. If the ***setup.var*** does not exist in the current directory, it will automatically be created by the program. When it is created, the ***SCRIPT_HOME*** variable will get set to the current directory.

For MS Windows, the scripts are run from the ***cmd tool*** window (Start->All Programs->Accessories->command promt).

Although the scripts can be run from any directory, in order to stay organized, it is recommended they be run from the following directories or any new directories you create under these directories:

Linux: $HOME/net-scripts
Windows: My Documents\Net-Sense\net-scripts

Below is a list of executable filenames for each of the scripts when run from the command line:

Cisco IOS Scripts	*Cisco Pro Scripts*	*Nexus (NXOS)Scripts*	*Cisco CatOS Scripts*	*Cisco PIX Scripts*	*Generic Scripts*	*Miscellaneous*
cisco_inventory_rpt	check_bgp_nbrs	**nxos_send_cmds**	**catos_send_cmds**	**pix_send_cmds**	**generic_send_cmds**	**encrypt_logins**
cisco_send_cmds	check_bgp_routes	nxos_config_cmds				
cisco_send_cmds_rcf	check_eigrp_nbrs					
cisco_passwd_change	check_eigrp_routes					
conf_bkup_show_run	config_devices_rvf					
cisco_config_cmds	ios_upgrade					
cisco_config_cmds_rcf						
copy_to_tftp						
ios_report						
pinger						
tracer						

There are some generic options which may or may not apply to all of the scripts. In addition, there may be some required arguments as well as some script specific options. If the required arguments are not entered on the command line, the program will report an error. All programs can be run with the "***-help***" option to see generic options as well as the required arguments for the script. Below, shows an example of the ***ios_report*** program run with the -***help*** option.

```
$ ios_report -help

*******************************************************
```

```
* For more information about Script Automation
* or support issues, contact Technical Support
* E-mail: support@net-sense.com
*****************************************************

-of <file>      Filename where output results are written

-pw <file>       Option for turning interactive off. When
                specified, script will not prompt user for passwords
                The filename with the login information must also
                be specified

-sf <file>      Sources the following file. Used if you need
                to provide additional input to the script
-rf <file>      File with a list of routers to run script against
                If this option is not required, then the script
                is not designed to cycle through a list of routers
-log <file>     File to save detailed trace of script run.  If omitted,
                the default trace log file is <script_name.log>.  The
                default trace log file will get over written each
                script run.  If used, this should be the FIRST argument
                on the command line.

Required Arguments: -of  -rf

$
```

Some options require a filename to be specified. In some cases that file will contain information that is needed by the script. Depending on the option, the format of the file may be a simple text file containing a list of routers or the file may be in TCL format. For the files that are in TCL format, TCL programming knowledge is NOT required. The TCL files edited by the user are only used to set script input variables. The only TCL knowledge required is the following:

- Lines that begin with a number sign "#" are comments.
  - Example:
    - # This line is a comment!!!
- Comments can also be entered in the middle of a line as long as they are followed by a ";# ".
  - Example:
    - set VTY_START "0" ;# Start at vty line 0
- Variables are set with the TCL ***set*** command:
  - **set *variable* "*value*"**
  - Example: set RTR_IP "10.1.1.1"

4.3 Post Script Run Checks

After a script has completed it is important to check the ***summary.log*** file. From the GUI, just click the ***Summary Log*** button (located in the middle of the GUI) and the file will be displayed. For command line script runs, this file is located in the same directory that the script is run from. This file reports any type of errors that occurred while the script was run. Furthermore, the contents of this file are cumulative. Meaning, every time this script is run, the results are appended to this file. If there were no errors, the script reports:

+++++++++Script Passed Test!+++++++++

If there were errors, the script reports:

"###### There were errors! ######"

along with additional information about the error.

Below shows a portion of the ***summary.log*** file for several script runs. This file can be viewed using the UNIX **more** <*filename*>, the **cat** <*filename*>, or the **tail -20** <filename> commands. This extract shows results for two script runs, the most recent run having errors.

```
[allan@linux-1 tcl]$> more summary.log

------------------------------------------------------------
Tue Dec 31 18:16:16 EST 2002
------Results for effic_passwd_change.tcl ------.
------Script Arguments: -rf ef.rt -save -hide

+++++++++Script Passed Test!+++++++++
------------------------------------------------------------
Tue Dec 31 18:17:49 EST 2002
------Results for effic_passwd_change.tcl ------.
------Script Arguments: -rf ef.rt -save -hide

ERROR20 Loop NA: Proc_bug: Login Failed for <cpe>. Possible bad password
ERROR20cont Loop NA: Skipping device.  This may cause problems for rest of scri
ERROR20 Loop NA: Proc_bug: Login Failed for <co>. Possible bad password
ERROR20cont Loop NA: Skipping device.  This may cause problems for rest of scri

########## There were errors! ##########
```

If the summary log shows that a script run had errors, then a more detailed investigation as to why the error occurred may be necessary. This is accomplished by looking at a more detailed log file of the script run. The name of the detailed log file is the "script_name.log". (For example, the default trace-log for the Cisco IOS Report script is ***ios_report.log***.) This file is located in the directory where the script is run from and contains a complete trace of everything that was displayed to the screen as well as the error message that was in the ***summary.log*** file. (NOTE: This file only contains the contents of the most recent script run. The contents of the previous script run are removed!!! (If you wish to keep the contents of this file, save this file to another name or use the -***log*** option on the command line when running the script) Using the error message in the sample ***summary.log***above, the user would search the ***effic_passwd_change.log*** file for the string "ERROR20" to obtain more information about the problem. Performing that search below, shows that the router reported "Wrong password!" when attempting to log in. In this particular case, the wrong passwords were entered to login to this router.

```
Connected to cpe.
Escape character is '^]'.

Efficient 5851 SDSL [ATM] Router (5851-005) v5.3.160 Ready
Login: *
Wrong password!   Try logging in again.
Login: ERROR20 Loop NA: Proc_bug: Login Failed for <cpe, Efficient device>. Po
ERROR20cont Loop NA: Skipping device.  This may cause problems for rest of scr
```

4.4 Generic Parameters

There are several generic/global parameters that affect all of the scripts. These generic parameters are kept in the ***setup.var*** file. Most of the settings below can be modified using the GUI (Options->Settings). For Linux installations, the ***setup.var*** file is located in the *net-scripts* directory under each user's home directory (i.e. $HOME/net-scripts/setup.var). For MS Windows, the file

location varies depending on the OS version. Listed here.

Changing the default values is optional. The table below outlines these parameters .

Parameter	Description
SCRIPT_HOME	The directory where the user runs the scripts from. For Unix/Linux, this must be changed from the default and is discussed in Section 3.2.2. For MS Windows, does not need to be changed if the default installation directory was accepted. ***Can also be viewed using the GUI (Options->Settings)***
log_user	This controls whether the script displays the complete output to the screen as it accesses each router/device. The value can be 0 or 1. A 1 will show the output and a 0 will hide the output. If you are using the GUI and running an X Window session to remotely access the ***Automater***, it's recommended the value be set to 0 for performance optimization. Also note, this parameter does not have the keyword " ***set***" preceding it. ***Can also be changed using the GUI (Options->Settings->Verbose Log Display)*** Default: 1
timeout	The amount of time (in seconds) the script will "wait" for an expected output. If it does not receive the expected output within the time frame, an error is logged. For example, when a script attempts to telnet into a router but does not get the password or login prompt within a certain amount of seconds (i.e, the timeout value), an error is logged. ***Can also be changed using the GUI (Options->Settings->Expect Timeout)*** Default: 12 seconds
DELAY_INTERVAL	The delay (in seconds) between telneting to different routers. The default is zero (no delay) but you may want to set this to a higher value is you would like to "slow things down" or try to read/watch the display as the script is running. ***Can also be changed using the GUI (Options->Settings->Delay Between Telnets)*** Default: 0 seconds
CUSTOM_LOGIN_PROMPT	This variable is actually commented out and should only be uncommented and set if you have a custom login prompt for your routers. For example, the default login prompt when using TACACS is "Username:" but this can be customized by the network administrator. If a username is required for access but the prompt for the username is not "Username:", then uncomment out this variable and set the appropriate login prompt. Default: Variable is commented out
CUSTOM_PASSWORD_PROMPT	This variable is actually commented out and should only be uncommented and set if you have a custom password prompt for your routers. For example, the default password prompt when using TACACS is "Password:" but this can be customized by the network administrator. If the default password prompt has been changed, then uncomment out this variable and set the appropriate password prompt. Default: Variable is commented out
EDITOR	This is the program used when the "Text Editor" button is pressed on the GUI. Each OS has a different default value. The

	setup_template.var file lists more choices to use. Defaulst: Linux: gedit Solaris: dtpad Windows: notepad

4.5 Generic Command Line Options

There are a common set of command line options that apply to all of the scripts. They are described below:

-pw <file>

This option turns on automatic reading of the specified login/password file. This is an optional argument for all of the scripts. If this option is omitted, the user will be prompted for password/login information at the beginning of each script run. Omitting this option is fine when all routers that the script is run against have the same login/password. If the script is run against a number of routers that have different login/passwords, then this option must be used. A sample login/password file (***logins_template.txt***) is provided and should be used as a template for creating your own login/password file. The template file contains instructional comments on how to modify this file. **WARNING: This password file should be encrypted if it stores passwords for "production" routers. Encryption of this file is done using the *encrypt_logins* program.** See Section 6.1 for more information on the password encryption utility.

-sf <file>

sf stands for "source tcl file". This is the argument used for scripts that require input variables. These variables can be easily changed by editing the file. A sample template file is always provided for scripts that require this argument. The name of the template file will be "***program_name_template.txt***". This file is in TCL format. The template file should be edited and saved to a new filename (preferably with a .txt extension). The template file will contain instructional comments for the configurable variables. This argument is only used when the help menu for the program states that it is a required argument. (NOTE, THIS OPTION ONLY APPLIES TO SEVERAL OF THE SCRIPTS)

-rf <file>

rf stands for ***router file***. The router file contains a list of routers that the script should be run against. This file is NOT in TCL format. Routers must be specified as one per line. Routers can be specified as an IP address or as a name that can be resolved to an IP address through DNS. This argument is only used when the help menu for the program states that it is a required argument. Lines that begin with a "#" are considered comments. **IMPORTANT**: The name/ip address used for the router in this file, must be the same exact name used in the login/password file. Otherwise, the login/password information for that router will not be found in the login/password file. (This is only relevant when the -pw <filename> option is used).

Below shows an example of a ***router file***, named *east_coast.rt*

```
[allan@localhost tcl]$ more east_coast.rt
##########################
# East Coast Routers
##########################
10.10.1.1
10.10.2.1
10.10.3.1
10.10.4.1
10.11.2.1
```

```
10.11.3.1
ny-rtr1
ny-rtr2
```

-ssh — Tells the script to use Secure shell, instead of the default telnet, when accessing the routers. This argument is only relevant when the script prompts the user for the login information (i.e. the login/password information is NOT specified with the -pw option). **If the -pw option is used, this argument is irrelevant!**

-log <file> — File to save detailed trace of script run. By default, the detailed trace log file is <***program_name.log***>. This default trace log file gets over written each script run. If you wish to save the detail-trace log to a separate filename, then use this option. NOTE, if used this should be the FIRST argument on the command line.

-ulog <file> — Automatically create a unique log file name. By default, the detailed trace log file is <***program_name.log***>. This default trace log file gets over written each script run. If you wish to save the detail-trace log to a separate filename, and have the script automatically create this file filename for you, then use this option. NOTE, if used this should be the FIRST argument on the command line. The file name will be ***program_name_timestamp.log*** (e.g. cisco_send_cmds_081610_22h31m19s.log). The name of this file is written to the summary.log file.

4.6 Script Errors and Aborts

There are several reasons why the script could exit before completion. Two are listed below.

- Prerequisite checks that are built into the script have failed.
- User aborts the script by selecting the Abort button from the GUI or entering “Control C if run from the command line.

4.6.1 Script Initiated Aborts

The scripts have multiple checks that are performed while being run. If any of these checks fail, the script will exit before completion. Below is a list of these possible errors:

- Command line has a missing required argument
- Filename following the -sf argument does not exist or is not readable
- Username or password is not entered within 30 seconds

If any of the above conditions occur, an error message should be displayed which identifies the problem. Below shows a sample error message where the -rf argument was omitted from the command line:

```
[allan@localhost tcl]$ cisco_config_cmds -cf cmd_list

ERROR3 Loop NA: Missing required command line argument <-rf>
ERROR3 Loop NA: Aborting Script

[asilver@localhost tcl]$
```

Another possible reason for the script aborting is there is an error with one of the input files. The input files could be one of the following: setup.var file, the login/password file, or a file used with the -sf argument. The most common example of this would be if the user removes or puts in an extra double quote (“) when editing one of the variables. If this type of error occurs, there will not be a “clean” error message. It will most likely look like a “strange” TCL error message. Below shows an example of one of those variables defined in the setup.var file.

```
set SCRIPT_HOME [file nativename "/home/allan/net-scripts"]
```

4.6.2 User Initiated Aborts

The user can abort the script at any time by performing one of the following:

- Type ***Control-C*** when script is run from the Command Line
- Click the "***Abort***" button on the GUI.

This will cause the script to immediately exit and not send any more commands to the routers. The detailed trace log file is still available but the file name will be script_name_raw.log. This file is not “cleaned-up” and may have some extraneous characters but it will provide an adequate trace of exactly where the script stopped.

Scripts

Part V

5 Scripts

This section discusses the details and options of each of the individual programs. Because the scripts can modify a large number of routers in a short period of time, the following precautions should be taken:

- Run the scripts against test routers until you become familiar with the scripts and their options. Here you can create intentional errors to see how a script behaves and learn how to easily track errors from the ***summary.log*** file to the detailed trace-log file (i.e. ***script_name.log***).
- Always run the script with the specific options against several test routers before running it against a large number of production routers. This will ensure that the script is doing what you intended it to do.
- If you have various software versions on your routers, when possible perform test runs against this software version on non-production routers first.
- When making large changes, run the script against a small subset of routers first and then confirm the results. If the desired results are obtained, run the script against the remainder of the routers.
- Possibly insert a delay between routers. This is done with the "DELAY_INTERVAL" variable in the ***setup.var*** file (Options->Settings). Here, you can introduce a delay after the script finishes with one router and before it connects to the next router.

5.1 Script Selector

The following table should assist in deciding which script to use.

What would you like to do?	Recommended Script
Send the same ***configuration*** commands to a group of routers	**cisco_config_cmds**
Send ***configuration*** commands to a group of routers but the actual configurations commands are different for each router.	**cisco_config_cmds_rcf** **or** **cisco_config_cmds_rvf**
Send a set of **non-configuration** commands (e.g. show ip route) to a group of routers.	**cisco_send_cmds**
Send a set of **non-configuration** commands (e.g. show ip route) to a group of routers and have the output from each router saved to individual files.	**cisco_send_cmds** with the ***-dir*** option
Send a set of ***non-configuration*** commands to a group of routers but the actual commands are different for each router.	**cisco_send_cmds_rcf**
Perform a clear counters on a set of routers.	**cisco_send_cmds**
Backup the router configs for a set of routers to a TFTP server	**copy_to_tftp**
Backup the router configs for a set of routers but the routers do not have access to a TFTP server	**conf_bkup_show_run**

Add or Modify the Cisco banner type commands including but not limited to: • motd • login • incoming	**cisco_config_banner**
Change any of the following passwords on a set of routers: • secret • enable • vty • console • auxilary	**cisco_passwd_change**
Send the same command(s) to a group of switches running CatOS	**catos_send_cmds**
Send the same command(s) to a group of PIX Firewalls	**pix_send_cmds**
Backup the configurations for PIX Firewalls (using show run)	**pix_send_cmds** with the ***-dir*** option
Verify any-to-any network connectivity by performing pings from a list of routers to a list of IP Addresses	**pinger**
Verify packet paths through the network by performing traceroutes from a list of routers to specified IP Addresses.	**tracer**
Encrypt the password/login file (If the GUI is not available)	**encrypt_logins**
Get an IOS Version Report for all routers in the network	**ios_report**
Get a list of all parts and Serial Numbers for all IOS devices in the network	**cisco_inventory_rpt**
Download IOS image to device	ios_upgrade

5.2 Cisco IOS Scripts

5.2.1 Cisco Command Sender (cisco_send_cmds)

This program will send any number of commands to a single Cisco IOS device or a list of IOS devices. The commands that are sent are defined in a text file (any name), that you create, or specified directly on the command line using the -cmd option. This script is ***different*** than the ***cisco_config_cmds*** script in that this script was primarily designed to send Cisco "exec" level commands. In other words, non-configuration commands. The commands are sent in sequential order from top to bottom of the file or the order in which they are defined if using the -cmd options. This script can also send configuration commands but the actual commands to enter and exit configuration mode must also be included in the list of commands to send.

Note, for global Cisco router configuration changes the *cisco_config_cmds* script is the preferred tool as that script has more comprehensive error checking designed for configuration commands only.

Program Name: cisco_send_cmds

Script Argument	Description
-rf <filename>	List of routers or IP Address/(DNS hostname) to run script against (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-ipaddr <ip_address or routername>	IP address or router/switch name (if defined in hosts file or DNS) Use ***-ipaddr*** to send commands to a single device or ***-rf*** to send commands to list of devices specfied in file. (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-cf <filename>	File which contains a list of commands to send to routers. One command per line. Lines that begin with a "#" are considered comments and will not be sent to the router. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time!
-cmd <command>	Command to send to router. Useful when only needing to send a few commands to a single device or list of devices. It saves you the time of having to create a text file with only a few commands in it. If the command contains spaces then it must be enclosed in double quotes (only when running script from cli, not GUI. This option can be used up to 7 times when used with the GUI. The command line version of this script can repeat this option an unlimited number of times. However, for more than 5 to 7 commands, it is recommended to put the commands in a text file and use the -cf option. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time! Example: ***-cmd "show tech"***
-testmode	Test Mode. Build and display the commands that would have been sent to the devices but DO NOT actually connect to the devices and send the commands. Useful for confirming command looping and variable substitutions are correct or for just double checking the commands that will be sent . List of commands are also written to the file $SCRIPT_HOME/<script_name>_sample_cmds.txt". (**OPTIONAL**)
-wm	Saves configuration file to NVRAM after sending the commands. This would only make sense if any of the commands were configuration commands. This option performs a "write memory" on the router. (**OPTIONAL**)
-ual (1 or 2)	User Access Level. Tells the script what access-level (1^{st} or 2^{nd}) to log into the router before sending the commands. By default the script will only go into 1^{st} level access. (**OPTIONAL**)
-dir <directory>	If specified, tells script to save output for each device into a separate file. Files will be saved to the directory specified. If the directory does not exist, the script will create it. By default output for all devices is only saved to the detailed trace file. The directory entered can be

	either an absolute directory or a relative directory. If running the script from the GUI, a relative directory is relative to the "SCRIPT_HOME" variable. (From the GUI see Options->Settings). (**OPTIONAL**)
-autodir <date | time>	Automatically create new unique directory to save output for each device into a separate file. The choices are ***date*** or ***time***. The ***date*** option will create a new directory consisting of just the date e.g. 08012010 for Aug 1st 2010. The ***time*** option will append the time to the date: e.g. 08012010_12h36m15s If used with the -dir option, the new unique directory will be created under the ***-dir directory***. If the -dir option is not used, then the new unique directory will be created under the ***SCRIPT_HOME*** directory. Note, if the ***date*** option is used and that directory name happens to all ready exist, then files in that directory will be overwritten. There are no safety prompts for the user when using this option.
-urfn	Use Route File Name. When saving the output data to a separate file for each device, the filename used will be the name/IP Address stated in the -rf file. The default filename is the router hostname configured on the router. This option only applies when used with the -***dir*** option. (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The command file (-cf <filename>) should contain a list of commands that will be sent to each router. This is not a TCL file, it is a plain text file that must contain one command per line. **Lines that begin with a "#" are considered comments and will not be sent to the router as a command.**

The following commands can be entered in the command file or on the command line and the script will automatically answer any additional confirmations or prompts.

 (***Note, output of the commands listed below will NOT be displayed in the contents of individual output files when using the -dir option***)

Command	Description

clear counters clear counter	The Cisco IOS ***clear counters*** command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to clear the counters.
clear logging clear log	The Cisco IOS ***clear logging*** (or clear log) command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to clear the log.
clear line <line number>	The Cisco IOS ***clear line*** <line number> command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to clear the line.
clear ip traffic	The Cisco IOS ***clear ip traffic*** command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to clear the ip traffic counters.
reload reload /noverify	The Cisco IOS ***reload*** command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to reload the router. Note, after entering the reload command, the router could respond with a message stating "***the configuration has change, do you wish to save the configuration file before rebooting?"*** If this happens, the router will NOT reload the router, it will log an error message and exit from the router. The reload /noverify option is supported and needed in the case when the "*file verify auto*" command is present in the IOS configuration. When the *file verify auto* command is configured, the router verifies the integrity of the IOS file before reloading the router. If the "*file verify auto*" option is present and the the /noverify option is not used, the Automater script will fail when trying to reload the router. Note, do not use the reload command in the input file and the -wm option on the command line together. If you wish to perform a write mem before issuing a reload, then add the "wr mem" to the input command file before the reload command. (This is because the -wm is performed after all of the commands in the input file are issued but the reload command will terminate the telnet session and the write mem can no longer be performed)
write mem wr mem	If the script sees this string as one of the commands to send, internally it will run a separate procedure that is looking for the string [OK]. If it doesn't see this, it will log an error. Note, you could also just use the -wm command line option if you want to save the configuration after all of the commands are sent.
configure replace config replace	The Cisco IOS ***configure replace*** command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation. In addition, it will also look for the string Rollback Done. If if doesn't see this, it will log an error.
SLEEP <seconds>	The SLEEP command is not a command that will be sent to the router. If desired, this is a method to introduce a delay between commands. Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The SLEEP command must be entered in all CAPITAL letters otherwise it will be interpreted as a command to send to the router. It must also start at the beginning of the line

	(no leading spaces or tabs). Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command. ***CASE SENSITIVE***
RETURN	Using the keyword RETURN is equivalent to sending just a carriage return or Enter key. Not commonly used. ***CASE SENSITIVE***
LOOPSTART <num> or LOOPEND	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will repeated are the commands between the LOOPSTART and LOOPEND commands. See Command Looping for more information. ***CASE SENSITIVE***
COUNTERVAR <variable> <start_num> <increment by>	Define a counter variable. Provides a method to define a variable in the command file; it must be used with the LOOPING feature. The variable must be a single alphabetical character. "start_num" is the number that the counter will start at. It will increment by the <increment by> number each time through the LOOP. See Counter Variables for more information. ***CASE SENSITIVE***

The following is a sample command file which performs a show ip ospf neighbors, show interface, then pauses for 15 seconds because the user wishes to visually examine the output of the show interface command (in real time). Then it clears the counters.

```
show ip ospf neighbors
show interface
SLEEP 15
clear counters
```

There is also an option for setting the User Access Level. By default the script will only log into first level access on the router. If you are sending commands that require second level access, then use the -ual option and set it to 2 (e.g. -ual 2).

If configuration commands are also included in the list of commands and you wish those changes to be saved to NVRAM, then the -**wm** option must be used. By default, the configuration commands will not be saved to NVRAM.

This program also has the option to be run in *Safe* and *SuperSafe* mode which should be considered when running scripts in production environments.

Sample Command 1: The following command will send the commands listed in file ***show_cmds.cmds*** to the routers listed in the file ***rtrs.rt.*** The script will go into enable mode before issuing the commands because of the -***ual 2*** option. If there are any errors while issuing any of the commands, the script will immediately terminate because of the -***ssafe*** option (SuperSafe Mode). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
cisco_send_cmds -pw logins.txt -rf rtrs.rt -cf show_cmds.cmds -ssafe -ual 2
```

Sample Command 2: The following script will send a "show tech" to the router 192.168.0.10***.*** The script will go into enable mode before issuing the commands because of the -***ual 2*** option. The output will be saved to the file show_tech_nyrtr1.log The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
cisco_send_cmds -log show_tech_nyrtr1.log -pw logins.txt -ipaddr 192.168.0.10 -cmd "show tech"
```

Sample Command 3: The following script will issue the command "show interface" 10 times, sleeping for 5 seconds between each time the command is entered. The commands will be sent to the router 192.168.0.10. The script will go into enable mode before issuing the commands because of the ***-ual 2*** option. If there are any errors while issuing any of the commands, the script will immediately terminate because of the ***-safe*** option (Safe Mode). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

Note, the backslash at the end of each line will only work in a Linux/Unix shell.

```
cisco_send_cmds -pw logins.txt -ipaddr 192.168.0.10 -ual 2 -safe \
       -cmd "LOOPSTART 10" \
       -cmd "show int | incl error" \
       -cmd "SLEEP 5" -cmd LOOPEND
```

5.2.2 Cisco Command Sender (cisco_send_cmds_rcf)

This program is very similar to the cisco_send_cmds script except it is designed to send a **different** set of exec-level commands to each Cisco IOS device. The specific commands, and the routers these commands are sent to, are defined in a text file that you create. The commands are sent in sequential order from top to bottom of the file. This script can also send configuration commands but the actual commands to enter and exit configuration mode must also be included in the list of commands to send.

Note, for global Cisco router *configuration* changes, where different configuration commands are needed for each router, the cisco_config_cmds_rcf script is the preferred tool as that script has more comprehensive error checking designed for configuration commands only.

Program Name: cisco_send_cmds_rcf

Script Argument	Description
-rcf <filename>	Short for **R**outer and **C**ommand **F**ile. File which contains a List of routers or IP Address along with a file name which contains the list of configuration commands to send to each router (**REQUIRED**)
-testmode	Test Mode. Build and display the commands that would have been sent to the devices but DO NOT actually connect to the devices and send the commands. Useful for confirming command looping and variable substitutions are correct or for just double checking the commands that will be sent . List of commands are also written to the file $SCRIPT_HOME/<script_name>_sample_cmds.txt". (**OPTIONAL**)
-wm	Saves configuration file to NVRAM after sending the commands. This would only make sense if any of the commands were configuration commands. This option performs a "write memory" on the router. (**OPTIONAL**)
-ual (1 or 2)	User Access Level. Tells the script what access-level (1^{st} or 2^{nd}) to log into the router before sending the commands. By default the script will only go into 1^{st} level access. (**OPTIONAL**)
-dir <directory>	If specified, tells script to save output for each device into a separate file. Files will be saved to the directory specified. If the directory does

	not exist, the script will create it. By default output for all devices is only saved to the detailed trace file. The directory entered can be either an absolute directory or a relative directory. If running the script from the GUI, a relative directory is relative to the "SCRIPT_HOME" variable. (From the GUI see Options->Settings). (**OPTIONAL**)
-autodir <date | time>	Automatically create new unique directory to save output for each device into a separate file. The choices are ***date*** or ***time***. The ***date*** option will create a new directory consisting of just the date e.g. 08012010 for Aug 1st 2010. The ***time*** option will append the time to the date: e.g. 08012010_12h36m15s If used with the -dir option, the new unique directory will be created under the ***-dir directory***. If the -dir option is not used, then the new unique directory will be created under the ***SCRIPT_HOME*** directory. Note, if the ***date*** option is used and that directory name happens to all ready exist, then files in that directory will be overwritten. There are no safety prompts for the user when using this option.
-urfn	Use Route File Name. When saving the output data to a separate file for each device, the filename used will be the name/IP Address stated in the -rf file. The default filename is the router hostname configured on the router. This option only applies when used with the -***dir*** option. (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

Below shows and example of an ***rcf*** file. This is not a TCL formatted file, it is just a text file. Lines that begin with a # are comments. The format of the file is rtrname:filename. Where rtrname is a DNS name or IP Address and filename is a file that contains the list of commands to send to that specific router. One entry per line. The format for the file that contains the specific commands for a particular router is identical to the cisco_send_cmds script.

```
# Sample template file for cisco_send_cmds_rcf script
# Lines that begind with a # are comments
# Format of file is router:filename
```

```
# Where file name contains a list of commands
# that will only be sent to that router

nyrtr1:nyrtr1.cmds
10.1.1.2:cmd_file2.cmds
```

Figure 2 Sample rcf file

Identical to the cisco_send_cmds script, the following commands can be entered in the command file and the router will automatically answer any additional confirmations or prompts. (Note, these commands are not entered in the -rcf file)

Notes:
(These commands are not entered in the -rcf file)
(Output of the commands listed below will NOT be displayed in the contents of individual output files when using the -dir option)

Command	Description
clear counters clear counter	The Cisco IOS ***clear counters*** command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to clear the counters.
clear logging clear log	The Cisco IOS ***clear logging*** (or clear log) command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to clear the log.
clear line <line number>	The Cisco IOS ***clear line*** <line number> command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to clear the line.
clear ip traffic	The Cisco IOS ***clear ip traffic*** command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to clear the ip traffic counters.
reload reload /noverify	The Cisco IOS ***reload*** command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to reload the router. Note, if after entering the reload command, the router could respond with a message stating "***the configuration has change, do you wish to save the configuration file before rebooting?"*** If this happens, the router will NOT reload the router, it will log an error message and exit from the router. The reload /noverify option is supported and needed in the case when the "*file verify auto*" command is present in the IOS configuration. When the *file verify auto* command is configured, the router verifies the integrity of the IOS file before reloading the router. If the "*file verify auto*" option is present and the the /noverify option is not used, the Automater script will fail when trying to reload the router. Note, do not use the reload command in the input file and the -wm option on the command line together. If you wish to perform a write mem before issuing a reload, then add the "wr mem" to the input command file before the reload command. (This is because the -wm is performed after all of the commands in the input file

	are issued but the reload command will terminate the telnet session and the write mem can no longer be performed)
write mem wr mem	If the script sees this string as one of the commands to send, internally it will run a separate procedure that is looking for the string [OK]. If it doesn't see this, it will log an error. Note, you could also just use the -wm command line option if you want to save the configuration after all of the commands are sent.
configure replace config replace	The Cisco IOS ***configure replace*** command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation. In addition, it will also look for the string Rollback Done. If if doesn't see this, it will log an error.
SLEEP <seconds>	The SLEEP command is not a command that will be sent to the router. If desired, this is a method to introduce a delay between commands. Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The SLEEP command must be entered in all CAPITAL letters otherwise it will be interpreted as a command to send to the router. It must also start at the beginning of the line (no leading spaces or tabs). Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command.
LOOPSTART <num> or LOOPEND	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will repeated are the commands between the LOOPSTART and LOOPEND commands. See Command Looping for more information.
COUNTERVAR <variable> <start_num> <increment by>	Define a counter variable. Provides a method to define a variable in the command file; it must be used with the LOOPING feature. The variable must be a single alphabetical character. "start_num" is the number that the counter will start at. It will increment by the <increment by> number each time through the LOOP. See Counter Variables for more information. ***CASE SENSITIVE***

The following is a sample command file which performs a show ip ospf neighbors, show interface, then pauses for 15 seconds because the user wishes to visually examine the output of the show interface command (in real time). Then it clears the counters.

```
show ip ospf neighbors
show interface
SLEEP 15
clear counters
```

There is also an option for setting the User Access Level. By default the script will only log into first level access on the router. If you are sending commands that require second level access, then use the -ual option and set it to 2 (e.g. -ual 2).

