

Net-Sense Automater Users Manual

Version 4.4

Net-Sense Automater

© 2008 Net-Sense Inc

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Cisco, Cisco IOS, and IOS are registered trademarks of Cisco Systems, Inc. Sun and Solaris are registered trademarks of Sun Microsystems, Inc. "Red Hat" is a registered trademark of Red Hat Software, Inc. Microsoft is a registered trademark of Microsoft Corporation. All other trademarks, service marks, register trademarks, or registered service marks mentioned in this document are the property of their respective owners.

THE INFORMATION AND MATERIAL CONTAINED IN THIS DOCUMENT ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY WARRANTY CONCERNING THE ACCURACY, ADEQUACY, OR COMPLETENESS OF SUCH INFORMATION OR MATERIAL OR THE RESULTS TO BE OBTAINED FROM USING SUCH INFORMATION OR MATERIAL. NEITHER NETSENSE INC, NOR THE AUTHOR SHALL BE RESPONSIBLE FOR ANY CLAIMS ATTRIBUTABLE TO ERRORS, OMISSIONS, OR OTHER INACCURACIES IN THE INFORMATION OR MATERIAL CONTAINED IN THIS DOCUMENT, AND IN NO EVENT SHALL NETSENSE, INC OR THE AUTHOR BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF SUCH INFORMATION OR MATERIAL.

Printed: March 2008 in (New Jersey, US)

Table of Contents

Foreword	0
Part I Introduction	2
Part II Installation - System Requirements	4
1 Windows Installation.....	4
Windows License Key	4
2 Unix/Linux Installation.....	4
Program Installation for System Administrator	4
End User Installation/Setup Procedures	6
Part III Running the Scripts	9
1 From the GUI.....	9
2 From the Command Line.....	10
3 Device Passwords.....	12
4 Post Script Run Checks.....	14
5 Generic Parameters.....	15
6 Generic Command Line Options.....	17
7 Script Errors and Aborts.....	18
Script Initiated Aborts	18
User Initiated Aborts	18
Part IV Scripts	21
1 Script Selector.....	21
2 Cisco Scripts.....	22
IOS Report (ios_report)	22
Flash Report (flash_report)	23
Inventory Report (cisco_inventory_rpt)	24
Router Configuration Backup Utility (copy_to_ftp)	26
Router Configuration Backup Utility 2 (conf_bkup_show_run)	27
Cisco Password Changer (cisco_passwd_change)	28
Global Router Configuration Tool (config_devices)	31
Global Router Configuration Tool (config_devices_rcf)	32
Cisco Command Sender (cisco_send_cmds)	34
Cisco Command Sender (cisco_send_cmds_rcf)	37
IP Accounting (ip_account)	40
BGP Attribute Checker (check_bgp_routes)	41
CatOS Command Sender (catos_send_cmds)	44
PIX Firewall Command Sender (pix_send_cmds)	46
PINGER: Verify Any-to-Any IP Connectivity (pinger)	48
VRF Support for Pinger Script.....	50
Pinger Skip List and Diagnostic Features.....	51
Tracer: Verify Packet Paths Through Network (tracer)	51
VRF Support for Tracer Script.....	54

Additional Tracer Features and Diagnostics.....	54
3 Generic Command Sender	55
4 Login/Password Encryption Utility.....	58
5 Command Looping.....	59
6 Sample Template Files.....	60
Login Password Template File	60
Pinger Template File	61
Pinger VRF Template File	64
Tracer Template File	67
Tracer VRF Template File	68
Cisco Password Change Template File	70
BGP Attribute Checker Template File	71
Generic Send Commands Template File	73
Cisco Send Commands RCF	74
Cisco Config Command Sender RCF	74
Part V Caveats	76
Part VI Technical Support	78
Index	0

Introduction

Part



1 Introduction

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 Unix 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:

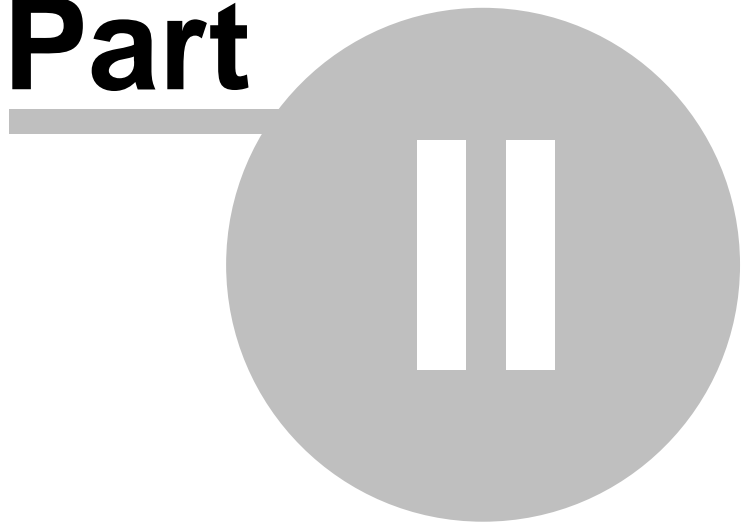
- Cisco Routers (IOS)
- Cisco Switches (IOS)
- Cisco Switches (CatOS)
- PIX Firewalls

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

Part



2 Installation - System Requirements

The Automater is supported on the following operating systems:

- Solaris (2.6 and above)
- Red Hat Linux 7.2 and above.
- Fedora Core Linux
- Suse Linx 9.2 and above
- Windows NT, 2000, XP, 2003

2.1 Windows Installation

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

- Windows NT
- Windows 2000
- Windows XP
- Windows 2003

To install the program, just double click on the .exe file and follow the on screen instructions. The default installation directory is **C:\Program FilesWet-Sense**. If the program is installed in a different directory, the file *[install-dir]/bin/setup.var* must be edited. Specifically, there is a line in the beginning of the file as follows:

```
set SCRIPT_HOME [file nativename "c:/Program Files/Net-Sense/userdata"]
```

The directory specified in that line must included the actual installation directory. Belows shows an example where the installation directory was changed from the default to *C:\My ProgramsWet-Sense*. (Note, the forward slashes will automatically be changed to backslashes for the MS Windows environment.)

```
set SCRIPT_HOME [file nativename "c:/My Programs/Net-Sense/userdata"]
```

2.1.1 Windows License Key

By default, the program 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. Assuming the program has been installed in the default installation directory, edit the file *C:\Program FilesWet-Sense\bin\license_key*. Remove the word TRIAL and put in your license key. Save and close the file.