If configuration commands are also included in the list of commands and you wish those changes to be saved to NVRAM, then the **-wm** option must be used. By default, the configuration commands will not be saved to NVRAM.

This program also has the option to be run in *Safe* and *SuperSafe* mode which should be considered when running scripts in production environments.

Sample Command: The following script run will telnet to each router listed in the bgp_show.rcf file and send only the commands associated with that router. The script will go into enable mode before issuing the commands because of the ***-ual 2*** option. If there are any errors while issuing any of the commands, the script will immediately terminate because of the ***-ssafe*** option (SuperSafe Mode). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
cisco_send_cmds_rcf -pw logins.txt -rcf bgp_show.rcf -ssafe -ual 2
```

5.2.2.1 Template File

Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\templates\cisco_send_cmds_rcf_template.txt
Linux: /usr/local/net-sense/templates/cisco_send_cmds_rcf_template.txt

```
#########################################################
# Sample template file for cisco_send_cmds_rcf script
# Lines that begin with a # are comments
# Format of file is router:filename
# Where file name contains a list of commands
# that will only be sent to that router.
#########################################################

njrtr1:njrtr1.cmds
10.1.1.2:cmd_file2.cmds
```

5.2.3 Global Router Configuration Tool (cisco_config_cmds)

This program will send any number of **configuration** commands to a single IOS device or a list of IOS devices. The commands that are sent are defined in a text file (any name), that you create, or specified directly on the command line using the -cmd option. This file MUST NOT include the commands needed to enter and exit configuration mode on the IOS device. The script will automatically enter those commands. The commands are sent in sequential order from top to bottom of the file or the order in which they are defined if using the -cmd options.

Program Name: cisco_config_cmds

Script Argument	Description
-rf <filename>	List of routers or IP Address/(DNS hostname) to run script against (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-ipaddr <ip_address or routername>	IP address or router/switch name (if defined in hosts file or DNS) Use ***-ipaddr*** to send commands to a single device or ***-rf*** to send commands to list of devices specfied in file. (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-cf <filename>	File which contains a list of configuration commands to send to router. These MUST BE configuration commands only!!! (Do not include commands to enter and exit configuration mode) One configuration command per line. Lines that begin with a "#" are considered comments and will not be sent to the router.

	(**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time!
-cmd <command>	Command to send to router. Useful when only needing to send a few commands to a single device or list of devices. It saves you the time of having to create a text file with only a few commands in it. If the command contains spaces then it must be enclosed in double quotes (only when running script from cli, not GUI. This option can be used up to 7 times when used with the GUI. The command line version of this script can repeat this option an unlimited number of times. However, for more than 5 to 7 commands, it is recommended to put the commands in a text file and use the -cf option. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time! Example: ***-cmd "ip host test 1.1.1.1"***
-testmode	Test Mode. Build and display the commands that would have been sent to the devices but DO NOT actually connect to the devices and send the commands. Useful for confirming your variable substitutions are correct. List of commands are also written to the file $SCRIPT_HOME/cisco_config_cmds_sample_cmds.txt". (**OPTIONAL**)
-wm	Saves configuration file to NVRAM after making the configuration changes. This option performs a "write mememory" on the router. (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The configuration command file (-cf <filename>) should contain a list of configuration commands that will be sent to each router. This is a plain text file that must contain one command per line. **Lines that begin with a "#" are considered comments and will not be sent to the router as a command.** Th **SLEEP** command is a special command that will **NOT** actually be sent to the router. If desired, this is a method to introduce a delay between configuration commands. The syntax is the **SLEEP** <***seconds***> (all capitals). Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The *SLEEP* command **MUST** be entered in all **CAPITAL** letters otherwise it will be interpreted as a command to send to the router. Note, this is not typically needed but may be useful if you would like to watch the

script as it is running to visually inspect the output of a particular command.

This script also has the option to run in ***Test Mode*** using the -testmode option defined above. With this option, the exact commands that would have been sent to the routers are written to a file (stored in the SCRIPT_HOME directory). The script does not actually login to any devices or send any commands. It is recommended this feature be used when using LOOPING and COUNTER variables (see below)

By default, the configuration commands will not be saved to NVRAM. Use the ***-wm*** option to save the configuration changes.

This program also has the option to be run in ***Safe*** and ***SuperSafe*** mode which should be considered when running scripts in production environments. If entering a configuration command results in an error on the router and the script is running in ***Safe*** or ***SuperSafe*** mode, the config will not be saved to NVRAM even if the -wm option is applied. If the script is NOT running in ***Safe*** or ***SuperSafe*** mode, and the -wm option is applied, the config WILL be saved to NVRAM even if there are one or more commands that caused configuration errors.

The following special commands are supported by this script.


(***Note, output of the commands listed below will NOT be displayed in the contents of individual output files when using the -dir option***)

Command	Description
SLEEP <seconds>	The SLEEP command is not a command that will be sent to the router. If desired, this is a method to introduce a delay between commands. Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The SLEEP command must be entered in all CAPITAL letters otherwise it will be interpreted as a command to send to the router. It must also start at the beginning of the line (no leading spaces or tabs). Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command. ***CASE SENSITIVE***
LOOPSTART <num> or LOOPEND	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will repeated are the commands between the LOOPSTART and LOOPEND commands. See Command Looping for more information. ***CASE SENSITIVE***
COUNTERVAR <variable> <start_num> <increment by>	Define a counter variable. Provides a method to define a variable in the command file; it must be used with the LOOPING feature. The variable must be a single alphabetical character. "start_num" is the number that the counter will start at. It will increment by the <increment by> number each time through the LOOP. See Counter Variables for more information. ***CASE SENSITIVE***

Sample Command 1: The following command will send the configuration commands listed in file ***snmp_cmds.cmds*** to the routers listed in the file ***rtrs.rt.*** After the configuration commands are entered, the config will be saved to NVRAM because of the ***-wm*** option. If there are any errors while sending a particular configuration command, the script will abort all remaining commands to that router and continue on to the next router because of the ***-safe*** option (SAFE Mode). The detailed trace-log will be saved to the file ***snmp_log*** instead of the default file name of ***cisco_config_cmds.log*** (notice this argument was listed first on the command line). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
cisco_config_cmds -log snmp_log -pw logins.txt -rf rtrs.rt -cf snmp_cmds.cmds -wm -safe
```

Sample Command 2: The following script will shutdown interface Gi 0/18 on switch 192.168.0.10. The output log will be saved to an automatically generated unique filename because of the -ulog option. The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option). At the first occurrence of an error, the script will abort because of the -safe option. If there are no errors, the config will be saved to NVRAM because of the -wm option.

```
cisco_config_cmds -ulog -pw logins.txt -ipaddr 192.168.0.10 -cmd "int Gi 0/18" -cmd "shutdown"
```

5.2.4 Global Router Configuration Tool (config_config_cmds_rcf)

Global Router Configuration Tool (cisco_config_cmds_rcf)

This program is very similar to the cisco_config_cmds script above except it is designed to send a **different** set of **configuration** commands to each Cisco IOS device. Do not use this script to send exec level commands (e.g., show commands). The specific configuration commands, and the IOS devices these commands are sent to, are defined in a text file that you create. The format of the text file is described below. This file containing configuration commands MUST NOT include the commands needed to enter and exit configuration mode on the router. The script will automatically handle that. The commands are sent in sequential order from top to bottom of the file.

Program Name: cisco_config_cmds_rcf

Script Argument	Description
-rcf <filename>	Short for **R**outer and **C**ommand **F**ile. File which contains a List of routers or IP Address along with a file name which contains the list of configuration commands to send to each router (**REQUIRED**)
-testmode	Test Mode. Build and display the commands that would have been sent to the devices but DO NOT actually connect to the devices and send the commands. Useful for confirming command looping and variable substitutions are correct or for just double checking the commands that will be sent . List of commands are also written to the file $SCRIPT_HOME/<script_name>_sample_cmds.txt". (**OPTIONAL**)
-wm	Saves configuration file to NVRAM after making the configuration changes. This option performs a “write mememory” on the router. (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)

-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

Below shows and example of an rcf file. This is not a TCL formatted file, it is just a text file. Lines that begin with a # are comments. The format of the file is rtrname:filename. One entry per line. The format for the file that contains the configuration commands for a particular router is identical to the "cisco_config_cmds" script.

```
# Sample template file for cisco_config_cmds_rcf script
# Lines that begind with a # are comments
# Format of file is router:filename
# Where file name contains a list of configuration
# commands that will only be sent to that router

nyrtr1:nyrtr1.cmds
10.1.1.1:cmd_file1.cmds
```

Figure 2 Sample rcf file

```
# List of configuration commands to send
# to router: nyrtr1

interface Ethernet 1/0
 ip route-cache cef
```

Figure 3 Contents for file nyrtr1.cmds

```
# List of configuration commands to send
# to router: 10.1.1.1

interface Ethernet 1/2
 ip route-cache cef
interface Ethernet 1/3
 shutdown
```

Figure 4 Contents for file cmd_file1.cmds

By default, the configuration commands will not be saved to NVRAM. Use the ***-wm*** option to save the configuration changes.

This program also has the option to be run in ***Safe*** and ***SuperSafe*** mode which should be considered when running scripts in production environments. If entering a configuration command results in an error on the router and the script is running in ***Safe*** or ***SuperSafe*** mode, the config will not be saved to NVRAM even if the -wm option is applied. If the script is NOT running in ***Safe*** or ***SuperSafe*** mode, and the -wm option is applied, the config WILL be saved to NVRAM even if there are one or more commands that caused configuration errors.

Sample Command: The following command will send configuration commands to the routers listed in file rtr_mod_100104.rcf. The commands that are sent to each router may be different and are also

defined by a filename listed in the file rtr_mod_100104.rcf. After the configuration commands are entered, the config will be saved to NVRAM because of the ***-wm*** option. If there are any errors while sending a particular configuration command, the script will abort the script because of the ***-ssafe*** option (SAFE Mode). The detailed trace-log will be saved to the file ***cisco_config_cmds_rcf.log*** because the -log option is not being used. The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
cisco_config_cmds_rcf -pw logins.txt -rcf rtr_mod_100104.rcf -wm -ssafe
```

5.2.4.1 Template File

Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\templates\cisco_config_cmds_rcf_template.txt
Linux: /usr/local/net-sense/templates/cisco_config_cmds_rcf_template.txt

```
# Sample template file for cisco_config_cmds_rcf script
# Lines that begin with a # are comments
# Format of file is router:filename
# Where filename contains a list of configuration
# commands that will only be sent to that router

nyrtr1:nyrtr1.cmds
10.1.1.1:cmd_file1.cmds
```

5.2.5 Cisco Password Changer (cisco_passwd_change)

The password changer script changes passwords on Cisco Routers. This script has the capability to change the following types of passwords on Cisco Routers:

- Secret
- Enable
- Vty
- Console
- Auxilary

Program Name: cisco_passwd_change

Script Argument	Description
-rf <filename>	List of routers or IP Address to run script against (**REQUIRED**)
-sf <filename>	Input variable file which tells the program which passwords to change (e.g. secret, vty, console,etc.). (**REQUIRED**)
-wm	Saves configuration file to NVRAM after changing the passwords. This option performs a "write memory" on the router. (**OPTIONAL**)
-nohide	Do Not hide the new password in the log files. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while changing passwords on any router (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename

	automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The actual passwords that are changed are controlled by the variables in the input file (-sf <filename>). A sample input file is provided (cisco_passwd_change_template.txt). If the end-user installation instructions were followed (Section 3.2.2), this sample template file should be in the same directory where you run the scripts from. (Note, for MS Windows, the installation utility automatically copies this template file to the C:\Program Files\net-sense\userdata directory.) The table below describes the variables in that file:

Variable	Description
CHANGE_AUX_PASSWD	This variable must be set to YES or NO. (Case Sensitive and must be CAPITAL LETTERS). If the auxiliary password should be changed, set this value to YES, else set it to NO.
CHANGE_CONSOLE_PASSWD	This variable must be set to YES or NO. (Case Sensitive and must be CAPITAL LETTERS). If the console password should be changed, set this value to YES, else set it to NO.
CHANGE_VTY_PASSWD	This variable must be set to YES or NO. (Case Sensitive and must be CAPITAL LETTERS). If the VTY password should be changed, set this value to YES, else set it to NO.
CHANGE_ENABLE_PASSWD	This variable must be set to YES or NO. (Case Sensitive and must be CAPITAL LETTERS). If the enable password should be changed, set this value to YES, else set it to NO.
CHANGE_SECRET_PASSWD	This variable must be set to YES or NO. (Case Sensitive and must be CAPITAL LETTERS). If the secret password should be changed, set this value to YES, else set it to NO.
SAME_ACCESS_PASSWORDS	This variable must be set to YES or NO. (Case Sensitive and must be CAPITAL LETTERS). In many networks, the console, auxiliary, and vty passwords are set to the same value. If this is the case, set this variable to YES, else set it to NO.
VTY_START VTY_END	These two variables define which vty terminals to change the password on. More specifically, they define the start and end values for the line vty command. For example, if VTY_START = 0 and VTY_END=4. then the command "line vty 0 4" would be entered right before changing the vty passwords.

When the script is run, it will prompt the user to enter the new passwords that will be configured on the router. (**WARNING**: A "**tab"** is not a valid character for a password). The new passwords will not be echoed to the screen. However, you will be prompted to view them if you wish before the script runs. Also, the new passwords will be visible as the script is running and actually entering them in the router. This is the normal behavior when changing passwords on the router. If you do not want to see the script echoing the commands that actually change the passwords, set the ***log_user*** variable from **1** to **0** in the ***setup.var*** file.

If any of the password configuration commands fail, for what ever reason, the remaining password

commands will not be sent to the router. By default, the script will move on to the next router in the list; unless the ***SuperSafe*** option is used, in which case, the script will completely abort.

Also, if you set the enable password equal to the secret password, or vice-versa, the router will respond with the following message:

"The enable password you have chosen is the same as your enable secret.
This is not recommended. Re-enter the enable password."

The script considers this to be an error and will not continue with additional password changes to that router. To avoid this scenario, do what Cisco recommends; don't set the enable and secret passwords to the same values!

```
[allan@linux-1 tmp]$ cisco_passwd_change -pw logins.txt -sf cisco_passwd_change_templ

*****************************************************
* For more information about Script Automation
* or support issues, contact Technical Support
* E-mail: support@net-sense.com
*****************************************************

 You have chosen to use the same password for your
 vty, aux, and console password.  You will only be
 prompted once for all of these passwords
 Please enter the new password that will be configured on the router

Please re-enter the new password

 ----- New Enable Password -----
Please enter the new password that will be configured on the router

Please re-enter the new password
 ----- New Secret Password -----
Please enter the new password that will be configured on the router

Please re-enter the new password

The following shows which passwords will be changed

 Change Auxilary Password:      YES
 Change Console Password:       YES
 Change Vty Password:           YES
 Change Enable Password:        YES
 Change Secret Password:        YES

 Do you want to see the values of these new
 passwords before the script runs? (yes/no)? yes

 New Auxilary Password: abc
 New Console Password:  abc
 New Vty Password:      abc
 New Enable Password:   def
 New Secret Password:   ghi

 Do you wish to continue with the password
 change script (yes/no)? yes
```

Sample Command: The following command will change the passwords for the routers listed in the file ***rtrs.rt***. Only the **secret** password on the router will be changed because that is what's set in the password input file ***passwd_setup.txt***. After the password is changed, the config will be saved to NVRAM because of the ***-wm*** option. If there are any configuration errors while changing the router password, the script will immediately terminate because of the ***-ssafe*** option (SuperSafe Mode). The script WILL prompt the user for passwords because the ***-pw*** option is not being used.

```
cisco_passwd_change -rf rtrs.rt -sf passwd_setup.txt -wm -ssafe
```

5.2.5.1 Template File

Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\templates\cisco_passwd_change_template.txt
Linux: /usr/local/net-sense/templates/cisco_passwd_change_template.txt

```
# Copyright 2002 NetSense Inc.
# www.net-sense.com
# Author: Allan Silverstein

################################################
# Template file for changing Cisco Passwords
# Set each of the variables below to YES or NO.
# YOU MUST USE CAPITAL LETTERS!!!
################################################
set CHANGE_AUX_PASSWD YES
set CHANGE_CONSOLE_PASSWD YES
set CHANGE_VTY_PASSWD YES
set CHANGE_ENABLE_PASSWD YES
set CHANGE_SECRET_PASSWD YES

#############################################################
# If the vty password, console password, and aux password
# will be the same, then this value should be YES.  Otherwise
# set it to NO
# YOU MUST USE CAPITAL LETTERS!!!
#############################################################
set SAME_ACCESS_PASSWORDS YES

#####################################################
# Enter the VTY Line Numbers when changing the VTY
# Password.  The default on Cisco is lines 0 through 4
# For example, if the commands to change the vty password
# are
# line vty 0 4
# password abcd
# then VTY_START would be 0 and VTY_END would be 4
#####################################################
set VTY_START 0
set VTY_END 4
```

5.2.6 Global Router Banner Configurator (cisco_config_banner)

This program is used to send banner type commands to Cisco IOS devices. The banner commands are unique to Cisco IOS in that they allow the user to enter and exit a type of editing mode to enter the message body of the banner. The unique operation of the editing mode and the use of a character delimiter, to enter/exit editing mode, is the reason why a separate script is required to send banner type commands. All of the banner type commands should be supported. Some examples are:

- banner login
- banner motd
- banner exec

The exact banner command, the character delimiter, and the message body are all defined in a text file specified through the -cf command line argument. This file requires a special format that must be followed. The template file (cisco_config_banner_template.txt) is located in the templates directory.

Program Name: cisco_config_banner

Script Argument	Description
-rf <filename>	List of routers or IP Address to run script against (**REQUIRED**)
-cf <filename>	File which contains information for sending the banner command. Lines that begin with a "#" are considered comments and will not be sent to the router; unless they are contained between the MESSAGE_START and MESSAGE_END key words. (**REQUIRED**)
-wm	Saves configuration file to NVRAM after making the configuration changes. This option performs a "write mememory" on the router. (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The configuration command file (-cf <filename>) is a special format file that must be followed. The template file (cisco_config_banner_template.txt) is located in the templates directory and a copy of this file should be used to create your own file when using this script. There are special key words in this file as follows:

Key Word	Description
CMD	Defines the exact syntax of the banner command minus the character delimiter. Example: CMD=banner motd
DELIMITER_CHAR	Delimiting character which tells IOS when to enter and exit banner editing mode. This must be one single character which must not be included in your message body.

	Example: DELIMITER_CHAR=^
MESSAGE_START MESSAGE_END	All lines between MESSAGE_START and MESSAGE_END will be included in the message body. Example: MESSAGE_BODY ********** Authorized Access Only ********** ********** All others Keep Out ********** MESSAGE_END

By default, the banner configuration will not be saved to NVRAM. Use the ***–wm*** option to save the configuration changes.

This program also has the option to be run in ***SuperSafe*** mode which should be considered when running scripts in production environments. If entering the banner command results in an error on the router and the script is running in ***Safe*** or ***SuperSafe*** mode, the config will not be saved to NVRAM even if the –wm option is applied. If the script is NOT running in ***Safe*** or ***SuperSafe*** mode, and the –wm option is applied, the config WILL be saved to NVRAM even if there are one or more commands that caused configuration errors.

Sample Command: The following command will send the banner message defined in listed in file ***new_motd_banner.txt*** to the routers listed in the file ***rtrs.rt.*** After the configuration commands are entered, the config will be saved to NVRAM because of the ***–wm*** option. If there are any errors while sending the banner to any of the routers, the script will abort all remaining commands to that router and abort the script because of the ***–ssafe*** option (Super SAFE Mode). The detailed trace-log will be saved to a unique filename (found in the summary.log file). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
cisco_config_banner -ulog -pw logins.txt -rf rtrs.rt -cf new_motd_banner.txt -wm -ssafe
```

5.2.6.1 Template File

Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\templates\cisco_config_banner_template.txt
Linux: /usr/local/net-sense/templates/cisco_config_banner_template.txt

```
##############################
# CMD must be one of the following
# ny-rtr1(config)#banner ?
#  LINE            c banner-text c, where 'c' is a delimiting character
#  config-save     Set message for saving configuration
#  exec            Set EXEC process creation banner
#  incoming        Set incoming terminal line banner
#  login           Set login banner
#  motd            Set Message of the Day banner
#  prompt-timeout  Set Message for login authentication timeout
#  slip-ppp        Set Message for SLIP/PPP
#
# ny-rtr1(config)#banner motd ?
#  LINE  c banner-text c, where 'c' is a delimiting character
#
```

```
# ny-rtr1(config)#banner motd
################################
CMD=banner motd

##########################################
# Delimter Character.  This is the character
# that will enter/exit editing mode.
##########################################
DELIMITER_CHAR=^

####################################
# Everthing between MESSAGE_START
# and MESSAGE_END will be in the message
# including blank lines
####################################
MESSAGE_START

*******  Anyone not authorized to login into **********
*******  this system will be prosecuted to the ********
*******  fullest extent of the law. ******************


MESSAGE_END
```

5.2.7 Report Scripts

Enter topic text here.

5.2.7.1 Inventory Report (cisco_inventory_rpt)

The Cisco Inventory Report Program provides an Inventory of all components in your network. Cisco IOS version report for Cisco Routers. The script cycles through a list of routers/switches and performs a ***show inventory***. The relevant data is extracted from the *show inventory* command and saved to a report file. The data included in this report is:

- Router Name/ IP Address
- ***Name*** of the specific component on the router/switch
- ***Description*** of the specific component on the router/switch
- ***Part ID*** of the specific component on the router/switch
- ***Version ID*** of the specific component on the router/switch
- ***Serial Number*** of the specific component on the router/switch

Program Name: cisco_inventory_rpt

Script Argument	Description
-rf <filename>	List of routers or IP Address to run script against (**REQUIRED**)
-of <filename>	Output File. File where inventory report will be written to. (**REQUIRED**)
-ual (1 or 2)	User Access Level. Tells the script what access-level (1st or 2nd) to log into the router before performing the show inventory command. By default the script will only go into 1st level access. (**OPTIONAL**)
-sort (dn | devicename) (pn | partname) (d | descr) (p | partid) (v | versionid) (s | serialnum)	Sort the data based on one of the Columns (**OPTIONAL**) dn - Device Name p - Part Name d - Description of Part p - Pard ID

	v - Version ID of Part s - Serial Number of Part
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

Sample Command: The following command runs the cisco_inventory_rpt program against the routers listed in the file ***rtrs.txt***. The script will not prompt the user for passwords a password file is being used with the **-pw** option. The resulting report will be written to the file ***inventory_report.txt***.

```
cisco_inventory_rpt -pw logins.txt –rf rtrs.txt –of inventory_report.txt
```

Router	NAME	DESCR	PID	VID	Serial Num
nyrtr-1	"Chassis"	"Cisco 7206VXR, 6-slot chassis"	CISCO7206VXR	---	74530813
nyrtr-1	"NPE-G1 0"	"Cisco 7200 Series Network Processing Engine NPE-G1"	NPE-G1	---	66342519
nyrtr-1	"disk2"	"256MB Compact Flash Disk for NPE-G1"	MEM-NPE-G1-FLD128	---	---
nyrtr-1	"module 1"	"VAM2"	SA-VAM2	V01	JAB87522765
nyrtr-1	"Power Supply 1"	"Cisco 7200 AC Power Supply"	PWR-7200-AC	---	---
sjrtr-3	"Chassis"	"1841 chassis, Hw Serial#: 965613917, Hw Revision: 5.0"	1841	5.0	FTK8635P487
sjrtr-3	"C1841 Motherboard with 2 Fast Ethernet"	"C1841 Motherboard with 2 Fast Ethernet"	C1841 Motherboard with 2 Fast Ethernet	5.0	FTK8635P487
sjrtr-3	"WIC/VIC 0"	"WAN Interface Card - Serial (1T)"	WAN Interface Card - Serial (1T)	1.0	13745564
ny-sw-3	1	"WS-C3560G-48TS"	WS-C3560G-48TS-S	02	W6B0372J0V2
njbb-3	"CISCO7609"	"Cisco Systems Cisco 7600 9-slot Chassis System"	CISCO7609	---	FOX0751001V
njbb-3	"WS-C6K-VTT 1"	"VTT FRU 1"	WS-C6K-VTT	---	NWG07490996
njbb-3	"CLK-7600 1"	"OSR-7600 Clock FRU 1"	CLK-7600	---	NWG10350306
njbb-3	"CLK-7600 2"	"OSR-7600 Clock FRU 2"	CLK-7600	---	NWG10350306
njbb-3	"module 3"	"WS-X6748-GE-TX CEF720 48 port 10/100/1000mb Ethernet Rev. 2.4"	WS-X6748-GE-TX	V02	SAL4021Q5BT
njbb-3	"switching engine sub-module of 3"	"WS-F6700-CFC Centralized Forwarding Card Rev. 2.1"	WS-F6700-CFC	V01	SAL85672BNR
njbb-3	"module 5"	"RSP720-3C-GE 2 ports Route Switch Processor 720 Rev. 5.2"	RSP720-3C-GE	V03	JAE6421VERH

njbb-3	"msfc sub-module of 5"	"7600-MSFC4 C7600 MSFC4 Daughterboard Rev. 1.1"	7600-MSFC4	0	JAE9934TM2J
njbb-3	"switching engine sub-module of 5"	"7600-PFC3C Policy Feature Card 3 Rev. 1.1"	7600-PFC3C	0	JAE3456G0D9
njbb-3	"module 6"	"RSP720-3C-GE 2 ports Route Switch Processor 720 Rev. 5.2"	RSP720-3C-GE	V03	JAE4249YHTE
njbb-3	"msfc sub-module of 6"	"7600-MSFC4 C7600 MSFC4 Daughterboard Rev. 1.1"	7600-MSFC4	0	JAE4249YNH5
njbb-3	"switching engine sub-module of 6"	"7600-PFC3C Policy Feature Card 3 Rev. 1.1"	7600-PFC3C	0	JAE4249TTPE
njbb-3	"FAN-MOD-09 1"	"Vertical 9-slot Fan FRU 1"	FAN-MOD-09	---	HYY17457345
njbb-3	"FAN-MOD-09 2"	"Vertical 9-slot Fan FRU 2"	FAN-MOD-09	---	HYY17457332
njbb-3	"PS 2 WS-CAC-3000W"	"AC power supply, 3000 watt 2"	WS-CAC-3000W	V01	RTY15678AYT

5.2.7.2 IOS Report (ios_report)

The IOS Version Report Program provides a Cisco IOS version report for Cisco Routers. The script cycles through a list of routers and performs a "show version". The relevant data is extracted from the show version command and saved to a report file. The data included in this report is:

- Router Name/ IP Address
- Platform Type
- Memory (Mbytes)
- IOS Version
- IOS Image Filename
- Name of Flash Directory where image is located
- Total Flash (Bytes)
- Free Flash (Bytes)

Program Name: ios_report

Script Argument	Description
-rf <filename>	List of routers or IP Address to run script against (**REQUIRED**)
-of <filename>	Output File. File where version report will be written to. The location of the file will be the SCRIPT_HOME directory unless a filename with an absolute path is used. (**REQUIRED**)
-sort (n | name) (p | platform) (m | memory) (v | version) (i | imagename) (d | flashdirectory) (t | totalflash) (f | freeflash)	Sort the data based on one of the Columns (**OPTIONAL**) n - Name of device p - Platform type m - Memory v - Version i - Image file name d - flash Directory name t - Total flash f - Free flash
-ual (1 or 2)	User Access Level. Tells the script what access-level (1^{st} or 2^{nd}) to log into the router before performing the show version command. By default the script will only go into 1^{st} level access. (**OPTIONAL**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)

-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

Sample Command: The following command runs the ios_report program against the routers listed in the file ***rtrs.txt***. The data will be sorted by Platform Type (***-sort -p***). The script will not prompt the user for passwords a password file is being used with the **-pw** option. The resulting report will be written to the output file ***report.txt*** in the ***SCRIPT_HOME*** directory.

```
ios_report -pw logins.txt -rf rtrs.txt -of report.txt -sort p
```

Sample Output Report:

```
                                Memory
Router        Platform          (MBytes)   Version      IOS Image File
---------     ---------         -------    ----------   --------------------
ny-rtr1       1841                 326     12.4(11)T    c1841-advipservicesk9-mz.124-11
ny-rtr2       2811                 218     12.4(11)T    c2800nm-advipservicesk9-mz.124-
ny-rtr3       7206VXR              224     12.4(11)T    c7200-advipservicesk9-mz.124-11
10.12.1.1     ASR1004             4131     15.1(3)S1    asr1000rp2-adventerprisek9.03.
mx-rtr9       CISCO2921/K9        2475     15.1(3)T2    c2900-universalk9-mz.SPA.151-3.
mx-rtr4       CISCO3925-CHASSIS    970     15.1(3)T2    c3900-universalk9-mz.SPA.151-3.
```

5.2.8 Config Backup Scripts

Enter topic text here.

5.2.8.1 Router Configuration Backup Utility (copy_to_tftp)

This script backs up the router configuration files to a TFTP server. It cycles through a list of routers performing a “copy running-config tftp:”. Note, this script is an exception in that it also works against Cisco devices running NX-OS code (e.g, Nexus switch).

NOTE: IF BACKING UP Cisco **NX-OS** DEVICES, YOU MUST USE A PASSWORD FILE (-pw option). OTHERWISE THE LOGIN TO THE NX-OS DEVICE WILL FAIL.

Program Name: copy_to_tftp

Script Argument	Description
-rf <filename>	List of routers or IP Address to run script against (**REQUIRED**)
-ipaddr <ip addr>	IP Address of TFTP Server. This does not have to be the same IP Address as the system that the script is being run from. The script can run from a Sun Solaris system but the TFTP server IP Address can be a PC. (**REQUIRED**)
-**tftproot** <tftp root directory>	This option will enable the script to “touch” the configuration filename on a UNIX TFTP server before backing up each router config. If used in combination with the –subdir option, the script will also create the directory under the ROOT TFTP directory. In order for the script to perform these operations, the script needs to know the TFTP Root directory. **In addition,**

	for this option to apply, the TFTP server ip address must be the same as the system that the script is being run from.
-subdir <directory name>	A sub-directory under the main TFTP directory that the configuration files should be written to. This is a relative path name NOT and absolute path name (**OPTIONAL**)
-autodir <date | time>	Automatically create unique directory name to save configuration files to. The choices are ***date*** or ***time***. The date option will create a new directory consisting of just the date e.g. 08012010 for Aug 1st 2010. The time option will append the time to the date: e.g. 08012010_12h36m15s The -tftproot option must be specified when using this option. The -subdir may also be used with this option if desired." The purpose of this new option is to make nightly backups of router configurations even easier. Previously, it was up to the end-user to create a unique name using only the -subdir option. If doing automated nightly backups, this required a little shell scripting (Linux/Unix) or Windows shell programming to create a unique directory name that was passed to the -subdir option. This is no longer necessary. See example below.
-vrf (VRF Name)	"VRF name. Only need to specify if backing up Nexus devices and the backup will be going through a vrf other than the the global route table. If using the separate management port on the Nexus device, then the vrf name should be ***management***. Note, only one vrf name can be specified on the command line. If multiple NXOS devices are being backed up and they use different vrf names, then this script must be run several times using a different vrf name each time. (**OPTIONAL**)
-wm	Saves configuration file to NVRAM after copying file to TFTP server. If tftp backup failed, the write mem will not be performed. This option performs a "write memory" on the router. (**OPTIONAL**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-urfn	Use Route File Name. Use ip address/name in route file for config file name instead of router hostname.
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The configuration filename that is saved to the TFTP server is <routername-confg>. This is the default file name that the router chooses. There is also an option to have the configuration filename be the name/ip_address that is specified in the router file (i.e. –rf <filename>). This is done using the ***–urfn*** option. For example, if one of the lines in the router file is 10.10.1.1, then the config file name would be *10.10.1.1-confg*. The directory that the configuration files are saved to, can also be changed using the "-subdir" option. By default the script will attempt to save the configuration file to the root TFTP directory. By using the ***–subdir*** option, you can have multiple configuration files for the same router saved under different directories. Multiple levels below the root tftp directory can be specified (See the 2nd example below)

Also when using a UNIX TFTP server, the file name for the router configuration files must be created before the script runs. This is typical behavior for UNIX TFTP servers. By using the ***–tftproot*** option the script will create (i.e. UNIX ***touch*** command) the configuration files on the TFTP server for you so you don't have to do it manually. Using the ***–subdir*** and ***–tftproot*** options together enables the script to create a new directory, under the tftp root directory, and "*touch"* the config files in that directory. **When using the *–tftproot* option, the TFTP server ip address must be the same as the system that the script is being run from.** (Note for Linux, A TFTP sub-directory created by the script (***-subdir***) will create the directory with the permissions of "read-write-execute" for "user-group-other" respectively [or 777]. Router configuration files that are *touched* by the script, will be created with "read-write" permissions for "user-group-other" respectively [or 666]).

For TFTP Server applications on Microsoft Windows Platforms, the filename usually does not need to pre-exist on the TFTP Server before the router attempts to write its configuration file to the TFTP Server.

Sample Command: The following command will backup the configuration files for the routers listed in the file ***rtrs.rt***. The files will be backed up to the 10.10.1.20 TFTP Server. The configs will be stored in a sub-directory under the root TFTP directory named 02-25-03_configs. The config files will automatically be *touched* on the TFTP server. The ***–tftproot*** option also causes the script to automatically create the directory specified in the ***–subdir*** option. The script WILL prompt the user for passwords because the *-pw* option is not being used.

```
copy_to_tftp -rf rtrs.rt -ipaddr 10.10.1.20 -subdir 02-25-03_configs -tftproot /tftpboot
```

Sample Command: This example emphasizes that the script can create multilevel directories under the root tftp directory. The following command will backup the configuration files for the routers listed in the file ***rtrs.rt***. The files will be backed up to the 10.10.1.20 TFTP Server. The configs will be stored in a sub-directory under the root TFTP directory named ***abc/02-25-03_configs***. The config files will automatically be *touched* on the TFTP server. The ***–tftproot*** option also causes the script to automatically create both directories specified in the ***–subdir*** option. The script WILL prompt the user for passwords because the -pw option is not being used.

```
copy_to_tftp -rf rtrs.rt -ipaddr 10.10.1.20 -subdir abc/02-25-03_configs -tftproot /tftpboot
```

Sample Command: This example shows the use of the autodir option. The following command will backup the configuration files for the routers listed in the file ***rtrs.rt***. The files will be backed up to the 10.10.1.20 TFTP Server. Assuming the date is August 5th 2010, the configs will be stored in /tftpboot/rtr_configs/08052010. Note, the rtr_configs directory does not have to exist before the script is run. The script will not prompt the user for passwords or a password encryption key because the -pw option is used with an unencrypted password file. The -ulog opition will automatically create a unique detailed trace log file.

```
copy_to_tftp -ulog -pw logins.txt -rf rtrs.rt -ipaddr 10.10.1.20 -tftproot /tftpboot -subdir rt
```

Sample Command: Same example as above but password file is encrypted. Here, the user will not be prompted for a password encryption key because the ***-nokey*** option is specified. This command line is a good example of all that would be need to put into a Unix/Linux shell file or a windows batch file. The shell file or batch file could then easily be called to run through cron or windows scheduler.

```
copy_to_tftp -ulog -nokey -pw logins_encr.txt -rf rtrs.rt -ipaddr 10.10.1.20 -tftproot /tftpboo
```

5.2.8.2 Router Configuration Backup Utility 2 (conf_bkup_show_run)

Router Configuration Backup Utility 2 (conf_bkup_show_run)

Unlike the copy_to_tftp script, this script backs up the router configurations by telneting to the router and performing a "show run". Any extra information from the telnet session is stripped off and the file is then saved to the system where the script is run from. This script is useful when a TFTP server is not available or the router cannot reach the TFTP server (e.g. routers outside a Firewall)

Program Name: conf_bkup_show_run

Script Argument	Description
-rf <filename>	List of routers or IP Address to run script against (**REQUIRED**)

-dir <directory name>	If specified, tells script to save config file for each device to this directory. If the directory does not exist, the script will create it. The directory entered can be either an absolute directory or a relative directory. If running the script from the GUI, a relative directory is relative to the "SCRIPT_HOME" variable. See below for more information. (From the GUI see Options->Settings). (**OPTIONAL**)
-autodir <date | time>	Automatically create unique directory name to save configuration files to. The choices are ***date*** or ***time***. The date option will create a new directory consisting of just the date e.g. 08012010 for Aug 1st 2010. The time option will append the time to the date: e.g. 08012010_12h36m15s The new directory will be created under the SCRIPT_HOME directory." SCRIPT_HOME can be found in the GUI (Options->Settings) or by viewing the setup.var file. The purpose of this option is to make nightly backups of router configurations even easier. Previously, it was up to the end-user to create a unique name using only the -dir option. If doing automated nightly backups, this required a little shell scripting (Linux/Unix) or Windows shell programming to create a unique directory name that was passed to the -subdir option. This is no longer necessary. See example below.
-wm	Saves configuration file to NVRAM after copying file to server. This option performs a "write memory" (i.e., copy run startup) on the router. If extraction of router config was not successful then the "write memory" is not performed. (**OPTIONAL**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-urfn	Use Route File Name. Use ip address/name in route file for config file name instead of router hostname.
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The configuration filename that is saved is <router_name--confg.txt>. There is also an option to have the configuration filename be the name/ip_address that is specified in the router file (i.e. –rf <filename>). This is done using the ***–urfn*** option. For example, if one of the lines in the router file is 10.10.1.1, then the config file name would be *10.10.1.1--confg.txt*. The directory that the configuration files are saved to can also be changed using the "-dir" or -autodir options. If the -dir and the -autodir options are omitted, the script will save the router configuration files to the SCRIPT_HOME directory.

If the directory does not exist, the script will create it. If the -dir option is used with a relative path name (not absolute path name), the stated directory will be created under the SCRIPT_HOME directory.

Sample Command: The following command will backup the configuration files for the routers listed in the file ***rtrs.rt***. The script will create the directory "02-25-03_configs" under the SCRIPT_HOME directory and the configuration files will be stored there. It will also save the configuration to NVRAM using the "write memory" command. The script WILL prompt the user for passwords because the *-pw* option is not being used.

```
conf_bkup_show_run -rf rtrs.rt -dir 02-25-03_configs -wm
```

5.2.9 PINGER: Verify Any-to-Any IP Connectivity (pinger)

The pinger program is a utility that performs pings from a Cisco Router to any IP Address. The script will cycle through a list of routers and perform pings from each router in a list. In addition, the pings can be sent with different source IP addresses. Note, the source IP address must be a valid interface IP Address on the router that is in an UP/UP condition. If the interface is not in an UP/UP condition, the script will not perform the pings using that source IP Address. The special keyword "default" can also be specified as a source IP address. In this case the router software will use the interface IP address, where the ping packet exits, for the source IP address. This script supports both IPv4 and IPv6 addressing.

Program Name: pinger
Template File: *pinger_template_rev4.txt*

Script Argument	Description
-sf <filename>	Input variable file which tells the program which routers the pings will be performed from, the source IP addresses to use for the pings and the addresses to ping. ***The sample template filename is pinger_template.txt*** (**REQUIRED**)
-pinfo	Print Info. This option tells the script to print out (and log) informational messages as to source/destination pairs for pings that were successful. Be default, only messages for pings that have failed are written to the displays and log file.
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The ping source/destination pairs that will be sent are controlled by the variables in the input file (-sf <filename>). This file is in TCL format and contains two TCL list variables; TOPO_LIST and PING_LIST. The TOPO_LIST variable lists the routers along with the source IP Addresses that will be used for the pings. The PING_LIST variable contains the list of IP Addresses that will be pinged.

Below shows a sample entry for the TOPO_LIST variable (for Revision 4 input file):

```
lappend TOPO_LIST [list nj1 "no" "global" "default" "64.145.25.3"]
```

The first item in the list "nj1" is the router that the pings will be performed from. The script will actually telnet/ssh into this router. This must be an IP Address or a name that can be resolved through DNS. (If the -pw <*password_file*> option is used, then this IP Address/Name must also be defined in the *password_file*. Note, they must match exactly [case sensitive]). The second item specifies whether or not the ping is being done in a VRF. The third item is the vrf name. If vrfs are not being used, then the

third item must be the string "global". The forth and fifth arguments are the source IP addresses that will be used for the pings.

Below shows a sample entry for the PING_LIST variable:

```
lappend PING_LIST [list "200.32.127.138" "ny2 l0"]
lappend PING_LIST [list "10.10.1.1" "global" "nj1 Lan"]
```

The first item in the list “10.10.1.1” is the IP Address that will be pinged. The second item in the list is the vrf name or "global" if address is not in vrf. If this string does not match the vrf name from TOPO_LIST, then the ping to this address will be skipped. The third item in the list, “nj1 Lan” is an informational comment about that IP Address. This comment is also printed out in the error log if a ping fails to this IP Address.

Below is a sample file (Rev 4) that will be used to describe the flow of the program in relation to the input file. Note, the line that starts with a “#” will not be used.

```
lappend TOPO_LIST [list ny2 "no" "global" "10.134.132.4" "200.32.127.138"
"default"]
lappend TOPO_LIST [list cny3 "no" "global" "172.33.1.1" "10.134.132.1"]
lappend TOPO_LIST [list ca1 "no" global" "10.31.240.253" "200.42.59.16"
"10.31.29.1"

lappend PING_LIST [list "10.145.25.1" "global" "????"]
#lappend PING_LIST [list "10.145.25.3" "global" "nj1 eth"]
lappend PING_LIST [list "10.126.4.102" "global" "nj1 eth"]
lappend PING_LIST [list "10.126.5.103" "global" "nj2 eth"]
lappend PING_LIST [list "10.126.6.252" "global" "cnj5 eth"]
```

1. Telnet to router “ny2”
2. Ping each of the IP addresses in the PING_LIST using 10.134.132.4 as the source IP address.
3. Ping each of the IP addresses in the PING_LIST using 200.32.127.138 as the source IP address.
4. Ping each of the IP addresses in the PING_LIST using the source IP Address that the router chooses to insert. More specifically, this is the IP Address of the interface that the packet leaves from.
5. Telnet to router “cny3”
6. Ping each of the IP addresses in the PING_LIST using 172.33.1.1 as the source IP address.
7. Ping each of the IP addresses in the PING_LIST using 10.134.132.1 as the source IP address.
8. Telnet to router “ca1”
9. Ping each of the IP addresses in the PING_LIST using 10.31.240.253 as the source IP address.
10. Ping each of the IP addresses in the PING_LIST using 200.42.59.16 as the source IP address.
11. Ping each of the IP addresses in the PING_LIST using 10.31.29.1 as the source IP address.

There is one additional variable in the input file to control the speed at which successive pings are generated from any one router. The variable is PING_DELAY_INTERVAL and is defined in milliseconds. If a script input file is setup to have a router generate pings to 100 different destinations, a delay (equal to the PING_DELAY_INTERVAL) will be inserted before each new ping destination. This variable was added because results have shown that low-end routers may experience a CPU spike because the script is cycling through the pings extremely fast resulting in a lot of telnet based traffic. The exact processes that cause the CPU to spike are the vty exec (i.e. telnet session) and IP

Traffic. Putting in a 250ms delay will limit the spike, caused by the script, to between 20-25%. A 500ms delay will limit the spike to about 10%. (Note, these results are based on a Cisco 1700 series router and do not include CPU utilization for other processes on the router.) For higher end routers this should be less of an issue. Also, if each router is sending to a low number of ping destinations (10 or under), this should be non-issue.

Several sample input files are provide here.

Sample Command: The following command will run the pinger utility using the information contained in the file ***ping_southeast.txt***. The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
pinger -pw logins.txt -sf ping_southeast.txt
```

5.2.9.1 VRF Support and Template Files

The Net-Sense Pinger script also supports the use of VRFs and IPv6. There are multiple template files and depending on the features you need you can choose the appropriate template file. Make a copy of one of the files below and change IP addresses to your network addresses. The file you create will be used as the filename to the -sf command line argument (-sf <filename>)

Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\templates
Linux: /usr/local/net-sense/templates/cisco_config_banner_template.txt

Template File Name	Description
pinger_template_rev1_no_vrfs.txt	IPv4 support: YES IPv6 support: NO Supports pings without using VRFs: YES Supports pings using VRFs: NO
pinger_template_rev1_vrfs.txt	IPv4 support: YES IPv6 support: NO Supports pings without using VRFs: NO Supports pings using VRFs: YES
pinger_template_rev3.txt	IPv4 support: YES IPv6 support: NO Supports pings without using VRFs: YES Supports pings using VRFs: YES
pinger_template_rev4.txt	IPv4 support: YES IPv6 support: YES Supports pings without using VRFs: YES Supports pings using VRFs: YES

5.2.9.2 Pinger Skip List and Diagnostic Features

Version 4.3.2 of The Automater introduced 2 new features to the Pinger script:

- Ability to skip pings to specific source/destination pairs
- Ability to issue commands to a router in the event a ping fails.

As stated in the pinger script description, the Pinger script will systematically cycle through a “matrix” of source/destination ping pairs. However, there may be cases where you know certain source/destination pairs will intentionally fail and you don’t want the script to log a “false” error message. This feature is enabled by adding a new variable, SKIP_LIST, to the input file. See the file ***pinger_template.txt*** and ***pinger_vrf_template.txt*** for more details and examples. If you do not want this feature, just keep the ***SKIP_LIST*** variable commented out (i.e. put a pound sign at the beginning of the line)

A second useful feature added to the Pinger script is the ability to issue some diagnositic commands (e.g. *show commands*) immediately after a ping fails. This is useful because sometimes the script catches problems that happen infrequently and intermittently. Thus, the script can issue the troubleshooting commands that you would have entered if you were manually performing the pings yourself. After the script finishes you can view the detailed trace log and examine the output of those commands. This feature is enabled by adding the new variable, ***DIAG_CMD_LIST***, to the input file (-sf filename). See the file ***pinger_template.txt*** and ***pinger_vrf_template.txt*** for more details and examples. If you do not want this feature, just keep the ***DIAG_CMD_LIST*** variable commented out (i.e. put a pound sign at the beginning of the line).

Version 4.4 introduced the ability to issue specific commands before the actual pings are sent. This feature is enabled by adding the new variable,***PRE_PING_CMD_LIST***, to the input file (-sf filename). See the file ***pinger_template.txt*** and ***pinger_vrf_template.txt*** for more details and examples. If you do not want this feature, just keep the ***PRE_PING_CMD_LIST*** variable commented out (i.e. put a pound sign at the beginning of the line).

5.2.10 Tracer: Verify Packet Paths Through Network (tracer)

The tracer program is a utility that performs traceroutes from a Cisco Router and compares the recorded path with the expected path. The expected path is predefined through an input file. The script will cycle through a list of routers and perform traceroutes to predefined destinations. If the recorded path does not match the expected path an error message is written to the summary log file. An error message is also logged if the traceroute fails. Due to new newly added features in for version 4.3.2, the new tracer input file is slightly different although the tracer script is backward compatible with the previous tracer input file.

New Featutes:

- Support for multiple paths to the same destination
- Ability to specify the source IP address used for the trace route
- Ability to issue user specified commands before traceroute is performed

Program Name: `tracer`
Template Files: `tracer_template.txt, tracer_vrf_template.txt`

Script Argument	Description
-nopath	Perform the traceroutes but do not compare the actual traceroute path against the expected traceroute path. This may be useful if you would just like to perform the traceroutes and record the output. If the actual path is not equal to the expected path, an error message will NOT be logged.
-sf <filename>	Input variable file which tells the program which routers the pings will be performed from, the source IP addresses to use for the pings and the addresses to ping. ***The sample template filename is pinger_template.txt*** (**REQUIRED**)

-ual (1 or 2)	User Access Level. Tells the script what access-level (1st or 2nd) to log into the router before sending the commands. By default the script will only go into 1st level access. (**OPTIONAL but most likely necessary**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The traceroutes that will be performed are controlled by the variables in the input file (-sf <filename>). This file is in TCL format and contains TCL list variables. The list variable ***RTR_LIST*** defines the list of routers that the script will telnet to, as well as the source IP address that should be used for the traceroutes. It also includes another variable, ***DEST_LIST_x***, that defines the traceroutes that will be performed while telneted into each router. The value of ***x***, in ***DEST_LIST_x***, is a numerical value that must be different for each router.

Below shows a sample entry for the ***RTR_LIST*** variable (non VRF scenario). The first item in the list "ny1" is the router that the traceroutes will be performed from. The script will actually telnet/ssh into this router. This must be an IP Address or a name that can be resolved through DNS. If the -pw < *password_file*> option is used, then this IP Address/Name must also be defined in the *password_file*. (Note, they must match exactly [case sensitive]). The second argument is another variable (***DEST_LIST_x***) that contains the associated list of traceroutes to check while in that device. The last item in the list is the source IP address to use when performing trace routes from this router. If you would like to use the default source IP address that the router would use, then just specify two double quotes with nothing between them (the sample line below shows this) or you could specify the key word ***default*** for the source IP.

```
lappend RTR_LIST [list "ny1" "DEST_LIST_1" ""]
```

Below shows a sample of ***DEST_LIST_x***. The first item in the list, "10.30.30.1" is the destination of the traceroute. The second item can have a value of ***yes*** or ***no*** and tells the script whether to compare the expected route to the actual route or to just check whether the trace route is successful. The third item, "1.1.1.1 1.1.2.1 1.1.3.1 1.1.4.1", is the actual expected trace route path. If a value of no is entered for the second item, the third item must still have a "dummy" expected trace route path or just two double quotes with nothing in between them (e.g. "")

```
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.2.1 1.1.3.1 1.1.4.1"]
```

New in version 4.3.2, the tracer script now supports traceroute paths with multiple "primary" paths due to "load balancing"/redundancy. For this scenario, all you do is add a second (or third, forth, or more!) list entry with the same exact destination but a different path. The script will then check both of these entries. Here is an example of traceroute paths to 10.30.30.1 with 3 possible different trace route paths:

```
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.2.1 1.1.3.1 1.1.4.1"]
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.100.1 1.1.150.1 1.1.4.1"]
```

```
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.200.1 1.1.250.1 1.1.4.1"]
```

Below shows the sample input file (***tracer_template.txt***) that is provided with the program and should be used as a template when creating your own input files (for non-vrf scenarios).

The template input file below instructs the script to do the following:

1. Telnet/ssh into router “br1” and perform traceroutes to the destinations listed in DEST_LIST_1.
2. Telnet/ssh into router “sj2” and perform traceroutes to the destinations listed in DEST_LIST_2
3. Telnet/ssh into router “192.168.1.40” and perform traceroutes to the destinations listed in in DEST_LIST_3

```
#####################
# DO NOT MODIFY THE
# TRACER_REV variable
#####################
set TRACER_REV 2

# Template for trace route script (tracer)

# Scenario: Normal Steady State

#############################################################################
# The list of routers to telnet to and perform a trace route
# Field Definitions:
# 1. rtr: Router to telnet to and perform trace routes
# 2. The associated trace route destinations to check on this router
# 3. Source IP Address used for traceroutes from this router.  To use the default
#    address that the router wants to use, just leave the value, between the two
#    double quotes, to nothing, or put in the key word default (see samples below)
#############################################################################
#                     rtr            dest_list        src_IP
#############################################################################
lappend RTR_LIST [list "br1"          "DEST_LIST_1"    "1.1.1.1"]
lappend RTR_LIST [list "sj2"          "DEST_LIST_2"    ""]
lappend RTR_LIST [list "192.168.1.40" "DEST_LIST_3"    "default"]

#############################################################################################
###
# Note, the yes/no field is whether or not the traceroute should be successfull.
# This is good for failure scenarios where you may want to confirm that a
# traceroute fails after a failure scenario was introduced
# The expected "trace route path list" is compared against the actual
# path when the script runs
#############################################################################################
#############
# For router: BR1
# Variables               subnet     yes/no "expected trace route path list"
#############################################################################################
#############
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.2.1 1.1.3.1 1.1.4.1"]
lappend DEST_LIST_1 [list "10.4.4.2" "yes" "1.1.1.1 1.1.6.1 1.1.7.1 1.1.8.1"]

#############################################################################################
#############
# For router: SJ2
# Variables               subnet     yes/no "trace route path list"
#############################################################################################
#############
lappend DEST_LIST_2 [list "10.30.30.1" "yes" "2.1.1.1 2.1.2.1 2.1.3.1 2.1.4.1"]
lappend DEST_LIST_2 [list "10.4.4.2" "yes" "2.1.1.1 2.1.6.1 2.1.7.1 2.1.8.1"]

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

```
# For router: 192.168.1.40 (3745)
# Variables               subnet     yes/no "trace route path list"
################################################################################################
#############
lappend DEST_LIST_3 [list "10.30.30.1" "yes" "3.1.1.1 3.1.2.1 3.1.3.1 3.1.4.1"]
lappend DEST_LIST_3 [list "10.30.30.1" "yes" "3.1.1.1 3.1.6.1 3.1.6.1 3.1.6.1"]
```

Sample Command: The following command will run the tracer utility using the information contained in the file ***tracer_northeast.txt***. The –ual option instructs the script to login into 2nd level access (i.e. privileged mode). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
tracer -pw logins.txt –sf tracer_northeast.txt –ual 2
```

5.2.10.1 VRF Support for Tracer Script

The Net-Sense Tracer script also supports the use of VRFs. This feature is available by using a different input tracer file (-sf filename) that supports the VRF information. The template file for this is ***tracer_vrf_template.txt***. The format of this file is almost identical to the tracer_template.txt file with some minor changes as defined below.

- The file contains the statement “set VRF 1”. Do NOT remove this statement
- The RTR_LIST variable contains one additional field to specify the VRF name. Below shows a VRF by the name of vpn2

```
lappend RTR_LIST [list "192.168.1.40" "vpn2" "$DEST_LIST_3"]
```

5.2.10.2 Additional Tracer Features and Diagnostics

Also new for version 4.3.2 is the ability to issue user defined commands **before** and **after** the actual trace route is performed. This may be useful if you would like to capture any data before the trace route is performed. Commands entered **after** a traceroute is performed are intended as a diagnostic feature and therefore are only issued if the traceroute fails.

Below shows these "pre" trace route commands are defined by through the list-variable **PRE_TRACE_CMD_LIST**. If you do not want to send any pre trace route commands, just comment out all of these lines.

```
lappend PRE_TRACE_CMD_LIST "show ip cef"
lappend PRE_TRACE_CMD_LIST "SLEEP 1.0"
lappend PRE_TRACE_CMD_LIST "PING"
lappend PRE_TRACE_CMD_LIST "ping 10.1.1.1"
```

Below shows these "post" trace route diagnostic commands are defined through the list-variable **DIAG_CMD_LIST**. If you do not want to send any *diagnostic* type commands after a traceroute fails, just comment out all of these lines.

```
lappend DIAG_CMD_LIST "show ip route"
lappend DIAG_CMD_LIST "SLEEP 1.0"
lappend DIAG_CMD_LIST "PING"
lappend DIAG_CMD_LIST "ping 10.1.1.1"
lappend DIAG_CMD_LIST "show ip route DST"
lappend DIAG_CMD_LIST "show ip eigrp neighbors"
lappend DIAG_CMD_LIST "show ip cef DST"
```

In addition to any cisco IOS commands, there are several special commands that the script understands when specified in the PRE_TRACE_CMD_LIST variable. These special commands are CASE SENSITIVE.

Special Pre-Trace Commands	Description
PING	If the command PING (all capitals) is specified, a ping will be sent to the destination for the upcoming traceroute. Note, the source IP address used for the ping will be the same one used for the traceroutes
SLEEP <number in seconds>	If the command SLEEP (all capitals) is specified, the script will delay, in seconds, the amount of time specified before issuing the next pre-trace command.
DST	Substitute the trace route destination IP address into this parameter. Example: lappend PRE_TRACE_CMD_LIST "show ip route DST"
SRC	Substitute the trace route source IP address into this parameter. Example: lappend PRE_TRACE_CMD_LIST "show ip interface brief | include SRC"
VRF	Substitute the trace route source IP address into this parameter. Example: lappend PRE_TRACE_CMD_LIST "show ip route vrf VRF DST" Note, this is only valid when using the tracer input file with support for vrfs (see tracer_vrf_template.txt)

5.2.10.3 Tracer Template File

Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\templates\tracer_template.txt
Linux: /usr/local/net-sense/templates/tracer_template.txt

```
##############################
# Copyright 2002 NetSense Inc.
# www.net-sense.com
# Author: Allan Silverstein
##############################

#####################
# DO NOT MODIFY THE
# TRACER_REV variable
#####################
set TRACER_REV 2

# Template for trace route script (tracer)

# Scenario: Normal Steady State

#############################################################################
# The list of routers to telnet to and perform a trace route
# Field Definitions:
# 1. rtr: Router to telnet to and perform trace routes
# 2. The associated trace route destinations to check on this router
# 3. Source IP Address used for traceroutes from this router.  To use the default
#    address that the router wants to use, just leave the value, between the two
#    double quotes, to nothing, or put in the key word default (see samples below)
#############################################################################
```

```
#                        rtr            dest_list      src_IP
#############################################################################
lappend RTR_LIST [list "br1"          "DEST_LIST_1"  "1.1.1.1"]
lappend RTR_LIST [list "sj2"          "DEST_LIST_2"  ""]
lappend RTR_LIST [list "192.168.1.40" "DEST_LIST_3"  "default"]

#############################################################################################
# Note, the yes/no field is whether or not the traceroute should be successfull.
# This is good for failure scenarios where you may want to confirm that a
# traceroute fails after a failure scenario was introduced
# The expected "trace route path list" is compared against the actual
# path when the script runs
#############################################################################################
# For router: BR1
# Variables                    subnet     yes/no "expected trace route path list"
#############################################################################################
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.2.1 1.1.3.1 1.1.4.1"]
lappend DEST_LIST_1 [list "10.4.4.2" "yes" "1.1.1.1 1.1.6.1 1.1.7.1 1.1.8.1"]

#############################################################################################
# For router: SJ2
# Variables                    subnet     yes/no "trace route path list"
#############################################################################################
lappend DEST_LIST_2 [list "10.30.30.1" "yes" "2.1.1.1 2.1.2.1 2.1.3.1 2.1.4.1"]
lappend DEST_LIST_2 [list "10.4.4.2" "yes" "2.1.1.1 2.1.6.1 2.1.7.1 2.1.8.1"]

#############################################################################################
# For router: 192.168.1.40 (3745)
# Variables                    subnet     yes/no "trace route path list"
#############################################################################################
lappend DEST_LIST_3 [list "10.30.30.1" "yes" "3.1.1.1 3.1.2.1 3.1.3.1 3.1.4.1"]
lappend DEST_LIST_3 [list "10.30.30.1" "yes" "3.1.1.1 3.1.6.1 3.1.6.1 3.1.6.1"]

#############################################################################
# Optional feature which will issue specified commands immediately before
# each traceroute
# None of the commands below can have the requirement of
# entering a confirm, yes , or no after the command
# In addition to listing commands that do not require a
# confirmation after entering the command, the following three
# special commands can be entered in this list
# 1. SLEEP <seconds>  - This command will introduce a delay before entering
#    the next command in the list. MUST BE ALL CAPITAL LETTERS
# 2. PING - This command will ping the address that will be used in the
#           traceroute.  Do not specify an ip address next to this command
#           MUST BE ALL CAPITAL LETTERS
# 3. ping <ip address> -  This command will ping the address specfied.
#############################################################################
#lappend PRE_TRACE_CMD_LIST "show ip cef"
#lappend PRE_TRACE_CMD_LIST "SLEEP 1.0"
#lappend PRE_TRACE_CMD_LIST "PING"
#lappend PRE_TRACE_CMD_LIST "ping 10.1.1.1"