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.tar.)
`cp automater.tar /tmp`
3. Change directory to the temporary directory
`cd /tmp`
4. Extract the tar file:
`tar xvf automater.tar`
5. Change directory into the sub-directory created from extraction of the tar file:
`cd /tmp/build_020503`
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_020503]# ./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_020503]#
```

7. Update the License key file. After obtaining the license key from technical support, 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. Under your HOME directory, create a sub-directory called **net-scripts** (or a different name if you'd like. Note, the documentation below assumes the directory name created under your HOME directory is **net-scripts**.)

```
linux-1:/home/jsmith> cd $HOME
linux-1:/home/jsmith> mkdir net-scripts
```

2. Copy the script template files into this newly created directory:

```
linux-1:/home/jsmith> cd net-scripts
linux-1:/home/jsmith/net-scripts> cp /usr/local/net-sense/templates/* .
```

3. Next, copy the file **setup_template.var** to **setup.var**.

```
linux-1:/home/jsmith/net-scripts> cp setup_template.var setup.var
```

4. Edit the variable **SCRIPT_HOME** in the file **setup.var**. This variable must be changed to the new directory created in Step 1. Change only the portion highlighted below.

```
#####
# Define the directory where the scripts will be run from.
#####
set SCRIPT_HOME [file nativename "/home/jsmith/net-scripts"]
```

-
5. Confirm that the bin sub-directory, where the system administrator installed the scripts (e.g. /usr/local/net-sense/bin), is in your path. Issue the command **env | grep ^PATH** to confirm. If this directory is not in your PATH variable, add this path to your **.profile** file. If the PATH variable already exists in your .profile, just append the path "/usr/local/net-sense/bin". If it does not exist, add a PATH environment variable and "export"

Example (Existing entry in .profile)

```
PATH=/usr/sbin
export PATH
```

Change to:

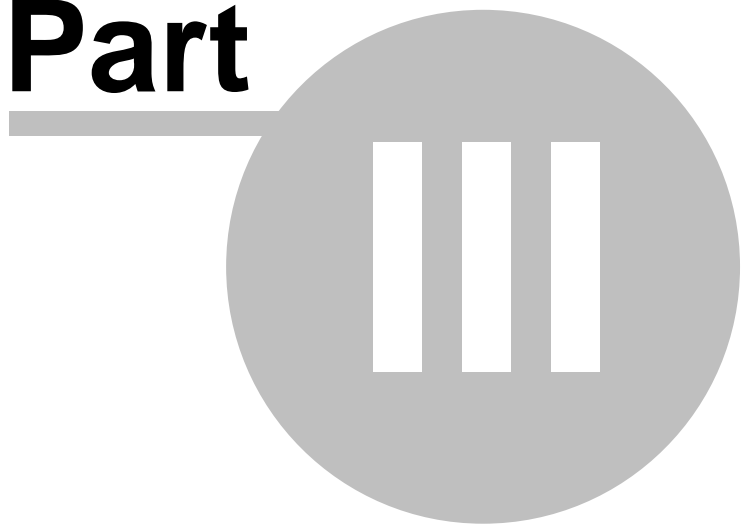
```
PATH=/usr/sbin:/usr/local/net-sense/bin
export PATH
```

Afterwards, confirm the new directory has been added to your path.

```
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:.
```

Running the Scripts

Part



3 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.

3.1 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. Change Directory to the net-scripts directory: **cd \$HOME/net-scripts**
2. Type in **automater** at the command prompt.

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



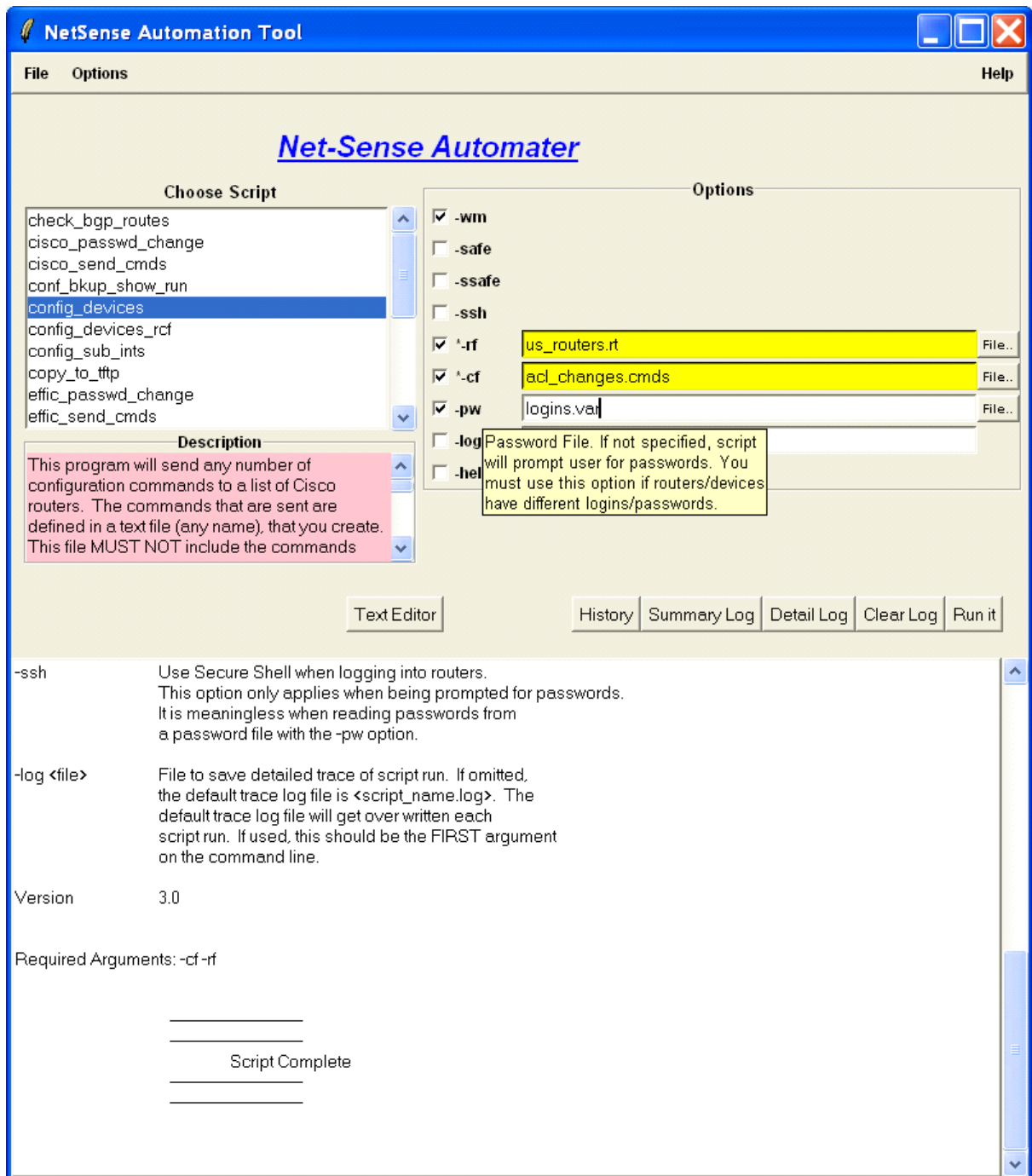
Note, for Unix/Linux you must be in the “net-scripts” subdirectory created during the End User Setup when starting the GUI!

The steps to running any of the scripts are listed below. Details for each of the scripts along with their options are described in Sections 4 & 6.

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** button. The output of the script is written to the Log Window.



3.2 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.

For Unix/Linux the scripts are run from the shell prompt and **MUST** be run from the directory created in **Section 3.2.2 Step 1** (End User Installation/Setup Procedures). They must be run from this directory because the program is looking for the file **setup.var**, created in during the installation and end-user setup.

For MS Windows, the scripts are run from the **cmd tool** window (Start->All Programs->Accessories->command prompt). For Windows the scripts can actually be run from any directory but it makes sense to run the scripts from the directory specified by the SCRIPT_HOME value (from the GUI see Options->Settings). This is the directory where most of your files will be kept when running the scripts from the GUI. Thus, you can recall those same files without having to reference the absolute path name of the file.

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

Script Name
 encrypt_logins
 ios_report
 flash_report
 copy_to_tftp
 conf_bkup_show_run
 cisco_passwd_change
 config_devices
 config_devices_rcf
 cisco_send_cmds
 cisco_send_cmds_rcf
 cisco_inventory_rpt
 pinger
 tracer
 ip_account
 check_bgp_routes
 catos_send_cmds
 pix_send_cmds
 generic_send_cmds

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”

3.3 Device Passwords

There are two ways to provide router (or other device types) usernames/passwords for the scripts. One, username/passwords can be stored in a file and that file is referenced as a script option (with the **-pw** option). Or secondly, when a running a script the script will interactively prompt the user to enter in the username/password information. This second method will automatically happen if the **-pw** option is not used for a script run.

The login/password file is used to store Usernames and Passwords for the routers. This feature is beneficial when routers in your network have many different passwords. This is an OPTIONAL but useful feature. If all of the routers in your network have the same passwords, then a password file is not needed. If a password file is needed, it should be encrypted using the encrypt_logins program.

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 this file for the IP Address/Name of the

router the script is telneting into. The exact "IP Address/Name" it searches for is the "IP Address/Name" that was defined in the file with the list of routers (-rf <filename>). Thus, it is important that these two values match or the login information will not be found for a particular router (case sensitive).

The program comes with a sample password file (logins_template.var) that should be used as a template to create your own router password file. 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	1 st 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	1 st level password. If TACACS is enabled, this is the Username password. If TACACS is not enabled, this is the "typical" 1 st level password for the router. This is the password used for Efficient Routers.
F5	2 nd 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	2 nd 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 2 nd 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 “ <i>ssh</i> ”. (Note, does not apply for Efficient Routers or Nortel Devices)
F8	This field MUST always be left blank (i.e. “”). It is a place holder used internal to the program.

Another useful feature is the default password option. This is the login/password information used if the script cannot find the router name in F2 of any of the entries. 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.var*) 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
```