#################################
# OPTIONAL Diagnostic Command List
#################################
#############################################################################
# Commands to enter on a router in the event a traceroute fails,
# or the actual path does not match the expected path
# Inserting the KEY words SRC or DST will substitute the
# source/destination of the failed traceroute in the command.
# The key word TRACEROUTE will perform a traceroute to the destination
# Leave these commands commented out if you do not wanted to issue
# any Diagnostic commands following a failed traceroute.
#
```

```
# None of the commands below can have the requirement of
# entering a confirm, yes , or no after the command
# In addition to listing commands that do not require a
# confirmation after entering the command, the following three
# special commands can be entered in this list
# 1. SLEEP <seconds>  - This command will introduce a delay before entering
#    the next command in the list. MUST BE ALL CAPITAL LETTERS
# 2. PING - This command will ping the address that will be used in the
#           traceroute.  Do not specify an ip address next to this command
#           MUST BE ALL CAPITAL LETTERS
# 3. ping <ip address> -  This command will ping the address specfied.
#########################################################################
#lappend DIAG_CMD_LIST "show ip route"
#lappend DIAG_CMD_LIST "SLEEP 1.0"
#lappend DIAG_CMD_LIST "PING"
#lappend DIAG_CMD_LIST "ping 10.1.1.1"
#lappend DIAG_CMD_LIST "show ip route DST"
#lappend DIAG_CMD_LIST "show ip eigrp neighbors"
#lappend DIAG_CMD_LIST "show ip cef DST"
```

5.2.10.4 Tracer VRF Template File

Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\templates\tracer_vrf_template.txt
Linux: /usr/local/net-sense/templates/tracer_vrf_template.txt

```
##############################
# Copyright 2002 NetSense Inc.
# www.net-sense.com
# Author: Allan Silverstein
##############################

#####################
# DO NOT MODIFY
# TRACER_REV variable
#####################
set TRACER_REV 2

# Template for trace route script (tracer)

#####################################3
# This template file should only be used
# if the trace routes are being performed from
# routers and you need to specify the
# VRF
#####################################3
set VRF 1

# Scenario: Normal Steady State

#########################################################################
# The list of routers to telnet to and perform a trace route
# Field Definitions:
# 1. rtr: Router to telnet to and perform trace routes
# 2. vrf: Name of VRF to use when performing traceroutes from this router
# 3. The associated trace route destinations to check on this router
# 4. Source IP Address used for traceroutes from this router.  To use the default
#    address that the router wants to use, just leave the value, between the two
#    double quotes, to nothing, or put in the key word default (see samples below)
#########################################################################
#                        rtr           VRF      dest_list    src_IP
#########################################################################
lappend RTR_LIST [list "br1"          "vpn1" "DEST_LIST_1" "1.1.1.1"]
lappend RTR_LIST [list "sj2"          "vpn1" "DEST_LIST_2" ""]
lappend RTR_LIST [list "192.168.1.40" "vpn2" "DEST_LIST_3" "default"]
```

```
##############################################################################################
# Note, the yes/no field is whether or not the BGP best route should be present in the routing
# table.  This is good for failure scenarios where you may want to confirm that a route is not
# reachable after a failure of some type.
##############################################################################################
# For router: BR1
# Variables                   subnet     yes/no "trace route path list"
##############################################################################################
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.2.1 1.1.3.1 1.1.4.1"]
lappend DEST_LIST_1 [list "10.4.4.2"   "yes" "1.1.1.1 1.1.6.1 1.1.7.1 1.1.8.1"]

##############################################################################################
# For router: SJ2
# Variables                   subnet     yes/no "trace route path list"
##############################################################################################
lappend DEST_LIST_2 [list "10.30.30.1" "yes" "2.1.1.1 2.1.2.1 2.1.3.1 2.1.4.1"]
lappend DEST_LIST_2 [list "10.4.4.2"   "yes" "2.1.1.1 2.1.6.1 2.1.7.1 2.1.8.1"]

##############################################################################################
# For router: 192.168.1.40 (3745)
# Variables                   subnet     yes/no "trace route path list"
##############################################################################################
lappend DEST_LIST_3 [list "10.30.30.1" "yes" "3.1.1.1 3.1.2.1 3.1.3.1 3.1.4.1"]
lappend DEST_LIST_3 [list "10.30.30.1" "yes" "3.1.1.1 3.1.6.1 3.1.6.1 3.1.6.1"]


##############################################################################
# Optional feature which will issue specified commands immediately before
# each traceroute
# None of the commands below can have the requirement of
# entering a confirm, yes , or no after the command
# In addition to listing commands that do not require a
# confirmation after entering the command, the following three
# special commands can be entered in this list
# 1. SLEEP <seconds>  - This command will introduce a delay before entering
#    the next command in the list. MUST BE ALL CAPITAL LETTERS
# 2. PING - This command will ping the address that will be used in the
#           traceroute.  Do not specify an ip address or vrf next to this command
#           MUST BE ALL CAPITAL LETTERS
# 3. ping <ip address> -  This command will ping the address specfied. Do not
#                         enter the vrf name, the script will automatially do
#                         that.
##############################################################################
#lappend PRE_TRACE_CMD_LIST "show ip cef"
#lappend PRE_TRACE_CMD_LIST "SLEEP 1.0"
#lappend PRE_TRACE_CMD_LIST "PING"
#lappend PRE_TRACE_CMD_LIST "ping 10.1.1.1"


##################################
# OPTIONAL Diagnostic Command List
##################################
##############################################################################
# Commands to enter on a router in the event a traceroute fails,
# or the actual path does not match the expected path
# Inserting the KEY words SRC, DST, or VRF will substitute the
# source/destination or vrf-name of the failed traceroute in the command.
# Leave these commands commented out if you do not wanted to issue
# any Diagnostic commands following a failed traceroute.
#
# None of the commands below can have the requirement of
# entering a confirm, yes , or no after the command
# In addition to listing commands that do not require a
# confirmation after entering the command, the following three
# special commands can be entered in this list
```

```
# 1. SLEEP <seconds>  - This command will introduce a delay before entering
#    the next command in the list. MUST BE ALL CAPITAL LETTERS
# 2. PING - This command will ping the address that will be used in the
#           traceroute.  Do not specify an ip address next to this command
#           MUST BE ALL CAPITAL LETTERS
# 3. ping <ip address> -  This command will ping the address specfied.
#######################################################################
#lappend DIAG_CMD_LIST "show ip route vrf VRF"
#lappend DIAG_CMD_LIST "SLEEP 1.0"
#lappend DIAG_CMD_LIST "PING"
#lappend DIAG_CMD_LIST "ping vrf VRF 10.1.1.1"
#lappend DIAG_CMD_LIST "show ip route vrf VRF DST"
#lappend DIAG_CMD_LIST "show ip eigrp vrf VRF neighbors"
#lappend DIAG_CMD_LIST "show ip cef vrf VRF DST"
```

5.3 Automater Pro Scripts

The Net-Sense Automater Pro Scripts are available through a separate license.

Please contact support@net-sense.com for more information.

5.3.1 BGP Attribute Checker (check_bgp_routes)

The BGP Attribute Checker is designed to verify the BGP attributes of specific routes on a Cisco router. This script will telnet/ssh to a list of routers and perform a “show ip bgp a.b.c.d” on a list of predefined routes. From this output, the BGP “best” path is extracted along with the attributes of the “best” path. These attributes are compared against the expected attributes, which are pre-defined in an input file. Each router in the list has its own unique routes and attributes to check. If the recorded attributes do not match the expected attributes, an error is logged in the “summary.log” file. This script can also accommodate MPLS VPNs where there are multiple routing tables (VRFs) on the router. This script supports both IPv4 and IPv6 routes.

Program Name: check_bgp_routes

Script Argument	Description
-sf <filename>	Input variable file which tells the program which bgp routes to check on which routers. Along with the expected BGP attributes of the route. ***The sample template filename is check_bgp_routes_template.txt*** (**REQUIRED**)
-record <filename>	Instead of verifying the BGP attributes that are defined in the input file. This option just records the BGP attributes and writes out data to to new file. The input file for the -sf option just needs to specify a skeleton input file and the script will fully populate all of the BGP attributes for each prefix. (**OPTIONAL**)
-nhop	Check the BGP Next Hop attribute (**OPTIONAL**)
-metric	Check the Metric (MED) attribute (**OPTIONAL**)
-lpref	Check the BGP local preference attribute (**OPTIONAL**)
-otype	Check the BGP Origin Type attribute (**OPTIONAL**)
-aspath	Check the BGP AS Path attribute (**OPTIONAL**)
-ual (1 or 2)	User Access Level. Tells the script what access-level (1^{st} or 2^{nd}) to log into the router before sending the commands. By default the script will only go into 1^{st} level access.

	(**OPTIONAL but most likely necessary**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

By default, all of the BGP attributes for a given route will be checked. However, if any of the BGP attribute flags (e.g. –nhop, -lpref, etc) are explicitly defined on the command line (or option box for GUI) then ONLY those attributes defined on the command line will be checked. This is useful if you only want to verify one or some of the attributes and not all of them. Thus, your input file (containing expected attributes) only needs to be accurate for the attributes that you wish to verify which saves time when defining the input file.

New with version 5.3 is the -record <filename> option. This is extremely useful when first creating an accurate input file which contains the attributes for certain prefixes. Here, you can use the "skeleton" input file provided (check_bgp_routes_record_template.txt). Copy this file to a new name and edit it to record the prefixes you are interested in for each router. Use this edited file as the argument to the -sf option. Then make up a new filename and use that filename as the argument to the -record option. This option makes creating the all the detailed of the attribute file much easier.

This script can also be used in service provider type environments where there are BGP/MPLS VPNs and multiple routing tables (VRFs) in each router. When setting up the input file, there is a variable which tells the script whether to enter a VPN name when issuing the command "show ip bgp a.b.c.d". If VRFs are present, then the actual command issued will be:

```
ch-per> show ip bgp vpnv4 vrf <vpn_name> <x.x.x.x> <mask>
```

The attributes of the routes/prefixes that will be checked are controlled by the variables in the input file (-sf <filename>). This file is in TCL format and contains TCL list variables. The list variable ***RTR_LIST*** defines the list of routers that the script will telnet/ssh to. It also includes another variable, ***PREFIX_LIST_x***, that defines the BGP routes that will be checked while connected to each router. The value of ***x***, in ***PREFIX_LIST_x***, is a numerical value that must be different for each router.

Below shows a sample entry for the ***RTR_LIST*** variable. The first item in the list "ny1", is the router that the script will telnet to. This must be an IP Address or a name that can be resolved through DNS. If the -pw <*password_file*> option is used, then this IP Address/Name must also be defined in the *password_file*. (Note, they must match exactly [case sensitive]). The 2nd argument is another variable (***PREFIX_LIST_x***) that contains the associated list of BGP routes and attributes to check while in that router

```
lappend RTR_LIST [list "ny1" "PREFIX_LIST_1"]
```

Below shows a sample of ***PREFIX_LIST_x*** and below that is the actual corresponding values that you would see in the router. (Note, this should be on a single line in your input file. If your line becomes too long and you'd like to break it into two lines, you can put a "\" (backslash) at the end of the first line and continue the BGP attribute definitions on the second line. There CANNOT BE ANY spaces after the backslash character.)

lappend PREFIX_LIST_1 [list "150.140.0.0/24" "no" "na" "BEST" "RU" "172.16.20.5" "yes" "172.16.20.5" "N/A" "100" "IGP" "N/A" "N

```
ny1#show ip bgp 150.140.0.0/24
BGP routing table entry for 150.140.0.0/16, version 1453
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  225 225 1005, (received & used)
    175.16.20.5 from 175.16.20.5 (10.10.2.1)
      Origin IGP, localpref 100, valid, external, best
ny1#
```

- The 1st item in the list, “150.140.0.0/24”, is the prefix/mask the BGP attributes will be checked on.
- The second argument must be either “**yes**” or “**no**” (case sensitive). This tells the script whether the routing database is a contained in a VRF. This is for environments where multiple routing tables exist on the router (BGP/MPLS VPNs). If there are multiple routing tables with VRFs, then this parameter should be “yes”. Otherwise for “typical” environments, this value should be “no”.
- The 3rd item is the “VPN Name”. This argument is only significant if the 2nd argument is “yes”. If you are not using BGP/MPLS VPNs, this value can be any string enclosed in double quotes; as it is not referenced. Note, it still must be defined!
- The 4th item, "BEST", states whether this should be the "Best" bgp route or if you are looking for any of the other possible alternate bgp choices for a given prefix. Many time you want to know that the alternate selection is in the BGP table. If your are looking for the alternate route, then this value should be set to "ANY"
- The 5th item, "RU" states whether the BGP entry is "received & used" or or "receive-only". Prefixes that are learned from other peers but are filtered by inbound route-lists or route-maps, will still show up when a "show ip bgp x.x.x." is performed. However, routes that are blocked by inbound prefix lists or route-maps will show up as "receive-only". Note, you must set "bgp soft-reconfiguration inbound" for a peer in order to see this information.
- The 6th item,"172.16.20.5", is the BGP neighbor that the route was learned from, this is not always the same as the BGP next-hop attribute.
- The 7th item, “yes”, can have a value of “***yes”*** or “***no”*** and tells the script whether this route should even be present in the BGP database. In some cases you may want to verify that routes are NOT in the BGP database. If you set this value to “***no***” and the route is present in the BGP database, an error will be logged. To check the attributes for a route, set this value to “***yes***”.
- The 8th item, “175.16.20.5” is the BGP “next-hop” attribute.
- The 9th item, “N/A”, is the BGP MED attribute.
- The 10th item, “100”, is the BGP local preference attribute.
- The 11th item, “IGP”, is the BGP “Origin-type”.
- The 12th item, "N/A" is the BGP weight attribute
- The 13th item, "N/A" is the BGP community attribute.
- The 14th and last item, “ 225 225 1005”, is the “AS Path” attribute. Spaces or tabs can be used to separate the AS numbers when defining the path. In some cases, the AS Path will be “local”. See the sample template for more examples.

If any of the attribute values are not present when issuing the command “show ip bgp x.x.x.x mask”, then put in the string “N/A” (case sensitive) in that field. This tells the script not to search for that attribute for a particular route. The sample above shows this scenario for the 5th item, the BGP MED attribute. Notice, the metric is not defined in the above sample output from the router.

Below shows the sample input file (***check_bgp_routes_template.txt***) that is provided with the program and should be used as a template when creating your own input files. Below shows the

sample input file (***ios_upgrade_template.txt***) that is provided with the program and should be used as a template when creating your own input files. (Note, for MS Windows, the installation utility automatically copies this template file to the C:\Program Files\net-sense\templates directory. For Linux, the templates are in /usr/local/net-sense/templates)

The template input file below instructs the script to do the following:

1. Telnet into router “br1” and check the BGP attributes of the routes listed in PREFIX_LIST_1 (Note, the router br1 uses VRFs)
2. Telnet into router “sj2” and check the BGP attributes of the routes listed in PREFIX_LIST_2 (Note, the router sj2 uses VRFs)
3. Telnet into router “192.168.1.27” and check the BGP attributes of the routes listed in PREFIX_LIST_3 (Note, the router 192.168.1.27 does not use VRFs)

```
#############################################################################
# The list of routers to telnet to and confirm the BGP routes
# Field Definitions:
# 1. rtr: Router to telnet to and check bgp routes
# 2. The associated Prefixs to check on this router
#                       rtr            prefix list name
#############################################################################
lappend RTR_LIST [list "br1"          "PREFIX_LIST_1"]
lappend RTR_LIST [list "sj2"          "PREFIX_LIST_2"]
lappend RTR_LIST [list "192.168.1.27" "PREFIX_LIST_3"]

#########################################################################################################
# The first yes/no field is whether or not the the prefixes being checked are in VRFs or not
#  This basically states whether the script needs to enter
#  show ip bgp" or "show ip vpnv4 vrf vpn_name bgp"
# Note, the second yes/no field is whether or not the BGP best route should be present in the routing
# table.  This is good for failure scenarios where you may want to confirm that a route is not
# reachable after a failure of some type.
# N/A denotes that the attribute is not displayed from the output of the "show ip bgp <prefix/mask> commanc
#########################################################################################################
# For router: br1                                  Use   Vrf    BEST    RO
# Variables                 subnet/mask            vrfs  name   ANY     RU    NBR         yes/no  next_hop
#########################################################################################################
lappend PREFIX_LIST_1 [list "10.134.134.0/28"     "yes" "VPN_A" "BEST" "RU" "0.0.0.0"      "yes" "172.25.0.
lappend PREFIX_LIST_1 [list "10.25.25.1/28"       "yes" "VPN_A" "ANY"  "RU" "172.21.32.10" "yes" "172.21.32
lappend PREFIX_LIST_1 [list "10.45.45.1/28"       "yes" "VPN_A" "ANY"  "RU" "172.24.32.10" "yes" "172.24.32
lappend PREFIX_LIST_1 [list "10.3.3.1/28"         "yes" "VPN_A" "BEST" "RU" "0.0.0.0"      "yes" "172.21.32
lappend PREFIX_LIST_1 [list "fd00:10::1/64"       "yes" "VPN_A" "BEST" "RU" "fd00:20::1"   "yes" "fd00:20::
lappend PREFIX_LIST_1 [list "fd00:12::1/64"       "yes" "VPN_A" "BEST" "RU" "fd00:30::1"   "yes" "fd00:30::

#########################################################################################################
# For router: sj2                                  Use   Vrf    BEST    RO
# Variables                 subnet/mask            vrfs  name   ANY     RU    NBR         yes/no  next_hop
#########################################################################################################
lappend PREFIX_LIST_2 [list "2.2.2.0/24"          "yes" "VPN_B" "BEST" "RU" "0.0.0.0"      "yes" "172.25.0.
lappend PREFIX_LIST_2 [list "3.3.3.0/24"          "yes" "VPN_B" "ANY"  "RU" "172.21.32.10" "yes" "172.21.32
lappend PREFIX_LIST_2 [list "4.4.4.0/24"          "yes" "VPN_B" "ANY"  "RU" "172.24.32.10" "yes" "172.24.32
lappend PREFIX_LIST_2 [list "5.5.5.0/24"          "yes" "VPN_B" "BEST" "RU" "0.0.0.0"      "yes" "172.21.32

#########################################################################################################
# For router: 192.168.1.27                         Use   Vrf    BEST    RO
# Variables                 subnet/mask            vrfs  name   ANY     RU    NBR         yes/no  next_hop
#########################################################################################################
lappend PREFIX_LIST_3 [list "2.2.2.0/24"          "no"  "na"   "BEST"  "RU" "0.0.0.0"      "yes" "172.25.0.
lappend PREFIX_LIST_3 [list "3.3.3.0/24"          "no"  "na"   "ANY"   "RU" "172.21.32.14" "yes" "172.21.32
lappend PREFIX_LIST_3 [list "4.4.4.0/24"          "no"  "na"   "ANY"   "RU" "172.24.32.14" "yes" "172.24.32
lappend PREFIX_LIST_3 [list "5.5.5.0/24"          "no"  "na"   "BEST"  "RU" "0.0.0.0"      "yes" "172.21.32
```

Sample Command: The following command will run the BGP attribute checker utility using the information contained in the file ***check_bgp_routes_east.txt***. The script will not prompt the user for

passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
check_bgp_routes -pw logins.txt -sf check_bgp_routes_east.txt -ual 2
```

5.3.1.1 Template File

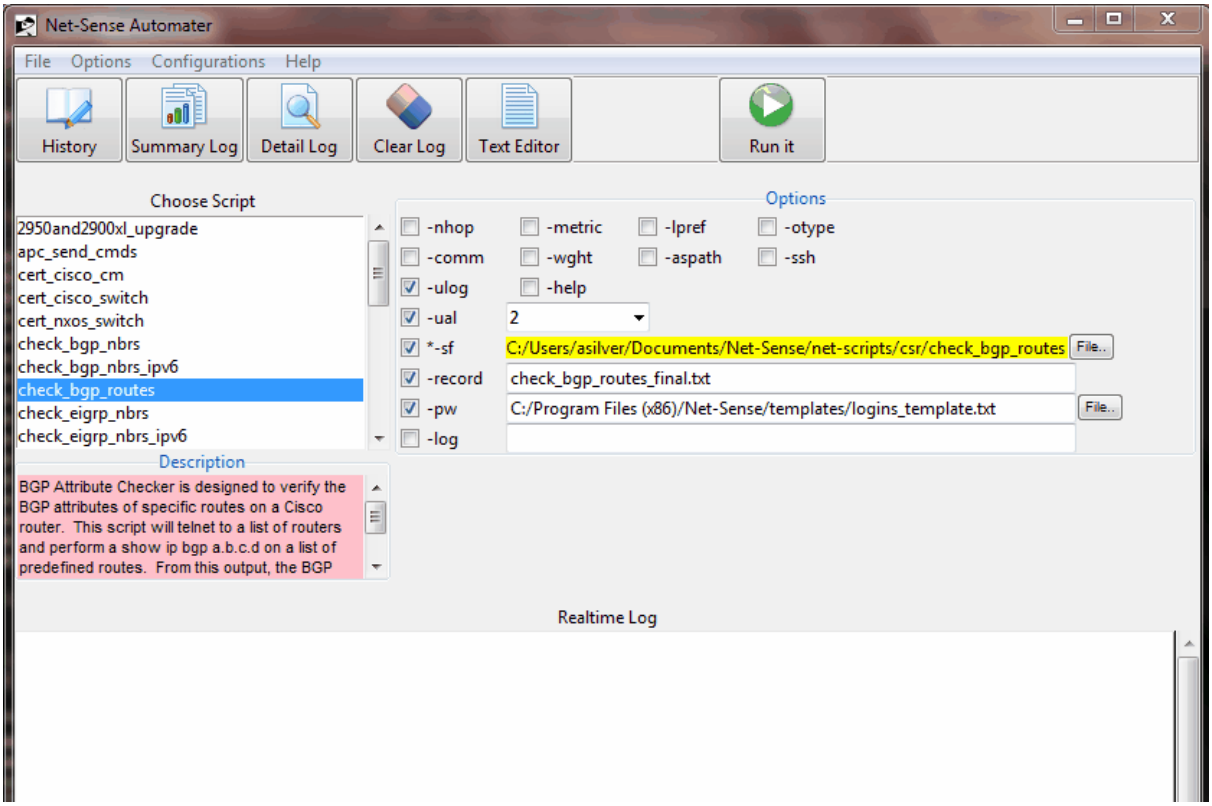
Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\templates\check_bgp_routes_template.txt
Linux: /usr/local/net-sense/templates/check_bgp_routes_template.txt

```
# Copyright 2002 NetSense Inc.
# www.net-sense.com
# Author: Allan Silverstein

set BGP_ROUTE_CHECK_REV 4

# Define BGP route prefixes to be tested

# Scenario: Normal

##############################################################################
# The list of routers to telnet to and confirm the BGP routes
# Field Definitions:
# 1. rtr: Router to telnet to and check bgp routes
# 2. The associated Prefixs to check on this router
#                       rtr             prefix list name
##############################################################################
lappend RTR_LIST [list "br1"          "PREFIX_LIST_1"]
lappend RTR_LIST [list "sj2"          "PREFIX_LIST_2"]
lappend RTR_LIST [list "192.168.1.27" "PREFIX_LIST_3"]

##############################################################################################
# The first yes/no field is whether or not the the prefixes being checked are in VRFs or not
#  This basically states whether the script needs to enter
#  show ip bgp" or "show ip vpnv4 vrf vpn_name bgp"
# Note, the second yes/no field is whether or not the BGP best route should be present in the r
# table.  This is good for failure scenarios where you may want to confirm that a route is not
# reachable after a failure of some type.
# N/A denotes that the attribute is not displayed from the output of the "show ip bgp <prefix/m
##############################################################################################
# For router: br1                                   Use   Vrf    BEST    RO
# Variables                   subnet/mask           vrfs  name   ANY     RU    NBR         yes/n
##############################################################################################
lappend PREFIX_LIST_1 [list "10.134.134.0/28"     "yes" "VPN_A" "BEST" "RU" "0.0.0.0"      "yes
lappend PREFIX_LIST_1 [list "10.25.25.1/28"       "yes" "VPN_A" "ANY"  "RU" "172.21.32.10" "yes
lappend PREFIX_LIST_1 [list "10.45.45.1/28"       "yes" "VPN_A" "ANY"  "RU" "172.24.32.10" "yes
lappend PREFIX_LIST_1 [list "10.3.3.1/28"         "yes" "VPN_A" "BEST" "RU" "0.0.0.0"      "yes
lappend PREFIX_LIST_1 [list "fd00:10::1/64"       "yes" "VPN_A" "BEST" "RU" "fd00:20::1"   "yes
lappend PREFIX_LIST_1 [list "fd00:12::1/64"       "yes" "VPN_A" "BEST" "RU" "fd00:30::1"   "yes

##############################################################################################
# For router: sj2                                   Use   Vrf    BEST    RO
# Variables                   subnet/mask           vrfs  name   ANY     RU    NBR         yes/n
##############################################################################################
lappend PREFIX_LIST_2 [list "2.2.2.0/24"          "yes" "VPN_B" "BEST" "RU" "0.0.0.0"      "yes
lappend PREFIX_LIST_2 [list "3.3.3.0/24"          "yes" "VPN_B" "ANY"  "RU" "172.21.32.10" "yes
lappend PREFIX_LIST_2 [list "4.4.4.0/24"          "yes" "VPN_B" "ANY"  "RU" "172.24.32.10" "yes
lappend PREFIX_LIST_2 [list "5.5.5.0/24"          "yes" "VPN_B" "BEST" "RU" "0.0.0.0"      "yes

##############################################################################################
# For router: 192.168.1.27                          Use   Vrf    BEST    RO
# Variables                   subnet/mask           vrfs  name   ANY     RU    NBR         yes/n
##############################################################################################
lappend PREFIX_LIST_3 [list "2.2.2.0/24"          "no"  "na"   "BEST"  "RU" "0.0.0.0"      "yes
```

```
lappend PREFIX_LIST_3 [list "3.3.3.0/24"          "no"  "na"   "ANY"   "RU" "172.21.32.14" "yes
lappend PREFIX_LIST_3 [list "4.4.4.0/24"          "no"  "na"   "ANY"   "RU" "172.24.32.14" "yes
lappend PREFIX_LIST_3 [list "5.5.5.0/24"          "no"  "na"   "BEST"  "RU" "0.0.0.0"      "yes
```

5.3.1.2 -record option

New with version 5.3 is the ***-record*** <filename> option. This is extremely useful when first creating an accurate input file which contains the attributes for certain prefixes. Here, you can use the "skeleton" input file provided (***check_bgp_routes_record_template.txt***). Copy this file to a new name and edit it to record the prefixes you are interested in for each router. Use this edited file as the argument to the -sf option. Then make up a new filename and use that filename as the argument to the -record option. When the script runs, the contents of the -sf file are copied to the -record file and all of the learned attributes are recorded and copied to the new -record file. The newly created file can then be used for subsequent script runs to validate the condition state of the network.



Below shows a portion of the ***check_bgp_routes_record_template.txt*** file. The first portion lists the routers the script will connect to along with the prefix list name that defines the routes to check on that router. The second portion lists the prefixes to check only for router csr1. For the second part of this skeleton file, only the following three fields need to be defined:

- subnet/mask
- Use Vrfs (yes or no field)
- Vrf Name (set to NA if not applicable)

```
#############################################################################
# The list of routers to connect to and confirm BGP route attributes
# Field Definitions:
# 1. rtr: Router/IP-Address to connect to and check bgp routes
# 2. The associated Prefixs to check on this router
#                          rtr                    prefix list name
#############################################################################
lappend RTR_LIST [list "csr1"                  "PREFIX_LIST_1"]
lappend RTR_LIST [list "csr2"                  "PREFIX_LIST_2"]
lappend RTR_LIST [list "192.168.21.13"         "PREFIX_LIST_3"]


#############################################################################################################
# For router: csr1
#                                                 Use  Vrf    BEST   RO
# Variables                      subnet/mask      vrfs name   ANY    RU NBR yes/no next_hop metric p
#############################################################################################################
lappend PREFIX_LIST_1 [list "192.168.100.2/32" "no" "NA"  ""    "" ""   ""     ""       ""     "
lappend PREFIX_LIST_1 [list "192.168.100.3/32" "no" "NA"  ""    "" ""   ""     ""       ""     "
lappend PREFIX_LIST_1 [list "192.168.100.4/32" "no" "NA"  ""    "" ""   ""     ""       ""     "
```

5.3.2 BGP Neighbor Checker (check_bgp_nbrs)

The check BGP neighbor script is designed to check the BGP state of a predefined list of BGP peers. The input file contains a list of routers, along with the BGP peers to check for each router. The script can validate whether the peer is supposed to be up or down. The script also contains support for VRFs as well as IPv4 and IPv6.

Program Name: check_bgp_nbrs

Script Argument	Description
-rtr <name or IP>	Only check BGP peers for this one router. The script will search the file specified by the -sf <filename> argument for this router. Then it will only check the BGP peers associated with this router. This is useful if your -sf <file> has many routers and BGP nbrs defined but you would like to run a quick test and only check the nbrs for a particular router. **(OPTIONAL)**
-sf <filename>	Input variable file, which defines the routers the script will telnet to and which BGP peers will be checked. (**REQUIRED**) ***The sample template filename is check_bgp_nbrs_template.txt***
-ual (1 or 2)	User Access Level. Tells the script what access-level (1^{st} or 2^{nd}) to log into the router before sending the commands. By default the script will only go into 1^{st} level access. (**OPTIONAL but most likely need to set to level 2**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)

-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The BGP peers that will be verified are controlled by the variables in the input file (-sf <filename>). This file is in TCL format and contains TCL list variables. The list variable ***RTR_LIST*** defines the list of routers that the script will telnet to. It also includes another variable, ***NBR_LIST_x***, that defines the BGP peers that will be checked while telneted into each router. The value of ***x***, in ***NBR_LIST_x***, is a numerical value that must be different for each router.