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

##### 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

```

3.5 Generic Parameters

There are several generic/global parameters that affect all of the scripts. These generic parameters are kept in the *setup.var* file. For Unix/Linux installations, the 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 is located in *[install-dir]/bin* (if the default installation directory was used, the file is c:\Program Files\Net-Sense\bin\setup.var). 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

	<p>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</p>
timeout	<p>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</p>
DELAY_INTERVAL	<p>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</p>
CUSTOM_LOGIN_PROMPT	<p>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</p>
CUSTOM_PASSWORD_PROMPT	<p>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</p>
Match_max -d	<p>Shouldn’t need to change this Default: 500000</p>
EDITOR	<p>This is the program used when the “Text Editor” button is pressed on the GUI. The default program used is “gedit” which is available with Redhat Linux. If you are using the Solaris version, this variable must be changed or the “Text Editor” button will not work. The setup_template.var file lists more choices to use. Default: gedit</p>
PDF_VIEWER	<p>This is the program used when the “Help” button is depressed on the GUI. If Adobe Acrobat is installed on the system, then change this variable to acroread (note, make sure the program acroread is in your PATH environment variable (e.g .profile, .bash_profile, or .cshrc). The default value is xpdf which is available on Redhat Linux. If you are using Solaris, you will need to Install Adobe Acrobat and set this variable to acroread or use another PDF viewer program and set this variable to the program name. For MS Windows Installations, the OS will determine which program to view the help file based on the filename extension. Note, for MS Windows the variable still needs to exist (just leave the default value).</p>

	Default: xpdf OBSOLETE: For Version 4.3 and above.
USERS_MANUAL	Users Manual file name. Shouldn't need to change this Default: users_manual.pdf OBSOLETE: For Version 4.3 and above.

3.6 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.var*) 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.var*". This file is in TCL format. The template file should be edited and saved to a new filename (preferably with a .var 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 can be specified as one per line or you can have multiple routers specified per line, separated by space or tab. 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 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
#####
# Routers on the East Coast
#####
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.

3.7 Script Errors and Aborts

There are several reasons why the script could exit before completion. One, is prerequisite checks that are built into the script have failed. Another is the user aborts the script by selecting the Abort button from the GUI or entering "Control C if run from the command line.

3.7.1 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]$ config_devices -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"]
```

3.7.2 User Initiated Aborts

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

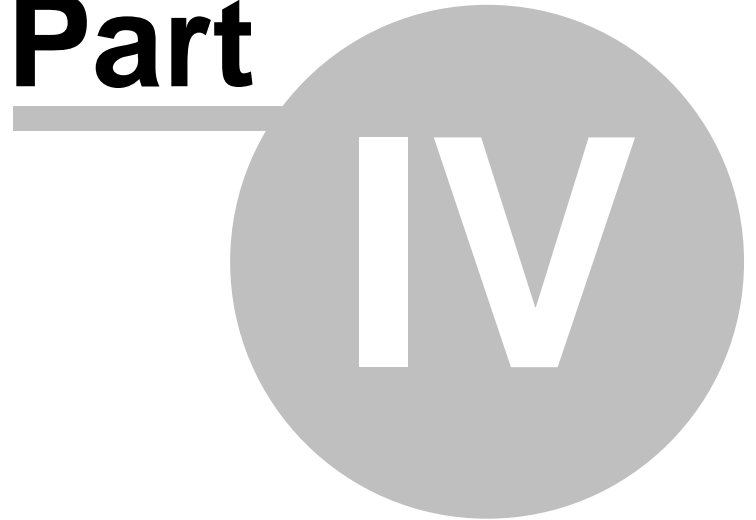
- Type **Control-C** when script is run from the Command Line
- Click the "**Abort**" button on the GUI.

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



4 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. Here, you can introduce a delay after the script finishes with one router and before it telnets into the next router.

4.1 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	config_devices
Send configuration commands to a group of routers but the actual configurations commands are different for each router.	config_devices_rcf
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
Change any of the following passwords on a set of routers:	cisco_passwd_change

<ul style="list-style-type: none"> • secret • enable • vty • console • auxiliary 	
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 <i>-dir</i> 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
Collect IP Accounting information from a set of routers	ip_account
Encrypt the password/login file	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

4.2 Cisco Scripts

4.2.1 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
- Router Model
- IOS Version
- IOS Feature Set
- Memory
- Boot Flash Version
- Boot Strap Version

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. (REQUIRED)
-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)

Sample Command: The following command runs the `ios_report` program against the routers listed in the file `rtrs.rt`. 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 `report`.

```
ios_report -pw logins.var -rf rtrs.rt -of report
```

Sample Output Report:

Router	Model	Version	Feature Set	Memory (KB)	BootFlash	BootStrap
10.1.1.1	1720	12.1(5)T8b,	(C1700-SY-M),	32	----N/A----	12.0(1)T,
10.1.2.1	3640	11.3(11)C,	(C3640-IS-M),	64	----N/A----	11.1(20)AA2,
10.1.3.1	3620	11.3(11)C,	(C3620-JS-M),	64	----N/A----	11.1(4)AX
10.1.4.1	3620	12.1(18)T2,	(C3620-JS-M),	64	----N/A----	11.1(4)AX
10.1.5.1	3640	12.2(13)T,	(C3640-JS-M),	128	----N/A----	11.1(12)AA,

4.2.2 Flash Report (flash_report)

The Flash Report Program provides a Flash report for Cisco Routers. The script cycles through a list of routers and performs a 'show version" and then a second command which is dependent on the extracted from the show version (e.g. show flash, show slot0, show bootflash, etc.). The relevant data is extracted from the commands and saved to a report file. The data included in this report is:

- Router Name/ IP Address
- Router Model
- IOS Version
- Total Flash
- Free Flash
- Flash Slot Number
- Flash Type/Description

Program Name: `flash_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. (REQUIRED)
-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)
-bootflash	Include bootflash information in the report
-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)

Sample Command: The following command runs the flash_report program against the routers listed in the file *rtrs.rt*. The script will prompt the user for passwords the **-pw** option is **NOT** being used. The report will also include bootflash information because the **-bootflash** option is defined. The resulting report will be written to the file *report*.

```
flash_report -rf rtrs.rt -of report -bootflash
```

Sample Output Report: (after importing data to MS Excel)

Router	Model	Version	Total Flash	Free Flash	Slot	Flash Type
rtr-ny1	7206VXR	12.2(2)T2	46976K	16804K	0	ATA PCMCIA card at slot 0 (Sector size 512 bytes).
			8192K	677K	-N/A-	BOOTFLASH: Flash internal SIMM (Sector size 256K).
rtr-ny2	3640	12.2(2)T	16384K	15865K	-N/A-	processor board System flash (Read/Write)
			20480K	3885K	1	processor board PCMCIA Slot1 flash (Read/Write)
rtr-ca1	7204VXR	12.1(13)	46976K	4252K	0	ATA PCMCIA card at slot 0 (Sector size 512 bytes).
			8192K	3057K	-N/A-	BOOTFLASH: Flash internal SIMM (Sector size 256K).
rtr-ca2	7206VXR	12.2(2)T2	46976K	12352K	0	ATA PCMCIA card at slot 0 (Sector size 512 bytes).
			4096K	0K	-N/A-	BOOTFLASH: Flash internal SIMM (Sector size 256K).
rtr-ma1	2621XM	12.3(2)	49152K	33728K	-N/A-	processor board System flash (Read/Write)
rtr-ma2	3745	12.3(2)T	125184K	93196K	-N/A-	ATA System CompactFlash (Read/Write)
			125184K	124828K	0	ATA Slot0 CompactFlash (Read/Write)
rtr-ma3	3725	12.2(6)T	125184K	93312K	-N/A-	ATA System CompactFlash (Read/Write)
			125184K	125184K	0	ATA Slot0 CompactFlash (Read/Write)
rtr-nj1	7206VXR	12.2(2)T2	47040K	13312K	0	ATA PCMCIA card at slot 0 (Sector size 512 bytes).
			251904K	251904K	2	ATA PCMCIA card at slot 2 (Sector size 512 bytes).
			16384K	9465K	-N/A-	BOOTFLASH: Flash internal SIMM (Sector size 256K).
rtr-nj2	3660	12.3(2)T	16384K	1221K	-N/A-	processor board System flash (Read/Write)

4.2.3 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 (1 st or 2 nd) to log into the router before performing the show inventory command. By default the script will only go into 1 st level access. (OPTIONAL)
-del <delimiter>	String used as delimiter between columns in the output file. The default delimiter is 3 consecutive colons(::). This may be useful to change when importing the results of this script into a program such as Microsoft Excel.
-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)

Sample Command: The following command runs the cisco_inventory_rpt program against the routers listed in the file *rtrs.rt*. 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.var -rf rtrs.rt -of inventory_report.txt
```

Sample Output Report (after importing data to MS Excel)

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

4.2.4 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:".

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)
-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)

The configuration filename that is saved to the TFTP server is <routername-config>. 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-config*. 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 **-tftpboot** 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 **-tftpboot** 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 -tftpboot option, the TFTP server ip address must be the same as the system that the script is being run from.** (Note, 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 **-tftpboot** 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 -tftpboot /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 **-tftpboot** 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 -tftpboot /tftpboot
```

4.2.5 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)
-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)

The configuration filename that is saved is <router_name--config.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--config.txt*. The directory that the configuration files are saved to can also be changed using the "-dir" option. If the -dir option is omitted; by default the script will save the router configuration files to the following locations:

Script Run From	Location
MS Windows GUI	c:\Program Files\net-sense\userdata
Unix/Linux GUI and command line	\$HOME/net-scripts
MS Windows command line	Current directory script is run from

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 is relative to the directories stated in the above table

Sample Command: The following command will backup the configuration files for the routers listed in the file *rtrs.rt*. The configuration files will be stored in the directory *\$HOME/net-scripts/02-25-03_configs*. The config files will automatically be *touched* on the TFTP server. The **-tftpboot** option also causes the script to automatically create the directory specified in the **-subdir** option. 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
```

4.2.6 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
- Auxiliary

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)

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.var). 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.var -sf cisco_passwd_change_temp1
*****
* 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 Auxiliary 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 Auxiliary 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.var*. 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.var -wm -ssafe
```

4.2.7 Global Router Configuration Tool (config_devices)

This program will send any number of **configuration** commands to a list of Cisco routers. The commands that are sent are defined in a text file (any name), that you create. This file 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: config_devices

Script Argument	Description
-rf <filename>	List of routers or IP Address to run script against (REQUIRED)
-cf <filename>	File which contains a list of configuration commands to send to router. These MUST BE configuration commands only!!! One configuration command per line. Lines that begin with a “#” are considered comments and will not be sent to the router. (REQUIRED)
-wm	Saves configuration file to NVRAM after making the configuration changes. This option performs a “write memory” 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 <code>-pw</code> option. (OPTIONAL)
-pw <filename>	Login/Password File. (OPTIONAL)
-log <filename>	Save detailed trace file to a name other than the default file name. (OPTIONAL)

The configuration command file (`-cf <filename>`) should contain a list of configuration 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.**

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 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 `config_devices.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.var` file (`-pw` option).

```
config_devices -log snmp_log -pw logins.var -rf rtrs.rt -cf snmp_cmds.cmds -wm -safe
```

4.2.8 Global Router Configuration Tool (`config_devices_rcf`)

Global Router Configuration Tool (`config_devices_rcf`)

This program is very similar to the `config_devices` script above except it is designed to send a **different** set of **configuration** commands to each Cisco router. Do not use this script to send exec level commands (e.g., `show` commands). The specific configuration commands, and the routers 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: config_devices_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)
-wm	Saves configuration file to NVRAM after making the configuration changes. This option performs a “write memory” 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)

Below shows an 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 “config_devices” script.

```
# Sample template file for config_devices_rcf script
# Lines that begin 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 **config_devices_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.var file (**-pw** option).