Below shows a sample entry for the ***RTR_LIST*** variable. The first item in the list "ny-rtr1" is the router that the peers will be checked for. The script will actually telnet into this router. This must be an IP Address or a name that can be resolved through DNS. If the -pw <*password_file*> option is used, then this IP Address/Name must also be defined in the *password_file*. (Note, they must match exactly [case sensitive]). The second argument is another variable (***DEST_LIST_x***) that contains the associated list of BGP peers to check while in that router.

```
lappend RTR_LIST [list "ny-rtr1" "$NBR_LIST_1"]
```

Below shows a sample of ***NBR_LIST_x***. The first item in list specifies if an IPv4 or IPv6 neighbor address is being checked (ipv4 or ipv6). The second item in the list, "***yes***" states whether or not to use VRFs. This item can have a value of ***yes*** or ***no*** . The third item in the list "vpn35" is the VRF name. If VRFs are not being used then set this field to NA (the field is ignored). The fortch item, "UP", states that this BGP peer should be UP. The possible values for this field are **UP** or **DOWN** (case insensitive). Note, the script considers a BGP peer to be UP only if the real BGP state is **ESTABLISHED**.

lappend NBR_LIST_1 [list "ipv4" "yes" "vpn35" "UP" "65" "10" "0" "20-24"]

The last four fields (for ipv4), or last eight fields (for ipv6) define the IP address(es) to the BGP peers to validate. These fields have limited support for ***wildcarding,*** meaning multiple BGP peers can be defined on one line. This is achieved by specifying a range in any of the IP Address fields. A range is defined by inserting a "-" between two numbers. The example above will validate that the following BGP peers are all "UP"

- 65.10.0.20
- 65.10.0.21
- 65.10.0.22
- 65.10.0.23
- 65.10.0.24
-

Here is an IPv6 example:
lappend NBR_LIST_1 [list "ipv6" "no" "NA" "UP" "FD00" "100" "0" "0" "0" "0" "A4" "1"]

Sample Command: The following command will run the check_bgp_nbrs script and verify the bgp peer state for the routers and peers defined in the file check_bgp_nbrs_eastcoast.txt. The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option). The -ual 2 option is needed because the script needs to go into enable mode. The -ulog option will cause the detailed trace log file to be saved to a unique filename that will not get overwritten.

```
check_bgp_nbrs -ulog -pw logins.txt -sf check_bgp_nbrs_eastcoast.txt -ual 2
```

5.3.2.1 Template File

Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\templates\check_bgp_nbrs_template.txt
Linux: /usr/local/net-sense/templates/check_bgp_nbrs_template.txt

```
##############################
# Copyright 2002 NetSense Inc.
# www.net-sense.com
# Author: Allan Silverstein
##############################

set BGP_NBR_CHECK_REV 2

########################################
# Template for Check BGP NBR script
# Support for both IPv4 and IPv6 addresses
########################################

# Scenario: Normal

#############################################################################
# The list of routers to telnet to and perform a show ip bgp neighbor
# Field Definitions:
# 1. rtr: Router to telnet to and perform BGP NBR checks
# 2. The associated BGP NBRS to check
#############################################################################
lappend RTR_LIST [list "nyrtr1" "NBR_LIST_1"]
lappend RTR_LIST [list "cartr1" "NBR_LIST_2"]

#############################################################################################################
# The IP Version field must be set to either ipv4 or ipv6
# Note, the yes/no field is whether or not VRFs are being used.  If set to yes
# then the 3rd field should be set to the VRF name.  If set to no, then enter NA
# The Peer Status Field should be set to UP or DOWN (not case sensitive)
# Octet fields can be defined with ranges (e.g. 30-40)
#############################################################################################################
# For router: nyrtr1-pr01
#                          IP     VRFs    VRF          Peer      1st   2nd   3rd    4th
# Variables               v4/v6  yes/no   name        Status    octet octet  octet  octet
#############################################################################################################
lappend NBR_LIST_1 [list "ipv4"  "yes"   "acme_vrf"   "UP"     "10"  "25"   "0"    "33-36"]
lappend NBR_LIST_1 [list "ipv4"  "yes"   "acme_vrf"   "UP"     "10"  "25"   "1"    "33"]
lappend NBR_LIST_1 [list "ipv4"  "no"    "NA"         "UP"     "10"  "25"   "2"    "33-36"]
lappend NBR_LIST_1 [list "ipv4"  "no"    "NA"         "UP"     "10"  "25"   "3"    "33"]
lappend NBR_LIST_1 [list "ipv6"  "no"    "NA"         "UP"     "FD00" "100" "0" "0" "0" "0" "A4
lappend NBR_LIST_1 [list "ipv6"  "no"    "NA"         "UP"     "FD00" "100" "0" "0" "0" "0" "A4
lappend NBR_LIST_1 [list "ipv6"  "no"    "NA"         "UP"     "FD00" "100" "0" "0" "0" "0" "A4
lappend NBR_LIST_1 [list "ipv6"  "no"    "NA"         "DOWN"   "FD00" "100" "0" "0" "0" "0" "A4


#############################################################################################################
# For router: cartr1-pr01
#                          IP     VRFs    VRF          Peer      1st   2nd   3rd    4th
# Variables               v4/v6  yes/no   name        Status    octet octet  octet  octet
#############################################################################################################
lappend NBR_LIST_2 [list "ipv4"  "yes"   "acme_vrf"   "UP"     "20"  "10"   "0"    "1-10"]
lappend NBR_LIST_2 [list "ipv4"  "yes"   "acme_vrf"   "UP"     "20"  "10"   "1"    "33"]
lappend NBR_LIST_2 [list "ipv4"  "no"    "NA"         "UP"     "20"  "10"   "2"    "100-105"]
lappend NBR_LIST_2 [list "ipv4"  "no"    "NA"         "UP"     "20"  "10"   "3"    "33"]
```

5.3.3 EIGRP Neighbor Checker (check_eigrp_nbrs)

The check EIGRP neighbor script is designed to verify that EIGRP neighbors are established between routers. An input file defines the EIGRP neighbors, along with the EIGRP peers to check for each router. EIGRP neighbors for the global route table as well as neighbors in VRFs can be validated. The script also contains support for VRFs as well as IPv4 and IPv6.

Program Name: check_eigrp_nbrs

Script Argument	Description
-sf <filename>	Input variable file, which defines the routers the script will telnet to and which EIGRP neighbors will be checked. (**REQUIRED)** ***The sample template filename is check_eigrp_nbrs_template.txt***
-rtr <ip or rtr_name>	Only check EIGRP neighbors for this one router. The script will search the file specified by the -sf <filename> argument for this router. Then it will only check the EIGRP neighbors associated with this router. This is useful if your -sf <file> has many routers and EIGRP nbrs defined but you would like to run a quick test and only check the nbrs for a particular router. **(OPTIONAL)**
-qcount	Validates that the qcount field, for a given neighbor, is 0. If it is not zero, then an error is logged. This is an optional check. (**OPTIONAL**)
-ual (1 or 2)	User Access Level. Tells the script what access-level (1st or 2nd) to log into the router before sending the commands. By default the script will only go into 1st level access. (**OPTIONAL but most likely need to set to level 2**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The EIGRP neighbors that will be verified are controlled by the variables in the input file (-sf <filename>). This file is in TCL format and contains TCL list variables. The list variable ***RTR_LIST*** defines the list of routers that the script will telnet to. It also includes another variable, ***NBR_LIST_x***, that defines the EIGRP nbrs that will be checked while telneted into each router. The value of ***x***, in ***NBR_LIST_x***, is a numerical value that must be different for each router.

Below shows a sample entry for the ***RTR_LIST*** variable. The first item in the list “ny-rtr1”, is the router that the neighbors will be checked for. The script will actually telnet into this router. This must be an IP Address or a name that can be resolved through DNS. If the -pw <*password_file*> option is used, then this IP Address/Name must also be defined in the *password_file*. (Note, they must match exactly [case sensitive]). The second argument is another variable (***NBR_LIST_x***) that contains the associated list of EIGRP peers to check while in that router.

```
lappend RTR_LIST [list "ny-rtr1" "$NBR_LIST_5"]
```

Below shows a sample of ***NBR_LIST_x***. The first item in the list states whether the following fields in that row are defining ipv4 or ipv6 neighbors. the value must be either "***ipv4***" or "***ipv6***". The second item in the list, "***yes***" is states whether or not to use VRFs. This item can have a value of ***yes*** or ***no*** . The third item in the list "NA" is the VRF name (here vrfs are not being used). If VRFs are not being used then set this field to NA (the field is ignored).

lappend NBR_LIST_26 [list "ipv4" "no" "NA" "10" "3" "1-4" "1"]

For IPv4 addresses, the last four fields define the IP address(es) to the EIGRP neighbors to validate. These 4 fields have limited support for ***wildcarding,*** meaning multiple EIGRP neighbors can be defined on one line. This is achieved by specifying a range in any of the IP Address fields. A range is defined by inserting a "-" between two numbers. The example above will validate that the following EIGRP neighbors are present.

- 10.3.1.1
- 10.3.2.1
- 10.3.3.1
- 10.3.4.1

For IPv6 address, the last eight fields define the IP address(es) to the EIGRP neighbors to validate. These 8 fields have limited support for ***wildcarding,*** meaning multiple EIGRP neighbors can be defined on one line. This is achieved by specifying a range in any of the IP Address fields. A range is defined by inserting a "-" between two numbers. The example below also shows the ipv6 EIGRP neighbors that will be validated:

lappend NBR_LIST_26 [list "ipv4" "no" "NA" "FD11" "10" "C" "100" "0" "0" "0" "8-B"]

- FD11:10:C:100::8
- FD11:10:C:100::9
- FD11:10:C:100::A
- FD11:10:C:100::B

Sample Command: The following command will run the ***check_eigrp_nbrs*** script and verify the eigrp neighbors on the routers defined in the file ***check_eigrp_nbrs_emea.txt***. The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option). The -ual 2 option is needed because the script needs to go into enable mode.

```
check_eigrp_nbrs -pw logins.txt -sf check_eigrp_nbrs_emea.txt -ual 2
```

5.3.3.1 Template File

Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\templates\check_eigrp_nbrs_template.txt
Linux: /usr/local/net-sense/templates/check_eigrp_nbrs_template.txt

```
############################
# Copyright 2002 NetSense Inc.
# www.net-sense.com
# Author: Allan Silverstein
############################

set EIGRP_NBR_CHECK_REV 2

# Template for EIGRP NBR check script

# Scenario: Normal

#########################################################################
# The list of routers to telnet to and perform a trace route
```

```
# Field Definitions:
# 1. rtr: Router to telnet to and perform EIGRP NBR checks
# 2. The associated EIGRP NBRS to check
#################################################################################
lappend RTR_LIST [list "nyrtr1" "NBR_LIST_1"]
lappend RTR_LIST [list "sjrtr1" "NBR_LIST_2"]

#########################################################################################################
# Note, the yes/no field is whether or not VRFs are being used.  If set to yes
# then the 2nd field should be set to the VRF name.  If set to no, then enter NA
# Octet fields can be defined with ranges (e.g. 30-40)
#########################################################################################################
# For router: nyrtr1-pr01
#                          IP      VRFs    VRF       1st   2nd   3rd     4th
# Variables              v4/v6   yes/no   name      octet octet  octet   octet
#########################################################################################################
lappend NBR_LIST_1 [list "ipv4"   "no"    "NA"      "10"  "10"   "1"    "2-51"]
lappend NBR_LIST_1 [list "ipv4"   "no"    "NA"      "10"  "200"  "9"    "4"]
lappend NBR_LIST_1 [list "ipv4"   "no"    "NA"      "10"  "201"  "100"  "7"]
lappend NBR_LIST_1 [list "ipv4"   "no"    "NA"      "10"  "207"  "9"    "10"]
lappend NBR_LIST_1 [list "ipv4"   "no"    "NA"      "10"  "222"  "9"    "4"]
lappend NBR_LIST_1 [list "ipv4"   "no"    "NA"      "10"  "120"  "102"  "1"]
lappend NBR_LIST_1 [list "ipv4"   "no"    "NA"      "10"  "5"    "34"   "7"]
lappend NBR_LIST_1 [list "ipv6"   "no"    "NA"     "FD00" "100"  "0"    "0" "0" "0" "A4" "1"] ;#
lappend NBR_LIST_1 [list "ipv6"   "no"    "NA"     "FD00" "100"  "0"    "0" "0" "0" "A4" "2"] ;#
lappend NBR_LIST_1 [list "ipv6"   "no"    "NA"     "FD00" "100"  "0"    "0" "0" "0" "A4" "3"] ;#


#########################################################################################################
# For router: sjrtr1-pr01
#                          IP      VRFs    VRF       1st   2nd   3rd     4th
# Variables              v4/v6   yes/no   name      octet octet  octet   octet
#########################################################################################################
lappend NBR_LIST_2 [list "ipv4"   "yes"   "VPN_A"   "10"  "125"  "5"    "5-15"]
lappend NBR_LIST_2 [list "ipv4"   "yes"   "VPN_A"   "10"  "125"  "100"  "1"]
```

5.3.4 EIGRP Route Checker (check_eigrp_routes)

This script is designed to validate EIGRP route attributes for a predefined list of EIGRP routes. The list of routes along with the expected EIGRP attributes for those routes is defined in an input file (-sf <filename>). The script will telnet to a list of routers and then perform a show ip route a.b.c.d x.x.x.x (or show ip route vrf vrf_name a.b.c.d x.x.x.x) for each route. The EIGRP attributes will then be extracted from the output and compared against the expected attributes that were defined in the input file. If the attributes do not match up, an error message will be logged. This script supports both IPv4 and IPv6 routes.

Program Name: check_eigrp_routes

Script Argument	Description
-sf <filename>	Input variable file, which is a database like file, which contains information about the EIGRP routes to check along with the attributes of those routes. The name of the file that contains EIGRP info for a steady state condition is: • **check_eigrp_routes_template.txt** (**REQUIRED**)
-record	Instead of verifying the EIGRP route attributes that are defined in the input file. This option just records the EIGRP attributes and writes out the data to

	a new file. The input file for the -sf option just needs to specify a skeleton input file and the script will fully poputlate all of the EIGRP attributes for each subnet. (**OPTIONAL**)
-nhop	Check Next Hop value (**OPTIONAL**)
-metric	EIGRP Route Metric (**OPTIONAL**)
-distance	Admin Distance of Route (**OPTIONAL**)
-delay	EIGRP Delay (**OPTIONAL**)
-hops	hop count (**OPTIONAL**)
-tag	Route Tag (**OPTIONAL**)
-ual (1 or 2)	User Access Level. Tells the script what access-level (1^{st} or 2^{nd}) to log into the router before sending the commands. By default the script will only go into 1^{st} level access. (**OPTIONAL but most likely need to set to level 2**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

By default, all of the EIGRP route metrics for a given route will be checked. However, if any of the EIGRP attibute flags (e.g. –nhop, -tag, etc) are explicitly defined on the command line (or option box for GUI) then ONLY those attributes defined on the command line will be checked. This is useful if you only want to verify one or some of the attributes and not all of them. Thus, your input file (containing expected attributes) only needs to be accurate for the attributes that you wish to verify which saves time when defining the input file.

The following EIGRP attributes are checked:

- Learned from Neighbor
- Next Hop
- Admin Distance
- Route Metric
- Delay
- Hops
- Route Tag

New with version 5.3 is the -record <filename> option. This is extremely useful when first creating an accurate input file which contains the attributes for certain prefixes. Here, you can use the "skeleton" input file provided (check_eigrp_routes_record_template.txt). Copy this file to a new name and edit it to record the prefixes you are interested in for each router. Use this edited file as the argument to the -sf option. Then make up a new filename and use that filename as the argument to the -record option. This option makes creating the all the detailed of the attribute file much easier.

Sample Command: The following command will run the check_eigrp_routes script and validate the EIGRP attributes for all of the routes defined in the file check_eigrp_routes_acme.txt. The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (

-pw option). The -ual 2 option is needed because the script needs to go into enable mode.

```
check_eigrp_routes -pw logins.txt -sf check_eigrp_routes_acme.txt -ual 2
```

5.3.4.1 Template File

Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\templates\check_eigrp_routes_template.txt
Linux: /usr/local/net-sense/templates/check_eigrp_routes_template.txt

```
# Copyright 2002 NetSense Inc.
# www.net-sense.com
# Author: Allan Silverstein

set EIGRP_ROUTE_CHECK_REV 4

# Define EIGRP route prefixes to be tested

# Scenario: Normal Steady State

#################################################################################
# The list of routers to telnet to and confirm the EIGRP route metrics
# Field Definitions:
# 1. rtr: Router to telnet to and check eigrp routes
# 2. The associated Prefixs to check on this router
#                          rtr       Prefix list to reference
#################################################################################
lappend RTR_LIST [list "nyrtr1"  "PREFIX_LIST_1"]
lappend RTR_LIST [list "sjrtr1"  "PREFIX_LIST_2"]
lappend RTR_LIST [list "gartr1"  "PREFIX_LIST_3"]

#########################################################################################################
# The first yes/no field is whether or not the route is in a VRF
#       This basically states whether the script needs to enter
#       "show ip route x.x.x.x" or "show ip route vrf vrf_name x.x.x.x"
# Note, the second yes/no field is whether or not the route should be present in the routing
# table.  This is good for failure scenarios where you may want to confirm that a route is not
# reachable after a failure of some type.
# Note, VRF name is only relevant if VRF yes/no field is "yes"
# N/A = Not Applicable  (all capitals)
#########################################################################################################

#########################################################################################################
# For router: nyrtr1                                                    Route
#                                                          VRF   VRF    there?
# Variables                  subnet                       yes/no name   yes/no next_hop         le
#########################################################################################################
lappend PREFIX_LIST_1 [list "10.10.1.0/24"              "no"  "N/A"  "yes"  "10.3.1.1"        "
lappend PREFIX_LIST_1 [list "10.10.2.0/24"              "no"  "N/A"  "yes"  "10.3.1.1"        "
lappend PREFIX_LIST_1 [list "10.10.3.0/24"              "no"  "N/A"  "yes"  "10.5.1.1"        "
lappend PREFIX_LIST_1 [list "FD00:1::100:1/64"          "no"  "N/A"  "yes"  "FDD00:1::101:1"  "

#########################################################################################################
# For router: sjrtr1                                                    Route
#                                                          VRF   VRF    there?
# Variables                  subnet                       yes/no name   yes/no next_hop         le
#########################################################################################################
lappend PREFIX_LIST_2 [list "10.10.1.0/24"              "no"  "N/A"  "yes"  "10.3.2.1"        "
lappend PREFIX_LIST_2 [list "10.10.4.0/24"              "no"  "N/A"  "yes"  "10.3.2.1"        "
lappend PREFIX_LIST_2 [list "10.10.7.0/24"              "no"  "N/A"  "yes"  "10.6.2.1"        "
lappend PREFIX_LIST_2 [list "FD00:1::102:1/64"          "no"  "N/A"  "yes"  "FDD00:1::103:1"  "
lappend PREFIX_LIST_2 [list "FD00:1::104:1/64"          "no"  "N/A"  "yes"  "FDD00:1::105:1"  "

#########################################################################################################
# For router: gartr1                                                    Route
```

```
#                                                          VRF   VRF   there?
# Variables                    subnet                     yes/no name  yes/no      next_hop
#########################################################################################################
lappend PREFIX_LIST_3 [list "10.10.4.0/24"                "yes" "VPN_A" "yes"  "10.201.1.1"
lappend PREFIX_LIST_3 [list "10.10.7.0/24"                "yes" "VPN_A" "yes"  "10.201.1.1"
lappend PREFIX_LIST_3 [list "10.10.10.0/24"               "yes" "VPN_A" "yes"  "10.201.1.1"
lappend PREFIX_LIST_3 [list "FD00:1::106:1/64"            "yes" "VPN_A" "yes"  "FDD00:1::107:1"
lappend PREFIX_LIST_3 [list "FD00:1::108:1/64"            "yes" "VPN_A" "yes"  "FDD00:1::109:1"
```

5.3.5 Router Configuration Tool with Variables

Global Router Configuration Tool with Variables (config_devices_rvf)

This program is similar to the cisco_config_cmds script except it features variable substitution within the command list. This script is designed to send **configuration** commands only. **DO NOT** use this script to send exec level commands (e.g., show commands). The specific configuration commands, variables, and the routers these commands are sent to, are defined in a text file that you create. In addition, the router list, variables, and commands are all defined in one file. The format of the text file is described below. The file containing the configuration commands MUST NOT include the commands needed to enter and exit configuration mode on the router. The script will automatically enter those. The commands are sent in sequential order from top to bottom of the file.

Program Name: config_devices_rvf

Script Argument	Description
-rvf <filename>	Short for **R**outer **V**ariable **F**ile. File which contains a List of routers or IP Address along with a file name which contains the list of configuration commands to send to each router (**REQUIRED**)
-testmode	Test Mode. Build and display the commands that would have been sent to the devices but DO NOT actually connect to the devices and send the commands. Useful for confirming your variable substitutions are correct. List of commands are also written to the file $SCRIPT_HOME/config_devices_rvf_sample_cmds.txt". (**OPTIONAL**)
-wm	Saves configuration file to NVRAM after making the configuration changes. This option performs a “write mememory” on the router. (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with –pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename

	automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

Below is the sample template file that is located in the templates directory. This is not a TCL formatted file, it is just a text file. Lines that begin with a # are comments. There are 2 "sections" to this file. The first section defines the list of routers along with the values of each variable for that specific router. The second section contains the list of commands to send along with references to those variables. All lines that begin with "--" define the router/switch to login to along with the variables. The script interprets all other lines (except comments, or special keywords e.g., SLEEP) as commands to to send to the routers. The first section contains lines as follows:

```
--nyrtr1:1.1.1.1:255.255.255.0:vlan 1:Segment-1:var5
```

The string following "--" is the router/switch that will be logged into (nyrtr1). Each of the other values are variables which must be separated by a colon. For the example above, there are 5 variables. Note, it is NOT a requirement that each variable defined be used. DO NOT PUT A COLON AFTER THE LAST VARIABLE. Each router/switch is allowed to have up to 9 variables defined for it. The name of each variable is VAR1, VAR2, VAR3, etc.

The 2nd section of this file contains the commands to send to each router. Referencing a variable is performed using the key word ***VAR*** (**all CAPITALS**) preceded by a $ and followed by the number of the variable you wish to reference (no spaces between them).

```
interface $VAR3
 ip address $VAR1 $VAR2
 description test_$VAR4
```

For nyrtr1 the following commands will be sent

```
interface vlan 1
 ip address 1.1.1.1 255.255.255.0
 description Mgmt_Segment-1
```

```
###################################
# Template:  Router Variable File
###################################
# Sample template file for config_devices_rvf script
# Lines that begind with a # are comments

# Specify the list of routers along with varialbles here
# All lines that begin with -- define the routers/switches
# that will be logged into.
# The format of this line is as follows
# router:var1:var2:var3:var4:var5:
# There can be up to 9 variables defined
# Variables are seperated by a colon ":"
# Do not put a colon at the end of the last variable
#########################################################
--nyrtr1:1.1.1.1:255.255.255.0:vlan 1:Segment-1:var5
--nyrtr2:1.1.1.2:255.255.255.0:vlan 1:Segment-2:var5
--nyrtr3:1.1.1.3:255.255.255.0:vlan 1:Segment-3:var5
--nyrtr4:1.1.1.4:255.255.255.0:vlan 1:Segment-4:var5
--nyrtr5:1.1.1.5:255.255.255.0:vlan 1:Segment-100:var5
--nyrtr6:1.1.1.6:255.255.255.0:vlan 1:Segment-156:var5
```

```
--nyrtr7:1.1.1.7:255.255.255.0:vlan 1:Segment-175:var5
--nyrtr8:1.1.1.8:255.255.255.0:vlan 1:Segment-176:var5
--nyrtr9:1.1.1.9:255.255.255.0:vlan 1:Segment-177:var5
--nyrtr10:1.1.1.10:255.255.255.0:vlan 1:Segment-178:var5

#########################################################
# List of commands to send to routers/switches
# Variables must be referenced as $VARx where x is the
# variable number defined above.  The string var must be
# all CAPITALS!
#########################################################
interface $VAR3
 ip address $VAR1 $VAR2
 description Mgmt_$VAR4
```

This script also has the option to run in ***Test Mode*** using the -testmode option defined above. With this option, the exact commands that would have been sent to the routers are written to a file (stored in the SCRIPT_HOME directory). The script does not actually login to any devices or send any commands. It is recommended this feature be used when working in production environment; as it may be easy to make a mistake with your variable substitutions.

By default, the configuration commands will not be saved to NVRAM. Use the ***–wm*** option to save the configuration changes.

This program also has the option to be run in ***Safe*** and ***SuperSafe*** mode which should be considered when running scripts in production environments. If entering a configuration command results in an error on the router and the script is running in ***Safe*** or ***SuperSafe*** mode, the config will not be saved to NVRAM even if the –wm option is applied. If the script is NOT running in ***Safe*** mode, and the –wm option is applied, the config WILL be saved to NVRAM even if there are one or more commands that caused configuration errors. If the script encounters an error sending a command and it is run in SuperSafe mode, the -wm will not be performed, as the script will immediately abort.

Sample Command: The following script will send configuration commands to the routers defined in the file rtr_update_rvf.txt. After the configuration commands are entered, the config will be saved to NVRAM because of the ***–wm*** option. If there are any errors while sending a particular configuration command, the script will abort the script because of the ***–ssafe*** option (SAFE Mode). The detailed trace-log will be saved to a unique time-stamped filename to the SCRIPT_HOME directory because the –ulog option is used. The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
config_devices_rvf -ulog -pw logins.txt –rvf rtr_update_rvf.txt –wm –ssafe
```

5.3.6 IOS Upgrader (ios_upgrade)

The IOS upgrade script is for upgrading Cisco IOS based devices. The script will connect to a list of routers/switches and download (via TFTP or FTP) the new IOS image specified. The devices upgraded along with the files downloaded are defined in an input file. In addition to downloading a new IOS image, the script also has the ability to do the following:

- Delete old image from flash
- Verify new IOS image is compatible with platform
- Verify there is enough free memory in flash before performing download
- Install new boot system commands in router configuration and save to NVRAM
- Run in Pre-test mode to perform to perform verification checks without downloading new IOS image

This is the only script in the Automater suite of scripts that connects to multiple routers/switches in parallel. This feature was added to save time because downloads can take a quite a bit of time, depending on available bandwidth. Although this saves time, you cannot see the upgrades taking place in real time in the display log window of the GUI The maximum number of simultaneous sessions is controlled by the MAX_SESSIONS variable in the input file. The default value for MAX_SESSIONS is 5.

Program Name: ios_upgrade

Script Argument	Description
-sf <filename>	Input variable file which lists the routers to be upgraded along with specific details about each router. It also contains file information for each IOS image. ***The sample template filename is ios_upgrade_template.txt*** (**REQUIRED**)
-pretest	Runs the script in PreTest Mode. In PreTest mode, the script will read the input file and log into each device to verify there is enough free memory in flash and that the new IOS image is compatible with the platform. The IOS download is not performed. It is highly recommended that this option be used before actually upgrading your IOS devices. (**OPTIONAL**)
-del	Delete a file in flash. The specific file to delete is defined in the input variable file (from the -sf option). The delete is performed before free memory is checked and before the new IOS file is downloaded. Note, even if the file to delete is specified in the input file, the file will not be deleted unless this option is used. (**OPTIONAL**)
-ovw	Overwrite the existing file in flash. This option is useful if the file that you are downloading has the same exact name as a file all ready in flash. (**OPTIONAL**)
-config	Configure new "boot system" commands which reflect the new IOS image downloaded. If used, after a successful IOS image download, the script will remove the existing boot system commands and put a new boot system command reflecting the new IOS image. It will then put the original boot system commands below the new one. Note, it will not reinstall a boot system command for a deleted IOS file. The script will also copy the running-config to startup-config. (**OPTIONAL**)
-dncf	Do Not Compare Files. By default the script will compare the new IOS image filename with the IOS image filename currently running on the device; to verify compatibility. The safety check assumes the the filenames of the IOS images are following Cisco's standard naming convention (e.g. c7200-js-mz.124-4.T) where the part preceding the first \"-\" states the platform type. If the new IOS image and the current running image do not match the platform type in the filename, then the script will not upgrade that device. Use this option to override this safety check. (**OPTIONAL**)
-autodir	Tells the script where to automatically create a unique directory name to store individual log files for the IOS upgrades. Using the date option will create a directory with just the date in it (e.g. 08012010 for Aug 8th 2010). Using the time option will create a directory with the date and time in it (e.g. 08012010_12h15m36s for Aug 8th 2010 12:15:36). The script will use "time" option by default. The directory will be created under the $SCRIPT_HOME/ios_upgrades directory. (OPTIONAL)

-ftp_passwd	When devices are being upgraded, transfers can be done via ftp or tftp. If ftp is selected then a username and password are necessary. The username and password can either be specified in the router/switch configuration file or on the command line when performing the ftp transfer. Specifying this option tells the script to use the ftp username/password variables defined in the input file: set FTP_USERNAME jsmith set FTP PASSWORD foobar and to put them right on the command line when issuing the ftp transfer command on the ios device.
-ual (1 or 2)	Login into device with 1st or 2nd level (privileged) access. Default is 1st level. You will most likely need 2nd level access for this script unless this is a customized RADIUS or TACACS environment.(**OPTIONAL but most likely necessary**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The devices to upgrade and the IOS images are defined in a TCL formatted text file that is read in by the script (***-sf*** <filename>). This file can be created/edit through the GUI or through a text editor. (It is recommended to use the GUI when getting started). To create/edit through the GUI, from the top menu bar: ***Configurations->IOS Upgrade File***. If editing the file directly through a text editor, you do not need to understand TCL to configure the input variable file; just follow the template file. This file contains two main lists.

The first list (***RTR_LIST***) defines the routers to upgrade and attributes about each router. Below shows a sample entry for the ***RTR_LIST*** variable and the table below it describes each of the six fields in the row.

```
lappend RTR_LIST [list "ny-rtr1" "c2900-universalk9-mz.SPA.153-2.T.bin"    "tftp"   "192.168.56.1" "flash0:
```

Field	Description
1	IOS device that the script will telnet/ssh to. This must be an IP Address or a name that can be resolved through DNS. If the -pw <*password_file*> option is used, then this IP Address/Name must also be defined in the *password_file*. (Note, they must match exactly [case sensitive], unless the DEFAULT entry is present in the password file).
2	New IOS image that will be downloaded to the device (e.g. c2900-universalk9-mz.SPA.153-2.T.bin)
3	Transfer Protocol to use. Currentl only TFTP and FTP are supported. Use lower case letters.

4	IP Address of TFTP/FTP Server. Different TFTP/FTP Servers can be used for different routers/switches. If images are not stored in the root directory of the TFTP/FTP server, then also include the directory name here. **For example, substitute 192.168.56.1/ios_images in the above example where the IP address of the TFTP/FTP server is defined.**
5	Flash location to write new IOS image. Note, you **MUST** include the colon after the name. (e.g. flash0: disk0: flash:).
6	File in flash to delete. This file will only be deleted if the -del script option is used. If there is no file to be deleted, leave the two double quotes with no space between them ("").

The second list (***IOS_IMAGES***) contains the IOS images and attributes for those images. Any IOS image referenced in field 2, of RTR_LIST, must be defined in the IOS_IMAGES list. Below shows a sample entry for the IOS_IMAGES variable and the table below it describes each of the 3 fields in the row.