```
config_devices_rcf -pw logins.var -rcf rtr_mod_100104.rcf -wm -ssafe
```

4.2.9 Cisco Command Sender (cisco_send_cmds)

This program will send any number of commands to a list of Cisco routers. The commands that are sent are defined in a text file (any name), that you create. This script is **different** than the **config_devices** 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. 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 **config_devices** 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 to run script against (REQUIRED)
-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)
-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)
-urfn	Use <u>Route File Name</u> . 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)

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
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	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. 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 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 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. 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 be 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 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 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.var` file (`-pw` option).

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

4.2.10 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 router. 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 `config_devices_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
<code>-rcf <filename></code>	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)
<code>-wm</code>	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)
<code>-ual (1 or 2)</code>	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)
<code>-dir <directory></code>	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)

-urfn	Use <u>R</u> oute <u>F</u> ile <u>N</u> ame. When saving the output data to a separate file for each device, the filename used will be the name/IP Address stated in the <code>-rf</code> file. The default filename is the router hostname configured on the router. This option only applies when used with the <code>-dir</code> 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 <code>-pw</code> option. (OPTIONAL)
-pw <filename>	Login/Password File. (OPTIONAL)
-log <filename>	Save detailed trace file to a name other than the default file name. (OPTIONAL)

Below shows an 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 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

nyrtrl:nyrtrl.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	<p>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.</p> <p>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)</p>
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 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 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. 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 be 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 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.var` file (`-pw` option).

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

4.2.11 IP Accounting (`ip_account`)

The IP accounting program will collect IP Accounting data from a list of Cisco routers and write the results to files. The script will cycle through each router issuing the “show ip accounting” command. A separate data file will be created for each router and the data in the file can be sorted by Packet Count, Byte Count, Source IP Address, or Destination IP Address. Note, “ip accounting” must be enabled on a router interface in order for data to be available. Please check the appropriate Cisco documentation for more on Cisco’s IP accounting feature.

Program Name: `ip_account`

Script Argument	Description
<code>-ual (1 or 2)</code>	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)
<code>-sort (s d p b)</code>	Sort output results based on one of the following: s source IP address d destination IP address p packet count b byte count The default is unsorted and will present the data exactly the way it is presented from the “show ip accounting” command. (OPTIONAL)
<code>-urfn</code>	Use <u>R</u> oute <u>F</u> ile <u>N</u> ame. Use ip address/name in route file for data file name instead of router hostname.

-dir <directory name>	A sub-directory under the user's "net-scripts" directory that the data files will be written to. By default, the data files will be written to the user's "net-sense" directory. (OPTIONAL)
-clc	Clear the IP accounting counters after retrieving the data. (OPTIONAL)
-nokey	Don't prompt user for encryption key when using encrypted password file. (OPTIONAL)
-ssafe	SuperSafe Mode. Abort Script if there is an error while issuing the "clear ip accounting" command. (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)

After the script runs, the IP accounting data will be either be in the user's "net-scripts" directory or a sub-directory under the user's "net-scripts" directory if the **-dir** option is specified. It is recommended to use the **-dir** option so your "net-scripts" directory will not be "cluttered" with excessive data files.

If you plan on clearing the counters for the IP accounting data, you will most likely need to set the User Access level to 2 with the **-ual 2** option. Unless, your using Cisco's AAA and the particular username has privilege to issue that command.

Sample Command: The following command will run the ip_account program and save the data files, sorted by byte count (**-sort** option), into the directory my_company_070103 (**-dir** option). The script will also be running in 2nd level privileged mode (**-ual 2**) and will clear the IP accounting counters (**-clc**) after collecting the data. If there are any errors while issuing the "clear ip accounting" command, the script will immediately abort because of the **-ssafe** option. The script will not prompt the user for passwords because the passwords are being read in from the logins.var file (**-pw** option).

```
ip_account -pw logins.var -sort b -dir my_company_070103 -ual 2 -clc -ssafe
```

4.2.12 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 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.

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.

	<i>The sample template filename is check_bgp_routes_template.var (REQUIRED)</i>
-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)

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.

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 to. It also includes another variable, **PREFIX_LIST_x**, that defines the BGP routes that will be checked while telneted into 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 second argument must be either "yes" or "no" (case sensitive). This tells the script whether the routing database is 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 third argument 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!

```
lappend RTR_LIST [list "ny1" "no" "vpn_643" "$PREFIX_LIST_1"]
```

The fourth argument is another variable (*PREFIX_LIST_x*) that contains the associated list of BGP routes and attributes to check while in that router. 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" "255.255.0.0" "yes" "175.16.20.5" "N/A" "100" "IGP" " 225 225 1005"]
```

```
ny1#show ip bgp 150.140.0.0 255.255.0.0
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", is the route the BGP attributes will be checked on.
- The 2nd item, "255.255.0.0", is the subnet mask for the route.
- The 3rd 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 4th item, "175.16.20.5" is the BGP "next-hop" attribute.
- The 5th item, "N/A", is the BGP MED attribute.
- The 6th item, "100", is the BGP local preference attribute.
- The 7th item, "IGP", is the BGP "Origin-type".
- The 8th 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.var*) that is provided with the program and should be used as a template when creating your own input files. 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 template input file below instructs the script to do the following:

1. Telnet into router "ny1" and check the BGP attributes of the routes listed in PREFIX_LIST_1
2. Telnet into router "sf1" and check the BGP attributes of the routes listed in PREFIX_LIST_2
3. Telnet into router "192.168.1.40" and check the BGP attributes of the routes listed in PREFIX_LIST_3
4. Telnet into router "ch-per" and check the BGP attributes of the routes listed in

PREFIX_LIST_4. Note, this router contains BGP/MPLS VPNs.

```
#####
# 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: nyl
# Variables          subnet          mask          yes/no next_hop metric pref origin as_path
#####
lappend PREFIX_LIST_1 [list "50.10.10.0" "255.255.255.0" "yes" "0.0.0.0" "N/A" "100" "IGP" "local"]
lappend PREFIX_LIST_1 [list "60.20.20.0" "255.255.255.0" "yes" "200.1.1.2" "N/A" "100" "IGP" "local"]
lappend PREFIX_LIST_1 [list "150.140.0.0" "255.255.0.0" "yes" "175.16.20.5" "N/A" "100" "IGP" "225 7"]
lappend PREFIX_LIST_1 [list "20.5.5.1" "255.255.255.255" "no" "200.2.2.21" "N/A" "100" "IGP" ""]

#####
# For router: sfl
# Variables          subnet          mask          yes/no next_hop metric pref origin as_path
#####
lappend PREFIX_LIST_2 [list "201.50.45.1" "255.255.255.255" "yes" "200.1.1.1" "N/A" "100" "IGP" "local"]
lappend PREFIX_LIST_2 [list "10.30.30.1" "255.255.255.255" "yes" "200.1.1.1" "N/A" "100" "IGP" "225 402
101 2501"]
lappend PREFIX_LIST_2 [list "100.60.0.0" "255.255.0.0" "yes" "200.1.1.1" "N/A" "100" "IGP" "225 402 101
2501"]

#####
# For router: 192.168.1.40
# Variables          subnet          mask          yes/no next_hop metric pref origin as_path
#####
lappend PREFIX_LIST_3 [list "100.60.0.0" "255.255.0.0" "yes" "200.2.2.26" "N/A" "100" "IGP" "402 402"]
lappend PREFIX_LIST_3 [list "20.5.5.1" "255.255.255.255" "yes" "200.2.2.26" "N/A" "100" "IGP" "402 402"]
lappend PREFIX_LIST_3 [list "200.70.70.1" "255.255.255.255" "yes" "200.2.2.101" "N/A" "100" "IGP" "402
402"]

#####
# For router: ch-per (PER)
# Variables          subnet          mask          yes/no next_hop metric pref origin as_path
#####
lappend PREFIX_LIST_4 [list "50.10.10.0" "255.255.255.0" "yes" "200.2.2.42" "N/A" "100" "IGP" "402 225
100 5500 510"]
lappend PREFIX_LIST_4 [list "201.50.45.1" "255.255.255.255" "yes" "200.2.2.42" "N/A" "100" "IGP" "402 225
100 5500 510"]
lappend PREFIX_LIST_4 [list "60.20.20.0" "255.255.255.0" "yes" "200.2.2.42" "N/A" "100" "IGP" "402 225
100 5500 510"]

#####
# 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. Does this router have a vrfs? yes for PERs. No for CE routers.
#    This basically states whether the script needs to enter
#    "show ip bgp" or "show ip v v vpn_name bgp"
# 3. VPN name on PER router. This is only relevant if the 2nd field is "yes"
# 4. The associated Prefixes to check on this router
#
#    rtr vrf? vpn_name prefix list
#####
lappend RTR_LIST [list "nyl" "no" "vpn_643" "$PREFIX_LIST_1"]
lappend RTR_LIST [list "sfl" "no" "vpn_643" "$PREFIX_LIST_2"]
lappend RTR_LIST [list "192.168.1.40" "no" "vpn_643" "$PREFIX_LIST_3"]
lappend RTR_LIST [list "ch-per" "yes" "vpn_12" "$PREFIX_LIST_4"]
```

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

```
check_bgp_routes -pw logins.var -sf bgpattr_east.var
```

4.2.13 CatOS Command Sender (catos_send_cmds)

This program will send any number of commands to a list of Cisco switches running CatOS. 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.

Program Name: catos_send_cmds

Script Argument	Description
-rf <filename>	List of switches or IP Address to run script against (REQUIRED)
-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 switch. (REQUIRED)
-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)
-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 switches. (OPTIONAL)
-nokey	Don't prompt user for encryption key when using encrypted password file. (OPTIONAL)
-ssh	Use Secure Shell when accessing switches. 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)

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. 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 be 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: 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.var file (`-pw` option).

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

4.2.14 PIX Firewall Command Sender (pix_send_cmds)

PIX Firewall Command Sender (pix_send_cmds)

This program will send any number of commands to a list of Cisco PIX Firewalls. 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. This script is not designed to send configuration commands, however, configuration commands can be sent by including the necessary commands to enter and exit configuration mode in the input text file.

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

Program Name: `pix_send_cmds`

Script Argument	Description
-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)
-rf <filename>	List of PIXs or IP Address to run script against (REQUIRED)
-cf <filename>	File which contains a list of commands to send to PIXs. One command per line. Lines that begin with a “#” are considered comments and will not be sent to the switch. (REQUIRED)
-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)
-urfn	<u>Use Route File Name</u> . 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)

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. 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 be 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.var file (`-pw` option).

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

4.2.15 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.

Program Name: pinger

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 <i>pinger_template.var</i> (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. By 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)

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:

```
lappend TOPO_LIST [list nj1 "64.145.25.3" "64.32.127.132"]
```

The first item in the list "nj1" is the router that the pings will be performed from. 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 and third 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 10"]
```

The first item in the list "200.32.127.138" is the IP Address that will be pinged. The second item in the list, "ny2 10" 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 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 "10.134.132.4" "200.32.127.138" "default"]
lappend TOPO_LIST [list cny3 "172.33.1.1" "10.134.132.1"]
lappend TOPO_LIST [list cal "10.31.240.253" "200.42.59.16" "10.31.29.1"]
```

<pre>lappend PING_LIST [list "10.145.25.1" "?????"]</pre>
<pre>#lappend PING_LIST [list "10.145.25.3" "nj1 eth"]</pre>
<pre>lappend PING_LIST [list "10.126.4.102" "nj1 eth"]</pre>
<pre>lappend PING_LIST [list "10.126.5.103" "nj2 eth"]</pre>
<pre>lappend PING_LIST [list "10.126.6.252" "cnj5 eth"]</pre>

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.

A sample input file is provided (pinger_template.var). 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.)

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

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

4.2.15.1 VRF Support for Pinger Script

The Net-Sense Pinger script also supports the use of VRFs. This feature is available by using a different input pinger file (-sf filename) that supports the VRF information. The template file for this is

pinger_vrf_template.var. The format of this file is almost identical to the `pinger_template.var` file with some minor changes as defined below.

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