```
lappend IOS_IMAGES [list "c2900-universalk9-mz.SPA.153-2.T.bin"  "94293408"
 "ccf0684f8c189a2e8a51ea8ba3bd1653"]
```

Field	Description
1	Name of IOS image (e.g. **`c2900-universalk9-mz.SPA.153-2.T.bin)`**
2	File size of the image. This value is used when the script checks if there is enough free memory in flash. If you would like to make sure you have a little "extra" free flash instead of just enough, you can make this value larger then the actual value size. Do not make this value smaller than the actual file size.
3	MD5sum checksum of IOS image. Get this value from Cisco's website. After the IOS image is downloaded to the device, the md5sum verify command is run and the value from the router/switch is compared to this value.

Below shows the sample input file (***ios_upgrade_template.txt***) that is provided with the program and should be used as a template if creating your input files using a text editor. (Note, for MS Windows, the installation utility automatically copies this template file to the C:\Program Files\net-sense\templates directory. For Linux, the templates are in /usr/local/net-sense/templates).

```
#####################################################
# Script variables to change default settings
# default for md5 timeout is 90 seconds
# XFER_TIMEOUT_BETWEEN_BANGS is the maximum amount of time allowed
# between successive bangs (!...!) while the transfer is taking place.
# The default value for xfer_timeout is 30 seconds
# Uncomment out the values below to change settings
#####################################################
#set MD5_VERIFY_TIMEOUT 120
#set XFER_TIMEOUT_BETWEEN_BANGS 40

#########################################################
# Maximum number of simultaneous upgrades
# Default is 5.
# Uncomment the variable below to change value
#########################################################
# set MAX_SESSIONS 10

#################################################################
# If using ftp and the ftp username and password are not defined
# in the config file on the router/switch, then uncomment out the
# two lines below and put in the correct username and password
# and specify -ftp passwd script option when running the script
```

```
#################################################################
#set FTP_USERNAME jsmith
#set FTP_PASSWORD foobar

#########################################################################################################################
#                        Device                                                  Protocol      Server
#                          IP          New IOS File                            tftp/ftp      IP/dir
#########################################################################################################################
lappend RTR_LIST [list "ny-rtr1" "c2900-universalk9-mz.SPA.153-2.T.bin"    "tftp"   "192.168.56.1"
lappend RTR_LIST [list "ny-rtr2" "c2900-universalk9-mz.SPA.153-2.T.bin"    "tftp"   "192.168.56.1"
lappend RTR_LIST [list "ny-rtr3" "c2900-universalk9-mz.SPA.153-2.T.bin"    "tftp"   "192.168.56.1"
lappend RTR_LIST [list "ny-rtr4" "c2900-universalk9-mz.SPA.153-2.T.bin"    "tftp"   "192.168.56.1"
lappend RTR_LIST [list "tx-rtr1" "c2800nm-adventerprisek9-mz.151-4.M6.bin" "ftp"   "192.168.101.1/ios_image
lappend RTR_LIST [list "tx-rtr2" "c2800nm-adventerprisek9-mz.151-4.M6.bin" "ftp"   "192.168.101.1/ios_image
lappend RTR_LIST [list "tx-rtr3" "c2800nm-adventerprisek9-mz.151-4.M6.bin" "ftp"   "192.168.101.1/ios_image
lappend RTR_LIST [list "tx-rtr4" "c2800nm-adventerprisek9-mz.151-4.M6.bin" "ftp"   "192.168.101.1/ios_image


#########################################################################################################################
#                                                                          File
#                                IOS File Name                             Size              MD5 Sum
#########################################################################################################################
lappend IOS_IMAGES [list "c2900-universalk9-mz.SPA.153-2.T.bin"     "94293408"   "ccf0684f8c189a2e8a51ea8ba
lappend IOS_IMAGES [list "c2800nm-adventerprisek9-mz.151-4.M6.bin"  "67878324"   "a93e1de4f44899a89c7f885a6
```

Sample Command: The following command will upgrade the routers listed in the file ***ios_upgrade_group1.txt***. The script will first go into enable mode because of the ***-ual 2*** option. Before downloading the image to each IOS device, the script will delete the specified file in flash (***-del*** option). If the IOS download and verification checks are successful, then the script will update the config file with the new boot system statements and save the config fo NVRAM (***-config*** option). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
ios_upgrade -pw logins.txt -sf ios_upgrade_group1.txt -del -config -ual 2
```

5.3.6.1 IOS Upgrade GUI Configuration

Below shows the dialog window for defining the IOS devices to upgrade.

The IOS Images section must be defined first. Any number of IOS Image names along with the file size and MD5 checksum must be defined. The table below lists the devices to upgraded along with the New IOS image file to download. Moving the mouse button over any of the column headings, in the table below, will provide a tool tip type help. Information on each of the columns is defined below. Tool Tip help is available by moving the mouse in any of the entry fields for the ***IOS Images*** and ***IOS Upgrade Parameters*** sections. If using ftp and the ftp username and password are not defined in the config file on the router/switch, then select the check button box next to FTP Username and enter in the correct username and password. This FTP username/password will then be used; even if one is defined in the router/switch configuration file.

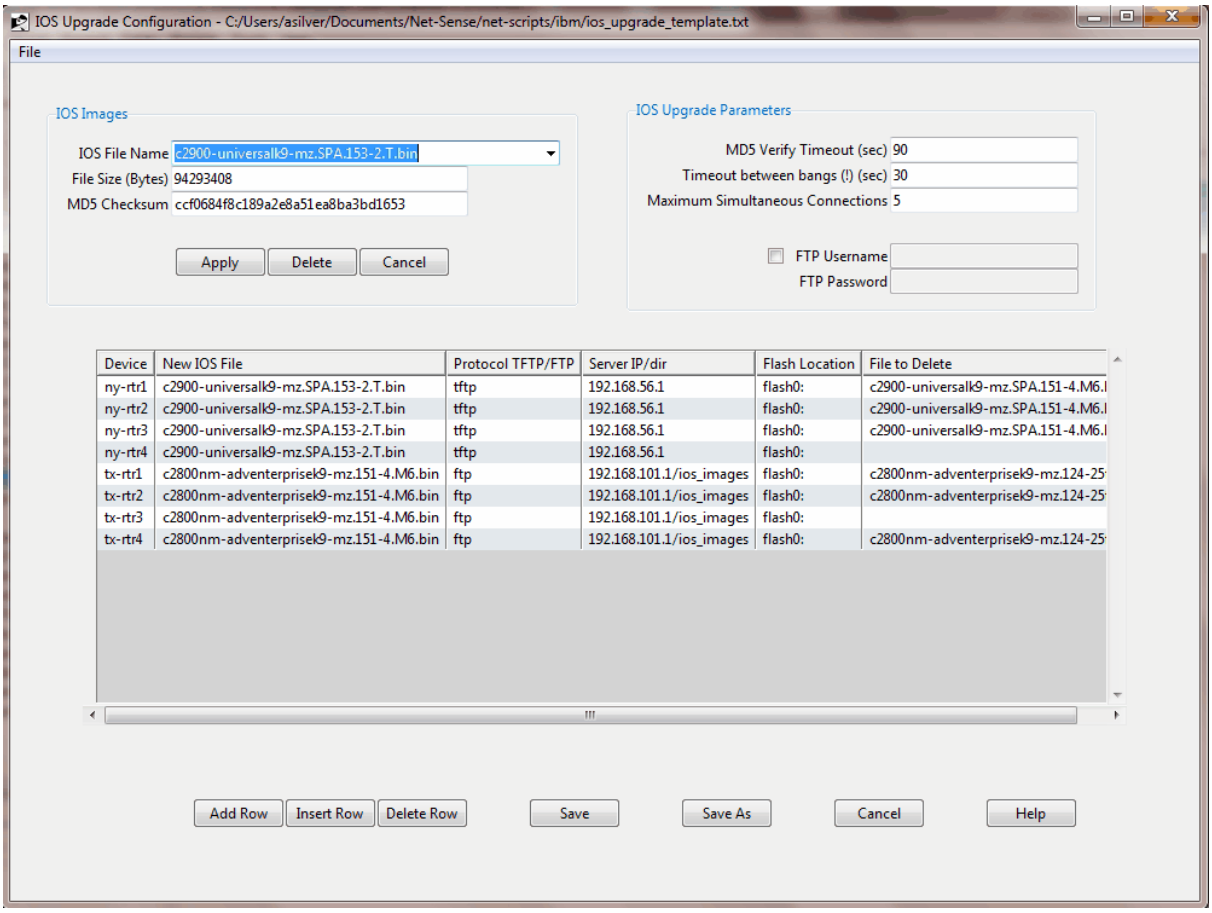


Table Column	Description
Device	IOS device that the script will telnet/ssh to. This must be an IP Address or a name that can be resolved through DNS. If the -pw <*password_file*> option is used, then this IP Address/Name must also be defined in the *password_file*. (Note, they must match exactly [case sensitive], unless the DEFAULT entry is present in the password file).
New IOS File	New IOS image that will be downloaded to the device (e.g. c2900-universalk9-mz.SPA.153-2.T.bin). Choose one of the images you have defined in the IOS Images section.
Protocol TFTP/FTP	Transfer Protocol to use. Currently only TFTP and FTP are supported. Choose one
Server IP/dir	IP Address of TFTP/FTP Server. Different TFTP/FTP Servers can be used for different routers/switches. If images are not stored in the root directory of the TFTP/FTP server, then also include the directory name here. **Ex: 192.168.56.1/ios_images**
Flash Location	Flash location to write new IOS image. Note, you **MUST** include the colon after the name. (e.g. flash0: disk0: flash:).
File to Delete	File in flash to delete. This file will only be deleted if the -del script option is used. If

	there is no file to be deleted, leave this field blank.

5.4 Cisco NX-OS Scripts

Cisco products running NX-OS (e.g. Nexus switches) have a slightly different CLI user interface than Cisco IOS devices. Therefore, new scripts have been created to work with NX-OS devices. The standard Cisco IOS based scripts will NOT work against NX-OS devices with one exception. The copy_to_tftp script will work for both Cisco IOS and Cisco NX-OS devices but you MUST use the -pw option with the script.

5.4.1 NX-OS Command Sender (Exec Level) (nxos_send_cmds)

This program will send any number of commands to a single Cisco Nexus switch or a list of NX-OS devices. The commands that are sent are defined in a text file (any name), that you create, or specified directly on the command line using the -cmd option. This script is ***different*** than the ***nxos_config_cmds*** script in that this script was primarily designed to send NX-OS "exec" level commands. In other words, non-configuration commands. The commands are sent in sequential order from top to bottom of the file or the order in which they are defined if using the -cmd options. This script can also send configuration commands but the actual commands to enter and exit configuration mode must also be included in the list of commands to send.

Note, for global Cisco configuration changes to NX-OS devices, the nxos_config_cmds script is the preferred tool as that script has more comprehensive error checking designed for configuration commands only.

Program Name: nxos_send_cmds

Script Argument	Description
-rf <filename>	List of switches or IP Address/(DNS hostname) to run script against (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-ipaddr <ip_address or routername>	IP address or switch name (if defined in hosts file or DNS) Use ***-ipaddr*** to send commands to a single device or ***-rf*** to send commands to list of devices specfied in file. (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-cf <filename>	File which contains a list of commands to send to switches. One command per line. Lines that begin with a "#" are considered comments and will not be sent to the router. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time!
-cmd <command>	Command to send to switch. Useful when only needing to send a few commands to a single device or list of devices. It saves you the time of having to create a text file with only a few commands in it. If the command contains spaces then it must be enclosed in double quotes (only when running script from cli, not GUI). This option can be used up to 7 times when used with the GUI. The command line version of this script can repeat this option an unlimited number of times. However, for more than 5 to 7 commands, it is recommended to put the commands in a text file and use the -cf option. (**REQUIRED -cf or -cmd**)

	NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time! Example: ***-cmd "show tech"***
-testmode	Test Mode. Build and display the commands that would have been sent to the devices but DO NOT actually connect to the devices and send the commands. Useful for confirming command looping and variable substitutions are correct or for just double checking the commands that will be sent . List of commands are also written to the file $SCRIPT_HOME/<script_name>_sample_cmds.txt". (**OPTIONAL**)
-wm	Saves configuration file to NVRAM after sending the commands. This option performs a "write memory" (i.e., copy running-config to startup-config) on the router. (**OPTIONAL**)
-dir <directory>	If specified, tells script to save output for each device into a separate file. Files will be saved to the directory specified. If the directory does not exist, the script will create it. By default output for all devices is only saved to the detailed trace file. The directory entered can be either an absolute directory or a relative directory. If running the script from the GUI, a relative directory is relative to the "SCRIPT_HOME" variable. (From the GUI see Options->Settings). (**OPTIONAL**)
-autodir <date | time>	Automatically create new unique directory to save output for each device into a separate file. The choices are ***date*** or ***time***. The ***date*** option will create a new directory consisting of just the date e.g. 08012010 for Aug 1st 2010. The ***time*** option will append the time to the date: e.g. 08012010_12h36m15s If used with the -dir option, the new unique directory will be created under the ***-dir directory***. If the -dir option is not used, then the new unique directory will be created under the ***SCRIPT_HOME*** directory. Note, if the ***date*** option is used and that directory name happens to all ready exist, then files in that directory will be overwritten. There are no safety prompts for the user when using this option.
-urfn	Use Route File Name. When saving the output data to a separate file for each device, the filename used will be the name/IP Address stated in the –rf file. The default filename is the router hostname configured on the router. This option only applies when used with the ***–dir*** option. (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw

	option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The command file (-cf <filename>) should contain a list of commands that will be sent to each router. This is not a TCL file, it is a plain text file that must contain one command per line. **Lines that begin with a "#" are considered comments and will not be sent to the router as a command.**

The following commands can be entered in the command file and the router will automatically answer any additional confirmations or prompts.

 (***Note, output of the commands listed below will NOT be displayed in the contents of individual output files when using the -dir option***)

Command	Description
reload	The Cisco NXOS ***reload*** command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to reload the box. Note, after entering the reload command, the device could respond with a message stating "***WARNING: There is unsaved configuration!!!"*** If this happens, the script will NOT reload the switch, it will log an error message and exit from the device. Note, do not use the reload command in the input file and the -wm option on the command line together. If you wish to perform a write mem before issuing a reload, then add the "copy running-config startup-config" to the input command file before the reload command. [This is because the -wm is performed after all of the commands in the input file are issued but the reload command will terminate the telnet session and the write mem (i.e., copy running-config startup-config) can no longer be performed]
copy running-config startup-config copy run startup	If the script sees any of these string as one of the commands to send, internally it will run a separate procedure that verifies the copy was successful. Note, the -wm command line option could also be used to save the configuration to NVRAM.
SLEEP <seconds>	The SLEEP command is not a command that will be sent to the switch. If desired, this is a method to introduce a delay between commands. Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The SLEEP command must be entered in all CAPITAL letters otherwise it will be interpreted as a command to send to the switch. It must also start at the beginning of the line (no leading spaces or tabs). Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command.
RETURN	Using the keyword RETURN is equivalent to sending just a carriage return or Enter key. Not commonly used. ***CASE***

	SENSITIVE
LOOPSTART <num> or LOOPEND	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will repeated are the commands between the LOOPSTART and LOOPEND commands. See Command Looping for more information.
COUNTERVAR <variable> <start_num>	Define a counter variable. Provides a method to define a variable in in the command file; it must be used with the LOOPING feature. The variable must be a single alphabet character. "start_num" is the number that the counter will start at. In will increment by one each time through the LOOP. See Counter Variables for more information.

The following is a sample command file which performs a show interface, then pauses for 15 seconds because the user wishes to visually examine the output of the show interface command (in real time). Then it clears the counters.

```
show interface
SLEEP 15
clear counters
```

If you wish the running config saved to NVRAM, then the **–wm** option must be used. By default, the configuration commands will not be saved to NVRAM.

This program also has the option to be run in *Safe* and *SuperSafe* mode which should be considered when running scripts in production environments.

Sample Command 1: The following command will send the commands listed in file ***show_cmds.txt*** to the NX-OS devices listed in the file ***lab_switches.txt.*** If there are any errors while issuing any of the commands, the script will immediately terminate because of the ***–ssafe*** option (SuperSafe Mode). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
nxos_send_cmds -pw logins.txt –rf lab_switches.txt –cf show_cmds.txt –ssafe
```

Sample Command 2: The following script will send a "show tech" to the switch 192.168.0.10. The output will be saved to the file show_tech_switch1.log The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
nxos_send_cmds -log show_tech_switch1.log -pw logins.txt –ipaddr 192.168.0.10 –cmd "show tech"
```

Sample Command 3: The following script will issue the command "show interface | incl error" 10 times, sleeping for 5 seconds between each time the command is entered. The commands will be sent to the switch 192.168.0.10. If there are any errors while issuing any of the commands, the script will immediately terminate because of the ***–safe*** option (Safe Mode). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

Note, the backslash at the end of each line will only work in a Linux/Unix shell.

```
nxos_send_cmds -pw logins.txt –ipaddr 192.168.0.10 -safe \
       -cmd "LOOPSTART 10" \
       -cmd "show int | incl error" \
       -cmd "SLEEP 5" -cmd LOOPEND
```

5.4.2 NX-OS Configuration Command Sender (nxos_config_cmds)

This program will send any number of **configuration** commands to a single Nexus switch or a list of switches running NX-OS. The commands that are sent are defined in a text file (any name), that you create, or specified directly on the command line using the -cmd option. This file MUST NOT include the commands needed to enter and exit configuration mode on the IOS device. The script will automatically enter those commands. The commands are sent in sequential order from top to bottom of the file or the order in which they are defined if using the -cmd options.

Program Name: nxos_config_cmds

Script Argument	Description
-rf <filename>	List of switches or IP Address/(DNS hostname) to run script against (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-ipaddr <ip_address or routername>	IP address or switch name (if defined in hosts file or DNS) Use ***-ipaddr*** to send commands to a single device or ***-rf*** to send commands to list of devices specfied in file. (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-cf <filename>	File which contains a list of commands to send to switches. One command per line. Lines that begin with a "#" are considered comments and will not be sent to the router. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time!
-cmd <command>	Command to send to switch. Useful when only needing to send a few commands to a single device or list of devices. It saves you the time of having to create a text file with only a few commands in it. If the command contains spaces then it must be enclosed in double quotes (only when running script from cli, not GUI). This option can be used up to 7 times when used with the GUI. The command line version of this script can repeat this option an unlimited number of times. However, for more than 5 to 7 commands, it is recommended to put the commands in a text file and use the -cf option. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time! Example: ***-cmd "show tech"***
-testmode	Test Mode. Build and display the commands that would have been sent to the devices but DO NOT actually connect to the devices and send the commands. Useful for confirming command looping and variable substitutions are correct or for just double checking the commands that will be sent . List of commands are also written to the file $SCRIPT_HOME/<script_name>_sample_cmds.txt". (**OPTIONAL**)
-wm	Saves configuration file to NVRAM after making the configuration

	changes. This option performs a “write mememory” on the router. (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with –pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The configuration command file (-cf <filename>) should contain a list of configuration commands that will be sent to each device. This is a plain text file that must contain one command per line. **Lines that begin with a “#” are considered comments and will not be sent to the router as a command.** Th **SLEEP** command is a special command that will **NOT** actually be sent to the router. If desired, this is a method to introduce a delay between configuration commands. The syntax is the **SLEEP** <***seconds***> (all capitals). Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The *SLEEP* command **MUST** be entered in all **CAPITAL** letters otherwise it will be interpreted as a command to send to the router. Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command.

By default, the configuration commands will not be saved to NVRAM. Use the ***–wm*** option to save the configuration changes.

This program also has the option to be run in ***Safe*** and ***SuperSafe*** mode which should be considered when running scripts in production environments. If entering a configuration command results in an error on the router and the script is running in ***Safe*** or ***SuperSafe*** mode, the config will not be saved to NVRAM even if the –wm option is applied. If the script is NOT running in ***Safe*** or ***SuperSafe*** mode, and the –wm option is applied, the config WILL be saved to NVRAM even if there are one or more commands that caused configuration errors.

The following special commands are supported by this script.


(***Note, output of the commands listed below will NOT be displayed in the contents of individual output files when using the -dir option***)

Command	Description
SLEEP <seconds>	The SLEEP command is not a command that will be sent to the router. If desired, this is a method to introduce a delay between commands. Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The SLEEP command must be entered in all CAPITAL letters otherwise it will be interpreted as a command to send to the router. It must also start at the beginning of the line (no leading spaces or tabs). Note, this is not typically needed but

	may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command.
LOOPSTART <num> or LOOPEND	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will repeated are the commands between the LOOPSTART and LOOPEND commands. See Command Looping for more information.
COUNTERVAR <variable> <start_num>	Define a counter variable. Provides a method to define a variable in in the command file; it must be used with the LOOPING feature. The variable must be a single alphabet character. "start_num" is the number that the counter will start at. In will increment by one each time through the LOOP. See Counter Variables for more information.

Sample Command 1: The following command will send the configuration commands listed in file ***snmp_cmds.txt*** to the NX-OS devices listed in the file ***lab_switches.txt.*** After the configuration commands are entered, the config will be saved to NVRAM because of the ***–wm*** option. If there are any errors while sending a particular configuration command, the script will abort all remaining commands to that device and continue on to the next device because of the ***–safe*** option (SAFE Mode). The detailed trace-log will be saved to a unique log file name instead of the default file name of ***nxos_config_cmds.log.*** The script will not prompt the user for passwords because the passwords are being read in from the ***logins.txt*** file (***-pw*** option).

```
nxos_config_cmds –ulog –pw logins.txt –rf lab_switches.txt –cf snmp_cmds.txt –wm –safe
```

Sample Command 2: The following script will shutdown interface eth 1/18 on switch 192.168.0.10***.*** The output log will be saved to an automatically generated unique filename because of the -ulog option. The script will not prompt the user for passwords because the passwords are being read in from the ***logins.txt*** file (***-pw*** option). At the first occurrence of an error, the script will abort because of the -safe option. If there are no errors, the config will be saved to NVRAM because of the -wm option.

```
nxos_config_cmds -ulog -pw logins.txt –ipaddr 192.168.0.10 –cmd "int eth 1/18" -cmd "shutdown"
```

5.5 Other Scripts

Enter topic text here.

5.5.1 Generic Command Sender

The generic command sender script was added to the suite of scripts to “handle” Non Cisco devices. The purpose of this script is to send a list of commands to any Non-Cisco device. The commands that are sent are defined in a text file (any name), that you create, or specified directly on the command line using the -cmd option. The commands are sent in sequential order from top to bottom of the file or the order in which they are defined if using the -cmd options. The primary feature of this script is the ability to login to many different types of CLI based devices. You, the user, define the login sequence and prompts with variables.

With the introduction of this script, two new ***device types*** for the login/password file were created; G1 and G2. G1 device types are devices that only prompt the user for a password when logging in. G2 device types are devices that prompt the user for a username and then a password when logging in. An example of a G1 type login is a Cisco router (when TACACS and Usernames are NOT configured). Here, you are just prompted for a password and then you are in the router. An example of a G2 type login is a Linux based system. Here, you are prompted for a Login Name and a Password.

If you are not using the login/password file, the G1/G2 convention does not apply; the script will prompt

you for the information needed.

Program Name: `generic_send_cmds`

Script Argument	Description
-rf <filename>	List of IP Address/(DNS hostname) to run script against (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-ipaddr <ip_address or devicename>	IP address or switch name (if defined in hosts file or DNS) Use ***-ipaddr*** to send commands to a single device or ***-rf*** to send commands to list of devices specfied in file. (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-cf <filename>	File which contains a list of commands to send to devices. One command per line. Lines that begin with a "#" are considered comments and will not be sent to the router. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time!
-cmd <command>	Command to send to device(s). Useful when only needing to send a few commands to a single device or list of devices. It saves you the time of having to create a text file with only a few commands in it. If the command contains spaces then it must be enclosed in double quotes (only when running script from cli, not GUI. This option can be used up to 7 times when used with the GUI. The command line version of this script can repeat this option an unlimited number of times. However, for more than 5 to 7 commands, it is recommended to put the commands in a text file and use the -cf option. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time! Example: ***-cmd "show version"***
-sf <filename>	This is the input file that defines the device prompts and other information about the "generic" device. (**REQUIRED**)
-ual (1 or 2)	User Access Level. Tells the script what access-level (1^{st} or 2^{nd}) to log into the device before sending the commands. For example, if you were sending commands to a UNIX system and the commands needed to be sent by "root", but you could not login remotely with the root username, then you would use a value of 2. The script would log you in with another Username and password and then su to root because of the "–ual 2" option. By default the script will only go into 1^{st} level access. (**OPTIONAL**)
-dir <directory>	If specified, tells script to save output for each device into a separate file. Files will be saved to the directory specified. If the directory does not exist, the script will create it. By default output for all devices is only saved to the detailed trace file. The directory entered can be either an absolute directory or a relative directory. If running the script

	from the GUI, a relative directory is relative to the "SCRIPT_HOME" variable. (From the GUI see Options->Settings). (**OPTIONAL**)
-autodir <date | time>	Automatically create new unique directory to save output for each device into a separate file. The choices are ***date*** or ***time***. The ***date*** option will create a new directory consisting of just the date e.g. 08012010 for Aug 1st 2010. The ***time*** option will append the time to the date: e.g. 08012010_12h36m15s If used with the -dir option, the new unique directory will be created under the ***-dir directory***. If the -dir option is not used, then the new unique directory will be created under the ***SCRIPT_HOME*** directory. Note, if the ***date*** option is used and that directory name happens to all ready exist, then files in that directory will be overwritten. There are no safety prompts for the user when using this option.
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The command file (-cf <filename>) should contain a list of commands that will be sent to each router. This is not a TCL file, it is a plain text file that must contain one command per line. **Lines that begin with a "#" are considered comments and will not be sent to the router as a command.**

The following commands can also be entered in the command file:

Command	Description
SLEEP <seconds>	The SLEEP command is not a command that will be sent to the switch. If desired, this is a method to introduce a delay between commands. Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The SLEEP command must be entered in all CAPITAL letters otherwise it will be interpreted as a command to send to the switch. Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command.

LOOPSTART <num> or LOOPEND	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will repeated are the commands between the LOOPSTART and LOOPEND commands. See Command Looping for more information.

There is also an option for setting the User Access Level. By default the script will only log into first level access on the router. If you are sending commands that require second level access, then use the –ual option and set it to 2 (e.g. –ual 2).

This program also has the option to be run in *Safe* and *SuperSafe* mode which should be considered when running scripts in production environments.

The device prompt definitions and other variables are defined in the input file (-sf <filename>). A sample input file is (***generic_send_cmds_template.txt***). If the end-user installation instructions were followed (Section3.2.2), this sample template file should be in the same directory where you run the scripts from. (Note, for MS Windows, the installation utility automatically copies this template file to the C:\Program Files\net-sense\userdata directory.)

The table below describes the variables in that file. The sequence, in which the variables are defined in the template file, should correspond to the actual order for when they are needed; when logging into a device. Although, the order of these variables is NOT significant, it probably helps to keep them in the order outlined in the template file.

Variable	Description
LOGIN_PROMPT	This is the prompt that the user sees when trying to login to a device and a Username/Login_name is required. Some examples include "Username:" or "Login:" (case sensitive). Note, this variable is not reference if a Username/Login_name is not used.
PASSWORD_PROMPT	The prompt that asks you for a first level password (case sensitive).
1ST_LEVEL_PROMPT	First level prompt character(s). This is the character(s) usually following the device name when in 1st level access mode. An example of a Cisco router is NY-router1> So the value of 1ST_LEVEL_PROMPT would be ">". If there is a "space" after this character, the space must be included! E.g. "> ".
TERM_LENGTH_ZERO_CMD	This is the command that will allow the device to send data to the user terminal without prompting the user to enter "return" or "space-bar" if more than one screen full of data is sent to the user terminal. For Cisco devices, the the command is "***terminal length 0***". If this does not apply to your devices, then set this value to an empty string. Example: set TERM_LENGTH_ZERO_CMD "".
2ND_LEVEL_ACCESS_CMD	The command that you would enter to go into a "priviledged" mode on the the device. For some devices this concept does not apply as there are multiple usernames/passwords for different levels of access. This variable will only be referenced if the "-ual 2" option is

	used on the command line when running the script. Example for a UNIX system would be “su”
2ND_LEVEL_PASSWORD_PROMPT	The prompt that asks you for the second level password. Again, this would only apply if the "-ual 2" option is used on the command line when running the script Eg: “Password: “
2ND_LEVEL_PROMPT	Second level prompt character(s). This is the character(s) usually following the device name when in 2nd level access mode. An example using a Cisco router is NY-router1# So the value of 2ND_LEVEL_PROMPT would be "#" Again, this would only apply if the "-ual 2" option is used on the command line when running the script. If there is a "space" after this character, the space must be included! E.g. "# "

ERROR_STRING	This is a character or string that will be displayed if an invalid command is sent to the device or the device rejects the command. The program will only look for this character/string starting at the beginning of a line. For example, on a Cisco device, if an errored command is entered, the error message will be written on a new-line beginning with the percent character (%).
	Cisco Example:
	`NY_router1>show junk`
	`^`
	`% Invalid input detected at '^' marker.`
	
	`NY_router1>`
	
	If you don't know what this error character/string is then make this value something you know will never come up when entering a command. For example, set it to "kdfjdkfjkj"
	set ERROR_STRING "kdfjdkfjkj"

Sample Command: The following command will send the commands listed in file ***show_cmds.cmds*** to the devices listed in the file ***east_coast.rt.*** If there are any errors while issuing any of the commands, the script will immediately terminate because of the ***–ssafe*** option (SuperSafe Mode) (Assuming the “ERROR_STRING” variable is correctly defined). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
generic_send_cmds -pw logins.txt -rf east_coast.rt -cf show_cmds.cmds -ssafe
```

5.5.1.1 Template File

Template File Location:
Windows: C:\Program Files (x86)\Net-Sense\generic_send_cmds_template.txt
Linux: /usr/local/net-sense/templates/generic_send_cmds_template.txt

```
#############################################################
# Template file for generic_send_cmds script
#############################################################

#############################################################################
# This is the prompt that the user sees when trying to
# login to a device and a Username/Login_name is required.
# Some examples include "Username:" or "Login:" (case sensitive)
# Note, this variable is not reference if a Username/login_name is not
# used
#############################################################################
set LOGIN_PROMPT "Login"

#############################################################################
# The prompt that asks you for a first level password (case sensitive)
#############################################################################
set PASSWORD_PROMPT "Password:"

#############################################################################
# First level prompt character(s).  This is the character(s) usually
# following the device name when in 1st level access mode.  An example
# of a Cisco router is
# NY-router1>
# So the value of 1ST_LEVEL_PROMPT would be ">"
# If there is a "space" after this character, the space must be included!
# E.g. "# "
#############################################################################
set 1ST_LEVEL_PROMPT ">"

#############################################################################
# This is the command that will allow the device to send
# data to the user terminal without prompting the user to enter "return"
# or "space-bar" if more than one screen full of data is sent to the
# user terminal. For Cisco devices, the the command
# is "terminal length 0".
# If this does not apply to your devices, then set this value to an empty
# string. Example:
# set TERM_LENGTH_ZERO_CMD ""
#############################################################################
set TERM_LENGTH_ZERO_CMD "term len 0"

#############################################################################
# The command that you would enter to go into a "priviledged" mode on the
# the device.  For some devices this concept does not apply as there
# are multiple usernames/passwords for different levels of access
# This variable will only be referenced if the "-ual 2" option is used
# on the command line when running the script
#############################################################################
set 2ND_LEVEL_ACCESS_CMD "enable"

#############################################################################
# The prompt that asks you for the second level password.  Again, this
# would only apply if the "-ual 2" option is used on the command line
# when running the script
#############################################################################
set 2ND_LEVEL_PASSWORD_PROMPT "Password: "

#############################################################################
# Second level prompt character(s).  This is the character(s) usually
# following the device name when in 2nd level access mode.  An example
# of a Cisco router is
# NY-router1#
# So the value of 2ND_LEVEL_PROMPT would be "#"
# Again, this would only apply if the "-ual 2" option is used on
# the command line when running the script
# If there is a "space" after this character, the space must be included!
```

```
# E.g. "# "
#########################################################################
set 2ND_LEVEL_PROMPT "# "

#########################################################################
# Exit Command for this device.  This is the command that will immediately
# log you off of the device.
# Examples: exit, quit, logoff
# If the variable EXIT_CMD is not defined, the default EXIT_CMD is "exit"
#########################################################################
set EXIT_CMD "exit"

#########################################################################
# This is a character or string that will be displayed if an invalid
# command is sent to the device or the device rejects the command
# The program will only look for this character/string starting
# at the beginning of a line.
# For example, on a Cisco device, if an errored command is entered, the
# error message will be written on a new-line beginngin with the
# percent character (%).
# Cisco Example:
# NY_router1>show junk
#            ^
# % Invalid input detected at '^' marker.
#
# us-ce3>

#
# If you don't know what this error character/string is then make
# this value something you know will never come up when entering
# a command. For example, set it to "kdfjdkfjkj"
# set ERROR_STRING "kdfjdkfjkj"
#########################################################################
set ERROR_STRING "%"
```

5.5.2 APC Command Sender

This program will send any number of commands to a list of APC devices. Since the APC device has a menu driven interface, the commands are primarily numbers that correspond to a particular action. The commands that are sent are defined in a text file (any name), that you create. The commands are sent in sequential order from top to bottom of the file. Do not add the very last command to log off the router, the script will automatically handle that.

Program Name: apc_send_cmds

Script Argument	Description
-rf <filename>	List of IP Address to run script against (**REQUIRED**)
-cf <filename>	File which contains a list of commands to send to APC devices. One command per line. Lines that begin with a "#" are considered comments and will not be sent to the router. (**REQUIRED**)
-dir <directory>	If specified, tells script to save output for each device into a separate file. Files will be saved to the directory specified. If the directory does not exist, the script will create it. By default output for all devices is only saved to the detailed trace file. The directory entered can be either an absolute directory or a relative directory. If running the script from the GUI, a relative directory is relative to the "SCRIPT_HOME" variable. (From the GUI see Options->Settings). (**OPTIONAL**)
-autodir <date | time>	Automatically create new unique directory to save output for each device

	into a separate file. The choices are ***date*** or ***time***. (**OPTIONAL**) The ***date*** option will create a new directory consisting of just the date e.g. 08012010 for Aug 1st 2010. The ***time*** option will append the time to the date: e.g. 08012010_12h36m15s If used with the -dir option, the new unique directory will be created under the ***-dir directory***. If the -dir option is not used, then the new unique directory will be created under the ***SCRIPT_HOME*** directory. Note, if the ***date*** option is used and that directory name happens to all ready exist, then files in that directory will be overwritten. There are no safety prompts for the user when using this option.
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The command file (-cf <filename>) should contain a list of commands that will be sent to each router. This is not a TCL file, it is a plain text file that must contain one command per line. **Lines that begin with a "#" are considered comments and will not be sent to the router as a command.** For the APC device there are two special cases where the command in the text file is followed by a double colon (::) and then another string. See the table below for more information.

The following table lists special commands can be entered in the command file. See command looping and counter variables for more information on these.


(***Note, output of the commands listed below will NOT be displayed in the contents of individual output files when using the -dir option***)

Command	Description
cmd::string	Send the command but don't look for the standard prompt for the APC device. Instead, for after sending this one command only, the value of string will be the prompt the script waits to see. This is necessary in the cases where you enter a command <number> and then the APC device prompts the user to enter in a new value for something (e.g. snmp community string, new syslog server, new ntp server, etc.)

cmd::ACCEPT	Send the command and then after the command is sent, verify that the command was successful. This should be done when performing configuration changes such as change the snmp community string. The script will specifically be looking for the string : Success in the display after the command is sent.
ESC	Send the escape character. Must be all capitals.
SLEEP <seconds>	The SLEEP command is not a command that will be sent to the router. If desired, this is a method to introduce a delay between commands. Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The SLEEP command must be entered in all CAPITAL letters otherwise it will be interpreted as a command to send to the router. It must also start at the beginning of the line (no leading spaces or tabs). Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command.
LOOPSTART <num> or LOOPEND	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will repeated are the commands between the LOOPSTART and LOOPEND commands. See Command Looping for more information.
COUNTERVAR <variable> <start_num>	Define a counter variable. Provides a method to define a variable in in the command file; it must be used with the LOOPING feature. The variable must be a single alphabet character. "start_num" is the number that the counter will start at. In will increment by one each time through the LOOP. See Counter Variables for more information.

The following is a sample command file which changes the ntp server value.

```
########################################
# Commands with Special Meaning
# ESC - Send Escape character
# SLEEP <seconds>
#    e.g. SLEEP 5
3
3
2
################################################
# Field Seperator is :: (two colons)
# If the second field does not contain the word
# ACCEPT, then it is the prompt that should be seen
# after that command is entered
################################################
1::Primary NTP Server :
1.1.1.4
########################################################
# If we see ACCEPT in the second field then we need to
# check that the following is found in
# the buffer
# Accept Changes                  : Success
########################################################
6::ACCEPT
ESC
ESC
ESC
```

This program also has the option to be run in *Safe* and *SuperSafe* mode which should be considered when running scripts in production environments.

Sample Command: The following command will send the commands listed in file new_ntp_server.txt to the APC devices listed in the file apc_devices.txt. If there are any errors while issuing any of the commands, the script will immediately terminate because of the ***–ssafe*** option (SuperSafe Mode). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option). A log file will automatically be created with a unique file name that will not get overwritten (***-ulog***)

```
apc_send_cmds -ulog -pw logins.txt –rf apc_devices.txt –cf new_ntp_server.txt –ssafe
```

5.5.3 CatOS Command Sender (catos_send_cmds)

This program will send any number of commands to a single Catalyst switch running CatOS or a list of switches. The commands that are sent are defined in a text file (any name), that you create, or specified directly on the command line using the -cmd option. The commands are sent in sequential order from top to bottom of the file or the order in which they are defined if using the -cmd options.

Program Name: catos_send_cmds

Script Argument	Description
-rf <filename>	List of switches or IP Address/((DNS hostname) to run script against (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-ipaddr <ip_address or devicename>	IP address or switch name (if defined in hosts file or DNS) Use ***-ipaddr*** to send commands to a single device or ***-rf*** to send commands to list of devices specfied in file. (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-cf <filename>	File which contains a list of commands to send to switches. One command per line. Lines that begin with a "#" are considered comments and will not be sent to the router. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time!
-cmd <command>	Command to send to switch. Useful when only needing to send a few commands to a single device or list of devices. It saves you the time of having to create a text file with only a few commands in it. If the command contains spaces then it must be enclosed in double quotes (only when running script from cli, not GUI). This option can be used up to 7 times when used with the GUI. The command line version of this script can repeat this option an unlimited number of times. However, for more than 5 to 7 commands, it is recommended to put the commands in a text file and use the -cf option. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time!

	Example: ***-cmd "show tech"***
-testmode	Test Mode. Build and display the commands that would have been sent to the devices but DO NOT actually connect to the devices and send the commands. Useful for confirming command looping and variable substitutions are correct or for just double checking the commands that will be sent . List of commands are also written to the file $SCRIPT_HOME/<script_name>_sample_cmds.txt". (**OPTIONAL**)
-ual (1 or 2)	User Access Level. Tells the script what access-level (1^{st} or 2^{nd}) to log into the switch before sending the commands. By default the script will only go into 1^{st} level access. (**OPTIONAL**)
-dir <directory>	If specified, tells script to save output for each device into a separate file. Files will be saved to the directory specified. If the directory does not exist, the script will create it. By default output for all devices is only saved to the detailed trace file. The directory entered can be either an absolute directory or a relative directory. If running the script from the GUI, a relative directory is relative to the "SCRIPT_HOME" variable. (From the GUI see Options->Settings). (**OPTIONAL**)
-autodir <date | time>	Automatically create new unique directory to save output for each device into a separate file. The choices are ***date*** or ***time***. The ***date*** option will create a new directory consisting of just the date e.g. 08012010 for Aug 1st 2010. The ***time*** option will append the time to the date: e.g. 08012010_12h36m15s If used with the -dir option, the new unique directory will be created under the ***-dir directory***. If the -dir option is not used, then the new unique directory will be created under the ***SCRIPT_HOME*** directory. Note, if the ***date*** option is used and that directory name happens to all ready exist, then files in that directory will be overwritten. There are no safety prompts for the user when using this option.
-urfn	Use Route File Name. When saving the output data to a separate file for each device, the filename used will be the name/IP Address stated in the –rf file. The default filename is the switch hostname configured on the switch. This option only applies when used with the ***–dir*** option. (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a configuration command to a switch in the list, all subsequent commands to that switch will not be sent. The script will continue on to the next switch on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the switchs. (**OPTIONAL**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing switchs. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)

-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The command file (-cf <filename>) should contain a list of commands that will be sent to each CatOS Switch. This is not a TCL file, it is a plain text file that must contain one command per line. **Lines that begin with a "#" are considered comments and will not be sent to the switch as a command.**

The following command can also be entered in the command file:

Command	Description
clear counters clear counter	The clear counters command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to clear the counters.
SLEEP <seconds>	The SLEEP command is not a command that will be sent to the switch. If desired, this is a method to introduce a delay between commands. Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The SLEEP command must be entered in all CAPITAL letters otherwise it will be interpreted as a command to send to the switch. It must also start at the beginning of the line (no leading spaces or tabs). Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command.
LOOPSTART <num> or LOOPEND	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will repeated are the commands between the LOOPSTART and LOOPEND commands. See Command Looping for more information.

The following is a sample command file which performs a show vlan, show port, then pauses for 15 seconds because the user wishes to visually examine the output of the show port command (in real time). Then it performs a show snmp.

```
show vlan
show port
SLEEP 15
show snmp
```

There is also an option for setting the User Access Level. By default the script will only log into first level access on the switch. If you are sending commands that require second level access, then use the –ual option and set it to 2 (e.g. –ual 2).

This program also has the option to be run in *Safe* and *SuperSafe* mode which should be considered when running scripts in production environments.

Sample Command 1: The following command will send the commands listed in file ***show_cmds.cmds*** to the switches listed in the file ***switches.rt.*** The script will go into enable mode before issuing the commands because of the ***–ual 2*** option. If there are any errors while issuing any of the commands, the script will immediately terminate because of the ***–ssafe*** option (SuperSafe Mode). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
catos_send_cmds -pw logins.txt -rf switches.rt -cf show_cmds.cmds -ssafe -ual 2
```

Sample Command 2: The following script will send a "show tech" to the switch 192.168.0.10. The output will be saved to the file show_tech_switch1.log The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
catos_send_cmds -log show_tech_switch1.log -pw logins.txt -ipaddr 192.168.0.10 -cmd "show tech"
```

5.5.4 PIX/ASA Firewall Command Sender (pix_send_cmds)

This program will send any number of commands to a single PIX/ASA Firewall or a list of PIX/ASA devices. The commands that are sent are defined in a text file (any name), that you create, or specified directly on the command line using the -cmd option. The commands are sent in sequential order from top to bottom of the file or the order in which they are defined if using the -cmd options. When using this script to send configuration commands, the actual commands to enter (config t) and exit (end) configuration mode must also be included in the list of commands to send.

(**It is highly recommended that you test this on non-production PIXs/ASAs before attempting configuration changes on production firewalls**.)

Program Name: pix_send_cmds

Script Argument	Description
-rf <filename>	List of PIXs/ASAs or IP Address/(DNS hostname) to run script against (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-ipaddr <ip_address or devicename>	IP address or PIX/ASA name (if defined in hosts file or DNS) Use ***-ipaddr*** to send commands to a single device or ***-rf*** to send commands to list of devices specfied in file. (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-cf <filename>	File which contains a list of commands to send to PIX/ASA. One command per line. Lines that begin with a "#" are considered comments and will not be sent to the router. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time!
-cmd <command>	Command to send to PIX/ASA. Useful when only needing to send a few commands to a single device or list of devices. It saves you the time of having to create a text file with only a few commands in it. If the command contains spaces then it must be enclosed in double quotes (only when running script from cli, not GUI. This option can be used up to 7 times when used with the GUI. The command line version of this script can repeat this option an unlimited number of times. However, for more than 5 to 7 commands, it is recommended to put the commands in a text file and use the -cf option. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time!

	Example: ***-cmd "show tech"***
-testmode	Test Mode. Build and display the commands that would have been sent to the devices but DO NOT actually connect to the devices and send the commands. Useful for confirming command looping and variable substitutions are correct or for just double checking the commands that will be sent . List of commands are also written to the file $SCRIPT_HOME/<script_name>_sample_cmds.txt". (**OPTIONAL**)
-wm	Saves configuration file to NVRAM after sending the commands. This would only make sense if any of the commands were configuration commands or if you just wanted to "make sure" that the configs are saved to NVRAM. This option performs a "write memory" on the PIX. (**OPTIONAL**)
-ual (1 or 2)	User Access Level. Tells the script what access-level (1^{st} or 2^{nd}) to log into the PIX before sending the commands. By default the script will only go into 1^{st} level access. (**OPTIONAL**)
-dir <directory>	If specified, tells script to save output for each device into a separate file. Files will be saved to the directory specified. If the directory does not exist, the script will create it. By default output for all devices is only saved to the detailed trace file. The directory entered can be either an absolute directory or a relative directory. If running the script from the GUI, a relative directory is relative to the "SCRIPT_HOME" variable. (From the GUI see Options->Settings). (**OPTIONAL**)
-autodir <date | time>	Automatically create new unique directory to save output for each device into a separate file. The choices are ***date*** or ***time***. The ***date*** option will create a new directory consisting of just the date e.g. 08012010 for Aug 1st 2010. The ***time*** option will append the time to the date: e.g. 08012010_12h36m15s If used with the -dir option, the new unique directory will be created under the ***-dir directory***. If the -dir option is not used, then the new unique directory will be created under the ***SCRIPT_HOME*** directory. Note, if the ***date*** option is used and that directory name happens to all ready exist, then files in that directory will be overwritten. There are no safety prompts for the user when using this option.
-urfn	Use Route File Name. When saving the output data to a separate file for each device, the filename used will be the name/IP Address stated in the –rf file. The default filename is the switch hostname configured on the switch. This option only applies when used with the ***–dir*** option. (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a command to a PIX firewall in the list, all subsequent commands to that PIX will not be sent. The script will continue on to the next PIX on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the commands to any of the PIXs. (**OPTIONAL**)

-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing PIXs. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The command file (-cf <filename>) should contain a list of commands that will be sent to each PIX firewall. This is not a TCL file, it is a plain text file that must contain one command per line. **Lines that begin with a "#" are considered comments and will not be sent to the switch as a command.**

The following command can also be entered in the command file:

Command	Description
clear counters clear counter	The clear counters command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to clear the counters.
clear logging clear log	The clear logging (or clear log) command requires confirmation after entering the command in the CLI. The script will automatically send the confirmation to clear the log.
write mem wr mem	If the script sees this string as one of the commands to send, internally it will run a separate procedure that is looking for the string [OK]. If it doesn't see this, it will log an error.
SLEEP <seconds>	The SLEEP command is not a command that will be sent to the switch. If desired, this is a method to introduce a delay between commands. Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The SLEEP command must be entered in all CAPITAL letters otherwise it will be interpreted as a command to send to the switch. It must also start at the beginning of the line (no leading spaces or tabs). Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command.
LOOPSTART <num> or LOOPEND	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will repeated are the commands between the LOOPSTART and LOOPEND commands. See Command Looping for more information.

The following is a sample command file which performs a show version, and show run.

```
show version
show run
```

There is also an option for setting the User Access Level. By default the script will only log into first level access on the switch. If you are sending commands that require second level access, then use the –ual option and set it to 2 (e.g. –ual 2).

This program also has the option to be run in *Safe* and *SuperSafe* mode which should be considered when running scripts in production environments.

Sample Command: The following command will send the commands listed in file ***show_cmds.cmds*** to the switches listed in the file ***pixs.rt.*** The script will go into enable mode before issuing the commands because of the ***–ual 2*** option. If there are any errors while issuing any of the commands, the script will immediately terminate because of the ***–ssafe*** option (SuperSafe Mode). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
pix_send_cmds -pw logins.txt -rf pixs.rt -cf show_cmds.cmds -ssafe -ual 2
```

Sample Command 2: The following script will send a "show tech" to the switch 192.168.0.10**.** The output will be saved to the file show_tech_switch1.log The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
catos_send_cmds -log show_tech_switch1.log -pw logins.txt -ipaddr 192.168.0.10 -cmd "show tech"
```

5.6 Sample Template Files

Several of the scripts require input files which define parameters/variables for a given script run. The exact input parameters differ depending on the script. This section contains templates for those input files. When creating your own input files, use the template file as a starting point and then save the file to a new file name. Copies of the template files are also installed with the software in the following locations (assuming default installation directories)

Windows:

C:\Program Files\Net-Sense\templates
C:\Program Files\Net-Sense\userdata

Unix/Linux:

/usr/local/net-sense/templates
$HOME/net-scripts

5.6.1 Login Password Template File

```
##############################
# Copyright 2002 NetSense Inc.
# www.net-sense.com
# Author: Allan Silverstein
##############################

#########################################################################
# This is a template file for login/password information for routers
# Lines that begin with a # are considered comments
#########################################################################
# The login/password information is entered in a table like format.
# Each row in the table has 8 fields.  Each field should be enclosed
# in double-quotest ("").
# If a field, such as "Username" does not apply, just put two double
# quotes with nothing in the middle. (e.g.  "")
#
# A decription of each field
# is as follows:
#
# 1.  Router Type: is C=Cisco, E=Efficient, G1=Generic1, G2=Generic2, N=Nortel, CAT=CatOS Switch
# 2.  Router IP: IP Address or Name. This must match the entry in the -rf <filename> exactly!!
# 3.  1st-level username: Used for TACACS, SSH,  or if router is setup with local Usernames
#     If using ssh, this is the login account ssh will use to login.  For ssh on Redhat Linux,
#     this is the argument to the -l flag.
```

```
# 4.  Passwd1: 1st-level password for Cisco Router (vty password or password for 1st-level Username)
# 5.  2nd-level Username: This would only apply if the router prompts you for a Username
#                       after you type in "enable"
# 6.  Passwd2: enable password for Cisco.  This is entered after typing in the word "enable" on the router
# 7.  Access Proram: Currently the only two choices are telnet or secure shell (ssh)
# 8.  spawn_id: This MUST be left blank.
##############################################################################
# For Efficient Routers, set the Router Type to E and only configure the
# Passwd1 field.  All other fields are ignored.  They can be two double quotes (i.e. "")
# or left with values, but those values will still be ignored.  Note, the spawn_id for
# Efficient Routers must still be left blank.
##############################################################################################################
# G1 device types are generic devices that only prompt the user for a Password.  There
# is no Login Name associated with the Password
##############################################################################################################
# G2 device types are generic devices that prompt the user for both a Login Name AND Password.
##############################################################################################################
# P device types are Cisco PIX Firewalls
##############################################################################################################
# For Nortel Routers, set the Router Type to N and only configure the
# "1st-level Username" and the Passwd1 field.  All other fields are ignored.
# They can be two double quotes (i.e. "") or left with values, although they are
# ignored they must still be present.  Note, the spawn_id for
# Nortel Routers must still be left blank.
# A possible 1st-level Username for Nortel routers is "Manager".
##############################################################################################################
# The special DEFAULT entry will be used for passwords if the router/device is not explicitly
# defined in field 2 (Router IP).  This DEFAULT entry MUST be the last row in the table or
# it will not work.  Note, a DEFAULT entry does not need to be defined, but is useful if you have
# many routers/devices and they all use the same passwords.  Thus you don't have to explicitly
# define each router/device in the table.
# DON'T FORGET TO CHANGE THE "ROUTER TYPE" FIELD, for the DEFAULT entry, to
# the device type you are using (e.g. C, E, G1, G2, N)  The default value is C for Cisco.
#
##############################################################################################################
#    "Router Type" "Router IP" "1st-level Username" "Passwd1" "2nd-level username "Passwd2" "Access Program
##############################################################################################################
set ALL_ROUTERS ""
lappend ALL_ROUTERS [list C "10.10.1.1" "" "foobar" "" "foobar" "telnet" "" ] ;# SanFran 1720
lappend ALL_ROUTERS [list C "nyrtr" "allan" "mypasswd" "" "my2ndpasswd" "telnet" "" ] ;# NY router
lappend ALL_ROUTERS [list C "10.10.2.1" "" "foobar" "" "foobar" "ssh" "" ] ;# LA 7500
lappend ALL_ROUTERS [list C "M2" "" "foobar" "" "foobar" "ssh" "" ] ;# MI 3660
lappend ALL_ROUTERS [list C "10.10.9.1" "" "foobar" "" "foobar" "telnet" "" ] ;# FL 4500
lappend ALL_ROUTERS [list C "10.10.17.1" "" "foobar" "" "foobar" "telnet" "" ] ;# NJ 12000
lappend ALL_ROUTERS [list G1 "nj-test3" "" "foobar" "" "foobar" "telnet" "" ] ;# abc box
lappend ALL_ROUTERS [list G2 "ny-fore2" "ami" "foobar" "" "foobar" "telnet" "" ] ;# Fore Switch
lappend ALL_ROUTERS [list P "ny-pix1" "" "foobar" "" "foobar" "telnet" "" ] ;# PIX Firewall
lappend ALL_ROUTERS [list N "nortel_nj_2" "Manager" "foobar" "" "foobar" "telnet" "" ] ;# Nortel Router
lappend ALL_ROUTERS [list C "DEFAULT" "" "abcde" "" "ddffgg" "telnet" "" ] ;# Default password
```

5.7 Special Script Features

This section describes additional features some of the scripts contain.

5.7.1 Command Looping

New with version 4.3.2 is the ability to send the same command(s) over and over without having to repeat those same commands over and over in the command input file. This is accomplished through the use of two keywords in the command file; LOOPSTART <num> and LOOPEND (all capital letters). All of the commands in between the LOOPSTART and LOOPEND commands will be repeated the number of times specified by the LOOPSTART command. The scripts that support this feature are

cisco_send_cmds
cisco_send_cmds_rcf
cisco_config_cmds

cisco_config_cmds_rcf
pix_send_cmds
catos_send_cmds
generic_send_cmds
nxos_send_cmds
nxos_config_cmds

The best way to describe this is with an example. The example below will issue the ***show version*** command, then the ***show clock*** command. Then it will enter the ***show proc cpu*** and ***show memory stat*** commands **100** times (note there will be a 5 second delay between each iteration of the command being entered because of the ***SLEEP 5*** command). After the loop ends, the ***show clock*** command will be entered once and the script is finished sending commands to that device.

```
show version
show clock
LOOPSTART 100
show proc cpu
show memory stat
SLEEP 5
LOOPEND
show clock
```

 Note: Nested Loops are not supported.

5.7.2 Counter Variables

New with version 4.5 is the ability to include commands, which are variables, that increment while inside a command loop (i.e., LOOPSTART/LOOPEND). This is accomplished through the use of the keyword COUNTERVAR (all capitals) in the command file. Multiple counter variables can be defined in one command file. When the variable is referenced inside the command LOOP, it must be preceded with a $. The amount the variable will increment by, each time through the loop, is defined by the increment_by parameter. The exact format is:

COUNTERVAR <*variable*> <*start_integer*> <*increment_by*>

variable - must be a single alphabet character
start_integer - start count
increment_by - amount incremented each time through the loop

The scripts that support this feature are:

cisco_send_cmds
cisco_send_cmds_rcf
cisco_config_cmds
cisco_config_cmds_rcf
pix_send_cmds
catos_send_cmds
generic_send_cmds
nxos_send_cmds
nxos_config_cmds

When used with the cisco_config_cmds script, the command file below will create 150 sub-interfaces along with some interface sub-commands. There are 2 counter variables defined: X and Y. X will start at the number 500 and Y will start at the number 1. Each time through, the loop each counter will increment by 1. Wherever $X and $Y are referenced inside the command loop, the current value of that variable will substituted.

After the sub-interfaces are created, the script will perform a "do show run". Note, the key word "do" is needed because this example command file was written to be used by the cisco_config_cmds script.

```
COUNTERVAR X 500 1
COUNTERVAR Y 1 1

LOOPSTART 150

interface FastEthernet 0/5/0.$X
encapsulation dot1Q $X
ip address 10.10.$Y.1 255.255.255.0

LOOPEND

do show run
```

 Note: Nested Loops are not supported.

 Note, if running the Linux command line version of any of the scripts below AND using the -cmd option with the COUNTERVAR feature, any occurrences of $ must be preceded with a backslash "\". This is because Linux shell substitution will occur on the command line if the backslash is omitted. This does not apply when running the scripts from the GUI or from the ***Microsoft Windows*** cmd shell.

cisco_config_cmds
cisco_send_cmds
nxos_config_cmds
nxos_send_cmds
pix_send_cmds
generic_send_cmds

Example:
cisco_config_cmds -ip 10.1.1.1 -cmd "COUNTERVAR A 100 1" -cmd "LOOPSTART 10" -cmd "vlan \$A" -cmd LOOPEND

5.7.3 Adding Comments to Log Files

In some cases it may be helpful to have comments interleaved into the detailed trace log file. For example, during lab testing, you may be sending many commands to a device and plan to review the detailed log file after the script completes. However, the exact output your expecting to see from each of the commands you are sending may not be clear. This is accomplished by putting the keyword COMMENT (ALL CAPS) followed by your comment, in the command file. There MUST NOT be any spaces before the word COMMENT. The following scripts support this feature:

cisco_send_cmds
cisco_send_cmds_rcf
pix_send_cmds
catos_send_cmds
generic_send_cmds
nxos_send_cmds

Below is a sample command file (used for the cisco_send_cmds script) showing the COMMENT feature as well as the SLEEP feature.

```
clear log

COMMENT: Baseline Commands before any clears
show ip eigrp neighbors
show ip eigrp neighbors detail
COMMENT: End of Baseline Commands

COMMENT: Start Command Test
clear ip eigrp neighbors 10.1.1.1
show ip eigrp neighbors
COMMENT: Wait 30 seconds
SLEEP 30
COMMENT: Only neighbor 10.1.1.1 should be reset with an new uptime
show ip eigrp neighbors

COMMENT: Start Command Test
clear ip eigrp neighbors
show ip eigrp neighbors
COMMENT: Wait 30 seconds
SLEEP 30
COMMENT: All neighbors should be up with a new uptime
show ip eigrp neighbors
```

5.7.4 Handling Interactive Type Commands

Typically, the command sender type scripts (e.g, cisco_send_cmds, pix_send-cmds, generic_send_cmds, etc..) read in a list of commands to send to the device and send them one at a time. After each command is sent, the script is expecting to see a particular prompt string. However, there are some commands that are "interactive" where the default prompt string is not displayed, or some type of confirmation prompt is presented by the device. In some cases, the scripts all ready have built in functionality to support these interactive type commands on IOS devices (e.g., clear log) and this feature is not needed. In other cases, the scripts do not have built in support to handle a particular interactive command. See the documentation page for each individual script for the built in, interactive, commands supported by each script.

For cases where there is not built in support for an interactive command, there is a special syntax in the command file which allows you to define a custom prompt for an individual command. **Note, this feature is only available with the AutomaterPro License.**

The syntax of the command is

<command> ::PROMPT "prompt"

where <command> is the command to send to the device and "prompt" is the prompt you are expecting to receive back from the device.

Going one step further you can even verify output for these types of commands using the additional CHECKFOR option (see example below)

<command> ::CHECKFOR "string" ::PROMPT "prompt"

Below is an example of manually saving a config to a cisco wireless LAN controller.

```
(WiSM-slot4-2) >save config

Are you sure you want to save? (y/n) y

Configuration Saved!

(WiSM-slot4-2) >
```

Below is the command file used to save the config of a cisco wireless LAN controller using the generic_send_cmds script. Note this is NOT and IOS type device, therefore we are using the generic_send_cmds script. Note, it is possible that this command file will not report a problem if the save config command failed.

```
save config ::PROMPT "(y/n) "
y
```

If we want to confirm that the device responded with "Configuration Saved" before the prompt returned, the file would look like this. Note, we did not need the additional ::PROMPT feature on the second line because the prompt that is expected after entering "y", is the default prompt for this box.

```
save config ::PROMPT "(y/n) "
y ::CHECKFOR "Configuration Saved"
```

if we wanted we could have also added the additional prompt keyword on the second line as follows:

```
save config ::PROMPT "(y/n) "
y ::CHECKFOR "Configuration Saved" ::PROMPT ">"
```

- Notice there is a space after the close parenthese ") " in each file. This is done because there is a space in the manual example before the "y" is entered.
- NOTE: Regular Expression metacharacters will not have any special meaning if included in ***PROMPT*** or ***CHECKFOR*** strings.

5.7.5 Test Mode

New with version 5.1 is the ability to run the scripts below in "***testmode***". In testmode, the script just displays the commands that would have been sent to the device(s) but does NOT actually login and connect to any of the devices. The commands that would have been sent are also saved to a file ($SCRIPT_HOME/<script_name>_sample_cmds.txt). This is extremely useful (and recommended) when the command file contains Variables and Looping. Testmode is activated by just selecting the testmode option within the GUI or by using -testmode if running the script through the command line.

cisco_send_cmds

cisco_send_cmds_rcf
cisco_config_cmds
cisco_config_cmds_rcf
pix_send_cmds
catos_send_cmds
generic_send_cmds
nxos_send_cmds
nxos_config_cmds

5.8 Juniper JUNOS Scripts

The Net-Sense Automater provides support for Juniper devices running JUNOS through the scripts in the table below. Note, these scripts require a Net-Sense AutomaterPro license.

Script Name	Description
jun_send_cmds	Send the same ***configuration*** commands to one or more devices running JUNOS
jun_config_cmds	Send a set of **non-configuration** commands (e.g. show ip route) to a group of routers.
jun_pinger	Originate pings from a Juniper device and confirm pings were successful or not.
jun_tracer	Perform traceroutes from a Juniper device and confirm the traceroute was successful and also verify the expected path for packets was taken.

5.8.1 JUNOS Command Sender (jun_send_cmds)

The JUNOS commander sender is the as the cisco_send_cmds script except it used for Juniper devices running JUNOS.

This program will send any number of commands to a single JUNOS device or a list of JUNOS devices. The commands that are sent are defined in a text file (any name), that you create, or specified directly on the command line using the -cmd option. This script is ***different*** than the ***jun_config_cmds*** script in that this script is designed to send ***non-configuration*** commands. The commands are sent in sequential order from top to bottom of the file or the order in which they are defined if using the -cmd options. This script can also send configuration commands but the actual commands to enter and exit configuration mode, along with the "commit", must also be included in the list of commands to send.

Note, for global JUNOS device configuration changes the jun_*config_cmds* script is the preferred tool as that script has more comprehensive error checking designed for configuration commands only.

Program Name: jun_send_cmds

Script Argument	Description
-rf <filename>	List of devices or IP Address/(DNS hostname) to run script against (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!

-ipaddr <ip_address or routername> or -ip <ip_address or routername>	IP address or router/switch name (if defined in hosts file or DNS) Use ***-ipaddr*** to send commands to a single device or ***-rf*** to send commands to list of devices specfied in file. (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-cf <filename>	File which contains a list of commands to send to devices. One command per line. Lines that begin with a "#" are considered comments and will not be sent to the router. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time!
-cmd <command>	Command to send to device. Useful when only needing to send a few commands to a single device or list of devices. It saves you the time of having to create a text file with only a few commands in it. If the command contains spaces then it must be enclosed in double quotes (only when running script from cli, not GUI). This option can be used up to 7 times when used with the GUI. The command line version of this script can repeat this option an unlimited number of times. However, for more than 5 to 7 commands, it is recommended to put the commands in a text file and use the -cf option. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time! Example: ***-cmd "show interface terse"***
-testmode	Test Mode. Build and display the commands that would have been sent to the devices but DO NOT actually connect to the devices and send the commands. Useful for confirming command looping and variable substitutions are correct or for just double checking the commands that will be sent . List of commands are also written to the file $SCRIPT_HOME/<script_name>_sample_cmds.txt". (**OPTIONAL**)
-ts	Display timestamp after entering each command on device. Immediately after logging into the device, the script will issue the command "set cli timestamp".
-dir <directory>	If specified, tells script to save output for each device into a separate file. Files will be saved to the directory specified. If the directory does not exist, the script will create it. By default output for all devices is only saved to the detailed trace file. The directory entered can be either an absolute directory or a relative directory. If running the script from the GUI, a relative directory is relative to the "SCRIPT_HOME" variable. (From the GUI see Options->Settings). (**OPTIONAL**)
-autodir <date | time>	Automatically create new unique directory to save output for each device into a separate file. The choices are ***date*** or ***time***. The ***date*** option will create a new directory consisting of just the date e.g. 08012010 for Aug 1st 2010. The ***time*** option will append the time to the date: e.g. 08012010_12h36m15s

	If used with the -dir option, the new unique directory will be created under the ***-dir directory***. If the -dir option is not used, then the new unique directory will be created under the ***SCRIPT_HOME*** directory. Note, if the ***date*** option is used and that directory name happens to all ready exist, then files in that directory will be overwritten. There are no safety prompts for the user when using this option.
-urfn	Use Route File Name. When saving the output data to a separate file for each device, the filename used will be the name/IP Address stated in the -rf file. The default filename is the router hostname configured on the router. This option only applies when used with the -***dir*** option. (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The command file (-cf <filename>) should contain a list of commands that will be sent to each router. This is not a TCL file, it is a plain text file that must contain one command per line. **Lines that begin with a "#" are considered comments and will not be sent to the router as a command.**

The following commands can be entered in the command file or on the command line and the script will automatically answer any additional confirmations or prompts.


(***Note, output of the commands listed below will NOT be displayed in the contents of individual output files when using the -dir option***)

Command	Description
SLEEP <seconds>	The SLEEP command is not a command that will be sent to the router. If desired, this is a method to introduce a delay between commands. Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The SLEEP command must be entered in all CAPITAL letters otherwise it will be interpreted as a command to send to the router. It must also start at the beginning of the line (no leading spaces or tabs). Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command. ***CASE SENSITIVE***

RETURN	Using the keyword RETURN is equivalent to sending just a carriage return or Enter key. Not commonly used. ***CASE SENSITIVE***
LOOPSTART <num> or LOOPEND	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will repeated are the commands between the LOOPSTART and LOOPEND commands. See Command Looping for more information. ***CASE SENSITIVE***
COUNTERVAR <variable> <start_num> <increment by>	Define a counter variable. Provides a method to define a variable in the command file; it must be used with the LOOPING feature. The variable must be a single alphabetical character. "start_num" is the number that the counter will start at. It will increment by the <increment by> number each time through the LOOP. See Counter Variables for more information. ***CASE SENSITIVE***

The following is a sample command file which performs a show ip ospf neighbors, show interface, then pauses for 15 seconds because the user wishes to visually examine the output of the show interface command (in real time). Then it clears the counters.

```
show ospf neighbors
show interface
SLEEP 15
clear interface statistics all
```

This program also has the option to be run in *Safe* and *SuperSafe* mode which should be considered when running scripts in production environments.

Sample Command 1: The following command will send the commands listed in file ***show_cmds.cmds*** to the devices listed in the file ***rtrs.rt.*** If there are any errors while issuing any of the commands, the script will immediately terminate because of the -***ssafe*** option (SuperSafe Mode). The script will also save the output to a unique filename becasue of the -ulog option. The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
jun_send_cmds -ulog -pw logins.txt -rf rtrs.rt -cf show_cmds.cmds -ssafe
```

Sample Command 2: The following script will send a "show tech" to the router 192.168.0.10***.*** The output will be saved to the file show_tech_nyrtr1.log The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
jun_send_cmds -log show_tech_nyrtr1.log -pw logins.txt -ipaddr 192.168.0.10 -cmd "request suppo
```

Sample Command 3: The following script will issue the command "show interface" 10 times, sleeping for 5 seconds between each time the command is entered. The commands will be sent to the router 192.168.0.10***.*** If there are any errors while issuing any of the commands, the script will immediately terminate because of the -***safe*** option (Safe Mode). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

Note, the backslash at the end of each line will only work in a Linux/Unix shell.

```
jun_send_cmds -pw logins.txt -ipaddr 192.168.0.10 -safe \
       -cmd "LOOPSTART 10" \
       -cmd "show int | match error" \
```

```
-cmd "SLEEP 5" -cmd LOOPEND
```

5.8.2 JUNOS Configuration Tool (jun_config_cmds)

The JUNOS config commands script is the as the cisco_config_cmds script except it used for Juniper devices running JUNOS.

This program will send any number of **configuration** commands to a single JUNOS device or a list of JUNOS devices. The commands that are sent are defined in a text file (any name), that you create, or specified directly on the command line using the -cmd option. This file MUST NOT include the commands needed to enter and exit configuration mode on the device. The script will automatically enter the "configuration", "commit" and "exit" commands. The commands are sent in sequential order from top to bottom of the file or the order in which they are defined if using the -cmd options.

Program Name: jun_config_cmds

Script Argument	Description
-rf <filename>	List of devices or IP Address/(DNS hostname) to run script against (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-ipaddr <ip_address or routername> or -ip <ip_address or routername>	IP address or router/switch name (if defined in hosts file or DNS) Use ***-ipaddr*** to send commands to a single device or ***-rf*** to send commands to list of devices specfied in file. (**REQUIRED -rf or -ipaddr**) NOTE: Cannot use the ***-ipaddr*** and the ***-rf*** option at the same time!
-cf <filename>	File which contains a list of configuration commands to send to device. These MUST BE configuration commands only!!! (Do not include commands to enter and exit configuration mode or the commit command) One configuration command per line. Lines that begin with a "#" are considered comments and will not be sent to the device. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time!
-cmd <command>	Command to send to device. Useful when only needing to send a few commands to a single device or list of devices. It saves you the time of having to create a text file with only a few commands in it. If the command contains spaces then it must be enclosed in double quotes (only when running script from cli, not GUI). This option can be used up to 7 times when used with the GUI. The command line version of this script can repeat this option an unlimited number of times. However, for more than 5 to 7 commands, it is recommended to put the commands in a text file and use the -cf option. (**REQUIRED -cf or -cmd**) NOTE: Cannot use the ***-cf*** and the ***-cmd*** option at the same time! Example: ***-cmd "set vlans VLAN100 vlan-id 100"***
-pri	Perform Configuration changes in "private" mode.

-x	Perform Configuration changes in exclusive mode.
-testmode	Test Mode. Build and display the commands that would have been sent to the devices but DO NOT actually connect to the devices and send the commands. Useful for confirming your variable substitutions are correct. List of commands are also written to the file $SCRIPT_HOME/jun_config_cmds_sample_cmds.txt". (**OPTIONAL**)
-safe	Safe Mode. If an error occurs while sending a configuration command to a router in the list, all subsequent commands to that router will not be sent. The script will continue on to the next router on the list and continue sending commands. (**OPTIONAL**)
-ssafe	SuperSafe Mode. Abort Script if there is an error while sending any of the configuration commands to any of the routers. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The configuration command file (-cf <filename>) should contain a list of configuration commands that will be sent to each device. This is a plain text file that must contain one command per line. **Lines that begin with a "#" are considered comments and will not be sent to the device as a command.** The **SLEEP** command is a special command that will **NOT** actually be sent to the router. If desired, this is a method to introduce a delay between configuration commands. The syntax is the **SLEEP** <***seconds***> (all capitals). Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The *SLEEP* command **MUST** be entered in all **CAPITAL** letters otherwise it will be interpreted as a command to send to the router. Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command.

This script also has the option to run in ***Test Mode*** using the -testmode option defined above. With this option, the exact commands that would have been sent to the routers are written to a file (stored in the SCRIPT_HOME directory). The script does not actually login to any devices or send any commands. It is recommended this feature be used when using LOOPING and COUNTER variables (see below)

This program also has the option to be run in ***Safe*** and ***SuperSafe*** mode which should be considered when running scripts in production environments. If entering a configuration command results in an error on the device and the script is running in ***Safe*** or ***SuperSafe*** mode, the script will exit configuration mode and a commit will not be peformed. If the script is NOT running in ***Safe*** or ***SuperSafe*** mode, the commit will be issued even if there are one or more commands that caused configuration errors.

The following special commands are supported by this script.

 (***Note, output of the commands listed below will NOT be displayed in the contents of individual output files when using the -dir option***)

Command	Description
SLEEP <seconds>	The SLEEP command is not a command that will be sent to the router. If desired, this is a method to introduce a delay between commands. Seconds can be a whole integer (e.g. 2) or a real number (e.g. 1.5). The SLEEP command must be entered in all CAPITAL letters otherwise it will be interpreted as a command to send to the router. It must also start at the beginning of the line (no leading spaces or tabs). Note, this is not typically needed but may be useful if you would like to watch the script as it is running to visually inspect the output of a particular command. ***CASE SENSITIVE***
LOOPSTART <num> or LOOPEND	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will repeated are the commands between the LOOPSTART and LOOPEND commands. See Command Looping for more information. ***CASE SENSITIVE***
COUNTERVAR <variable> <start_num> <increment by>	Define a counter variable. Provides a method to define a variable in the command file; it must be used with the LOOPING feature. The variable must be a single alphabetical character. "start_num" is the number that the counter will start at. It will increment by the <increment by> number each time through the LOOP. See Counter Variables for more information. ***CASE SENSITIVE***

Sample Command 1: The following command will send the configuration commands listed in file ***snmp_cmds.cmds*** to the devices listed in the file ***rtrs.rt.*** After the configuration commands are entered. If there are any errors while sending a particular configuration command, the script will abort all remaining commands to that device and continue on to the next device because of the ***-safe*** option (SAFE Mode). The detailed trace-log will be saved to the file ***snmp_log*** instead of the default file name of ***jun_config_cmds.log*** (notice this argument was listed first on the command line). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
jun_config_cmds -log snmp_log -pw logins.txt -rf rtrs.rt -cf snmp_cmds.cmds -safe
```

Sample Command 2: The following script will shutdown interface ge-0/0/8 on switch 192.168.0.10**.** The configuration will be done in exclusive mode because the ***-x*** argument is used. The output log will be saved to an automatically generated unique filename because of the -ulog option. The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option). At the first occurrence of an error, the script will abort because of the -safe option. If there are no errors, the config will be saved to NVRAM because of the -wm option.

```
jun_config_cmds -ulog -pw logins.txt -ipaddr 192.168.0.10 -cmd "set interface ge-0/0/8 disable"
```

5.8.3 JUNOS Pinger

The JUNOS pinger script is the same as the pinger script except it used to source pings from Juniper devices running JUNOS.

The jun_pinger program is a utility that performs pings from a JUNOS device to any IP Address. The script will cycle through a list of routers and perform pings from each router in the list. In addition, the pings can be sent with different source IP addresses. Note, the source IP address must be a valid interface IP Address on the router that is in an UP/UP condition. If the interface is not in an UP/UP condition, the script will not perform the pings using that source IP Address. The special keyword “default” can also be specified as a source IP address. In this case the router software will use the

interface IP address, where the ping packet exits, for the source IP address. This script supports both IPv4 and IPv6 addressing.

Program Name: jun_pinger
Template File: *pinger_template_rev4.txt*

Script Argument	Description
-sf <filename>	Input variable file which tells the program which routers the pings will be performed from, the source IP addresses to use for the pings and the addresses to ping. ***The sample template filename is pinger_template.txt*** (**REQUIRED**)
-pinfo	Print Info. This option tells the script to print out (and log) informational messages as to source/destination pairs for pings that were successful. Be default, only messages for pings that have failed are written to the displays and log file.
-sv	Skip Validation. Before an address is pinged, the script verifies the address is a valid IPv4 or IPv6 address. Using the -sv skips this validation check. This is useful if the IP address is a string that gets resolved to an IP address via DNS lookup.
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The ping source/destination pairs that will be sent are controlled by the variables in the input file (-sf <filename>). This file is in TCL format and contains two TCL list variables; TOPO_LIST and PING_LIST. The TOPO_LIST variable lists the routers along with the source IP Addresses that will be used for the pings. The PING_LIST variable contains the list of IP Addresses that will be pinged.

Below shows a sample entry for the TOPO_LIST variable (for Revision 4 input file):

```
lappend TOPO_LIST [list nj1 "no" "global" "default" "64.145.25.3"]
```

The first item in the list "nj1" is the router that the pings will be performed from. The script will actually telnet/ssh into this router. This must be an IP Address or a name that can be resolved through DNS. (If the -pw <*password_file*> option is used, then this IP Address/Name must also be defined in the *password_file*. Note, they must match exactly [case sensitive]). The second item specifies whether or not the ping is being done in a VRF. The third item is the vrf name. If vrfs are not being used, then the third item must be the string "global". The forth and fifth arguments are the source IP addresses that will be used for the pings.

Below shows a sample entry for the PING_LIST variable:

```
lappend PING_LIST [list "200.32.127.138" "ny2 l0"]
```

```
lappend PING_LIST [list "10.10.1.1" "global" "nj1 Lan"]
```

The first item in the list “10.10.1.1” is the IP Address that will be pinged. The second item in the list is the vrf name or "global" if address is not in vrf. If this string does not match the vrf name from TOPO_LIST, then the ping to this address will be skipped. The third item in the list, “nj1 Lan” is an informational comment about that IP Address. This comment is also printed out in the error log if a ping fails to this IP Address.

Below is a sample file (Rev 4) that will be used to describe the flow of the program in relation to the input file. Note, the line that starts with a “#” will not be used.

```
lappend TOPO_LIST [list ny2 "no" "global" "10.134.132.4" "200.32.127.138"
"default"]
lappend TOPO_LIST [list cny3 "no" "global" "172.33.1.1" "10.134.132.1"]
lappend TOPO_LIST [list ca1 "no" global" "10.31.240.253" "200.42.59.16"
"10.31.29.1"

lappend PING_LIST [list "10.145.25.1" "global" "????"]
#lappend PING_LIST [list "10.145.25.3" "global" "nj1 eth"]
lappend PING_LIST [list "10.126.4.102" "global" "nj1 eth"]
lappend PING_LIST [list "10.126.5.103" "global" "nj2 eth"]
lappend PING_LIST [list "10.126.6.252" "global" "cnj5 eth"]
```

1. Telnet to router “ny2”
2. Ping each of the IP addresses in the PING_LIST using 10.134.132.4 as the source IP address.
3. Ping each of the IP addresses in the PING_LIST using 200.32.127.138 as the source IP address.
4. Ping each of the IP addresses in the PING_LIST using the source IP Address that the router chooses to insert. More specifically, this is the IP Address of the interface that the packet leaves from.
5. Telnet to router “cny3”
6. Ping each of the IP addresses in the PING_LIST using 172.33.1.1 as the source IP address.
7. Ping each of the IP addresses in the PING_LIST using 10.134.132.1 as the source IP address.
8. Telnet to router “ca1”
9. Ping each of the IP addresses in the PING_LIST using 10.31.240.253 as the source IP address.
10. Ping each of the IP addresses in the PING_LIST using 200.42.59.16 as the source IP address.
11. Ping each of the IP addresses in the PING_LIST using 10.31.29.1 as the source IP address.

There is one additional variable in the input file to control the speed at which successive pings are generated from any one router. The variable is PING_DELAY_INTERVAL and is defined in milliseconds. If a script input file is setup to have a router generate pings to 100 different destinations, a delay (equal to the PING_DELAY_INTERVAL) will be inserted before each new ping destination. This variable was added because results have shown that low-end routers may experience a CPU spike because the script is cycling through the pings extremely fast resulting in a lot of telnet/ssh based traffic. If each router is sending to a low number of ping destinations (10 or under), this should be non-issue.

Several sample input files are provide here.

Sample Command: The following command will run the pinger utility using the information contained in the file ***ping_southeast.txt***. The script will not prompt the user for passwords because the

passwords are being read in from the logins.txt file (***-pw*** option).

```
jun_pinger -pw logins.txt –sf ping_southeast.txt
```

5.8.4 JUNOS Tracer

The JUNOS tracer script is the same as the tracer script except it used to source pings from Juniper devices running JUNOS.

The tracer program is a utility that performs traceroutes from a JUNOS device and compares the recorded path with the expected path. The expected path is predefined through an input file. The script will cycle through a list of routers and perform traceroutes to predefined destinations. If the recorded path does not match the expected path an error message is written to the summary log file. An error message is also logged if the traceroute fails. Due to new newly added features in for version 4.3.2, the new tracer input file is slightly different although the tracer script is backward compatible with the previous tracer input file.

New Featutes:

- Support for multiple paths to the same destination
- Ability to specify the source IP address used for the trace route
- Ability to issue user specified commands before traceroute is performed

Program Name: `jun_tracer`
Template Files: `tracer_template.txt, tracer_vrf_template.txt`

Script Argument	Description
-nopath	Perform the traceroutes but do not compare the actual traceroute path against the expected traceroute path. This may be useful if you would just like to perform the traceroutes and record the output. If the actual path is not equal to the expected path, an error message will NOT be logged.
-sf <filename>	Input variable file which tells the program which routers the pings will be performed from, the source IP addresses to use for the pings and the addresses to ping. ***The sample template filename is pinger_template.txt*** (**REQUIRED**)
-nokey	Don't prompt user for encryption key when using encrypted password file. (**OPTIONAL**)
-ssh	Use Secure Shell when accessing routers. Do NOT use with -pw option. (**OPTIONAL**)
-pw <filename>	Login/Password File. (**OPTIONAL**)
-log <filename>	Save detailed trace file to a name other than the default file name. (**OPTIONAL**)
-ulog	Unique Log file. Save detailed trace log file to a Unique filename automatically created by script. Filename will be in format of scriptname_timestamp.log. (**OPTIONAL**)

The traceroutes that will be performed are controlled by the variables in the input file (-sf <filename>). This file is in TCL format and contains TCL list variables. The list variable ***RTR_LIST*** defines the list of routers that the script will telnet/ssh to, as well as the source IP address that should be used for the traceroutes. It also includes another variable, ***DEST_LIST_x***, that defines the traceroutes that will be

performed while telneted into each router. The value of ***x***, in ***DEST_LIST_x***, is a numerical value that must be different for each router.

Below shows a sample entry for the ***RTR_LIST*** variable (non VRF scenario). The first item in the list "ny1" is the router that the traceroutes will be performed from. The script will actually telnet/ssh into this router. This must be an IP Address or a name that can be resolved through DNS. If the -pw < *password_file*> option is used, then this IP Address/Name must also be defined in the *password_file*. (Note, they must match exactly [case sensitive]). The second argument is another variable (***DEST_LIST_x***) that contains the associated list of traceroutes to check while in that device. The last item in the list is the source IP address to use when performing trace routes from this router. If you would like to use the default source IP address that the router would use, then just specify two double quotes with nothing between them (the sample line below shows this) or you could specify the key word ***default*** for the source IP.

```
lappend RTR_LIST [list "ny1" "DEST_LIST_1" ""]
```

Below shows a sample of ***DEST_LIST_x***. The first item in the list, "10.30.30.1" is the destination of the traceroute. The second item can have a value of ***yes*** or ***no*** and tells the script whether to compare the expected route to the actual route or to just check whether the trace route is successful. The third item, "1.1.1.1 1.1.2.1 1.1.3.1 1.1.4.1", is the actual expected trace route path. If a value of no is entered for the second item, the third item must still have a "dummy" expected trace route path or just two double quotes with nothing in between them (e.g. "")

```
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.2.1 1.1.3.1 1.1.4.1"]
```

New in version 4.3.2, the tracer script now supports traceroute paths with multiple "primary" paths due to "load balancing"/redundancy. For this scenario, all you do is add a second (or third, forth, or more!) list entry with the same exact destination but a different path. The script will then check both of these entries. Here is an example of traceroute paths to 10.30.30.1 with 3 possible different trace route paths:

```
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.2.1 1.1.3.1 1.1.4.1"]
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.100.1 1.1.150.1 1.1.4.1"]
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.200.1 1.1.250.1 1.1.4.1"]
```

Below shows the sample input file (***tracer_template.txt***) that is provided with the program and should be used as a template when creating your own input files (for non-vrf scenarios).

The template input file below instructs the script to do the following:

1. Telnet/ssh into router "br1" and perform traceroutes to the destinations listed in DEST_LIST_1.
2. Telnet/ssh into router "sj2" and perform traceroutes to the destinations listed in DEST_LIST_2
3. Telnet/ssh into router "192.168.1.40" and perform traceroutes to the destinations listed in in DEST_LIST_3

```
#####################
# DO NOT MODIFY THE
# TRACER_REV variable
#####################
set TRACER_REV 2

# Template for trace route script (tracer)

# Scenario: Normal Steady State
```

```
################################################################################
# The list of routers to telnet to and perform a trace route
# Field Definitions:
# 1. rtr: Router to telnet to and perform trace routes
# 2. The associated trace route destinations to check on this router
# 3. Source IP Address used for traceroutes from this router.  To use the default
#    address that the router wants to use, just leave the value, between the two
#    double quotes, to nothing, or put in the key word default (see samples below)
################################################################################
#                     rtr            dest_list        src_IP
################################################################################
lappend RTR_LIST [list "br1"          "DEST_LIST_1"    "1.1.1.1"]
lappend RTR_LIST [list "sj2"          "DEST_LIST_2"    ""]
lappend RTR_LIST [list "192.168.1.40" "DEST_LIST_3"    "default"]

################################################################################################
###
# Note, the yes/no field is whether or not the traceroute should be successfull.
# This is good for failure scenarios where you may want to confirm that a
# traceroute fails after a failure scenario was introduced
# The expected "trace route path list" is compared against the actual
# path when the script runs
################################################################################################
#############
# For router: BR1
# Variables               subnet     yes/no "expected trace route path list"
################################################################################################
#############
lappend DEST_LIST_1 [list "10.30.30.1" "yes" "1.1.1.1 1.1.2.1 1.1.3.1 1.1.4.1"]
lappend DEST_LIST_1 [list "10.4.4.2" "yes" "1.1.1.1 1.1.6.1 1.1.7.1 1.1.8.1"]

################################################################################################
#############
# For router: SJ2
# Variables               subnet     yes/no "trace route path list"
################################################################################################
#############
lappend DEST_LIST_2 [list "10.30.30.1" "yes" "2.1.1.1 2.1.2.1 2.1.3.1 2.1.4.1"]
lappend DEST_LIST_2 [list "10.4.4.2" "yes" "2.1.1.1 2.1.6.1 2.1.7.1 2.1.8.1"]

################################################################################################
#############
# For router: 192.168.1.40 (3745)
# Variables               subnet     yes/no "trace route path list"
################################################################################################
#############
lappend DEST_LIST_3 [list "10.30.30.1" "yes" "3.1.1.1 3.1.2.1 3.1.3.1 3.1.4.1"]
lappend DEST_LIST_3 [list "10.30.30.1" "yes" "3.1.1.1 3.1.6.1 3.1.6.1 3.1.6.1"]
```

Sample Command: The following command will run the tracer utility using the information contained in the file ***tracer_northeast.txt***. The –ual option instructs the script to login into 2nd level access (i.e. privileged mode). The script will not prompt the user for passwords because the passwords are being read in from the logins.txt file (***-pw*** option).

```
tracer -pw logins.txt –sf tracer_northeast.txt –ual 2
```

Caveats

Part VI

6 Caveats

Caveats

- ***service prompt config***: By default Cisco IOS contains a "configuration prompt" when the user is in configuration mode. However, this was not always the case. For old versions of IOS a configuration prompt was not returned after the user entered a configuration command. After the user entered "return", if nothing came back to the screen the command was considered successful. Cisco changed this in later versions of IOS but for backward compatibility they gave the user the option to turn off the configuration prompt. Turning off the configuration prompt (***no service prompt config***) causes a problem for any of the scripts that enter commands in configuration mode. Running any of the scripts below with the configuration prompt off will result in errors. Do Not Run Them until the service prompt config command has been entered on the router.

Current scripts that are affected are:

- ***cisco_config_cmds***
- ***cisco_passwd_change***
- ***cisco_send_cmds*** (Only if sending config commands with this script)

Below shows a sample output in configuration mode when a configuration prompt is present:

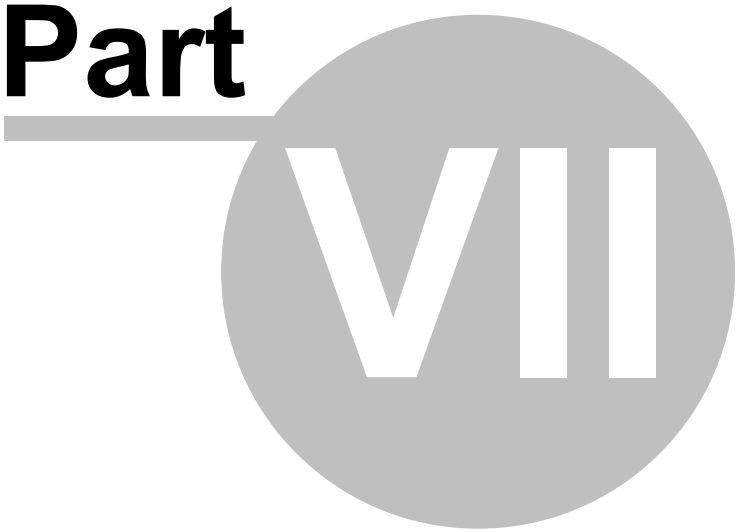
```
rtr-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
rtr-1 (config)#ip host test 1.1.1.1
rtr-1 (config)#ip host test2 2.2.2.2
rtr-1 (config)#end
rtr-1#conf t
```

Below shows a sample output in configuration mode when there is not a prompt in config mode:

```
rtr-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
ip host test 1.1.1.1
ip host test2 2.2.2.2
end
rtr-1#conf t
```

Workaround: Enter the configuration command "***service prompt config***" manually on all routers or use the ***cisco_config_cmds*** script to do this. With the ***cisco_config_cmds*** script, enter the command "***service prompt config***" in the command file. Each router the script is run against will report an error after entering the "***config terminal***" command. This error can be ignored when reviewing the "***summary.log***" file.

Technical Support



7 Technical Support

For any technical issues, questions, or problems please visit the web site at www.net-sense.com or contact Net-Sense Technical Support at support@net-sense.com.

Index

- A -

- B -

- C -

- I -

- L -

- N -

- O -

- P -

- S -

- T -

- U -

- V -

Endnotes 2... (after index)

Back Cover