```
lappend TOPO_LIST [list njl acme_vrf "64.145.25.3" "64.32.127.132"]
```

4.2.15.2 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.var*** and ***pinger_vrf_template.var*** 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 diagnostic 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.var*** and ***pinger_vrf_template.var*** 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).

4.2.16 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 Features:

- 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`

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 <i>pinger_template.var</i> (REQUIRED)
-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)

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. The first item in the list "ny1" is the router that the traceroutes will be performed from. 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 traceroutes to check while in that router. 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.var*) that is provided with the program and should be used as a template when creating your own input files. 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 template input file below instructs the script to do the following:

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

```
#####
###
# 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"]

#####
# 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"]
```

Sample Command: The following command will run the tracer utility using the information contained in the file *tracer_northeast.var*. 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.var` file (`-pw` option).

```
tracer -pw logins.var -sf tracer_northeast.var -ual 2
```

4.2.16.1 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 pinger file (`-sf` filename) that supports the VRF information. The template file for this is *tracer_vrf_template.var*. The format of this file is almost identical to the `tracer_template.var` 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"]
```

4.2.16.2 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.var)

4.3 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. The commands are sent in sequential order from top to bottom of the file. The primary feature of this script is the ability to login to many different types of devices. You 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. G2 device types are devices that prompt the user for a login/username and then a password. 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 UNIX 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 devices or IP Address to run script against (REQUIRED)
-cf <filename>	File which contains a list of commands to send to the devices. One command per line. Lines that begin with a "#" are considered comments and will not be sent to the router. (REQUIRED)
-sf	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)
-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)

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	Allows the capability to send the same command(s) over and over, a specified number of times. The command(s) that will

LOOPEND	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.var***). 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. 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 <code>NY-router1></code> So the value of 1ST_LEVEL_PROMPT would be <code>></code> . If there is a "space" after this character, the space must be included! E.g. <code>> </code> .
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 " <i>terminal length 0</i> ". If this does not apply to your devices, then set this value to an empty string. Example: <code>set TERM_LENGTH_ZERO_CMD ""</code> .
2ND_LEVEL_ACCESS_CMD	The command that you would enter to go into a "privileged" 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 <code>su</code>

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.var file (*-pw* option).

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

4.4 Login/Password Encryption Utility

The `encrypt_logins` utility is used to encrypt the logins/password file that is used with "-pw" option for many of the scripts. This mainly applies to 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. This is also needed if the scripts will be launched automatically with cron or through MS Windows' scheduler.

Program Name: **encrypt_logins**

Script Argument	Description
-if <filename>	Input File. Un-encrypted Password File (REQUIRED)
-of <filename>	Output File. New encrypted Password File. (REQUIRED)
-nokey	Using this option creates the encrypted password file in a way that the key does not need to be 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 creates a new file (-of <filename>) with all of the Usernames and Passwords encrypted. When the script is run, the user must enter an encryption key. 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. After the encrypted file is created, the script will prompt you to delete the un-encrypted password file. Leaving a file on the system with un-encrypted usernames/passwords is a HIGH Security Risk!! For production environments, it is recommended the un-encrypted password file be removed from the system. Below shows an example of running the script:

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

The un-encrypted password file is still on the system

This is a security risk!!
Do you wish to remove the un-encrypted password
file now? (yes/no)? yes

[asilver@localhost tcl]$
```

4.5 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
pix_send_cmds
catos_send_cmds
generic_send_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.

4.6 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
```

4.6.1 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
```

4.6.2 Pinger Template File

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

#####
```

```

# Delay before performing next ping.  If there are alot of ping destinations
# it could drive up the CPU on low end routers.  Putting in a delay between
# pings should lower the CPU.  This value is in milli seconds!!!
# The numbers below were for a Cisco 1750 router running 12.2 code
# 250ms delay throttles CPU usage to 20 - 25%
# 500ms delay throttles CPU usage to 10 - 12%
# Example: 1000ms = 1 second
#
#####
set PING_DELAY_INTERVAL 250

#####
# Data Structure List describing the topology for the pinger utility.
# The first argument in each row is the address/name to telnet to.
# The 2nd (plus) argument for each row must be a valid IP interface address
# on that router.  This address will be used for the source IP address when
# using extended pings.
# Additional source IP addresses can be added after the 2nd list element of each row
# Each new address must be enclosed in double quotes
# The keyword "default" can also be used as a source IP address.  In this
# case the router will use the interface IP address that the ping exits the router.
# For example, the first router that pings will be performed from is nj1.  Pings will
# be sent from this router first using a source IP Address of 64.145.25.3.  The
# addresses that are pinged are contained in the PING_LIST below.  After all of the
# addresses in PING_LIST are pinged, Router nj1 will then ping those same IP
# Addresses but will now use a source IP address of 64.32.127.132. After those
# pings are finished nj1 will then use the router's default source IP address
# and ping all of the address in the PING_LIST
#####
lappend TOPO_LIST [list nj1 "64.145.25.3" "64.32.127.132" "default"]
lappend TOPO_LIST [list nj2 "150.145.25.4" "150.32.127.134"]
lappend TOPO_LIST [list cnj3 "200.33.2.1" "192.145.25.1"]
#lappend TOPO_LIST [list cnj5 "28.126.4.252"]
lappend TOPO_LIST [list cnj7 "10.33.1.3" "10.33.2.3" "10.128.1.1" "10.128.2.1"]
lappend TOPO_LIST [list ny1 "10.134.132.3" "200.32.127.136"]
lappend TOPO_LIST [list ny2 "10.134.132.4" "200.32.127.138"]
lappend TOPO_LIST [list cny3 "172.33.1.1" "10.134.132.1"]
lappend TOPO_LIST [list cal "10.31.240.253" "200.42.59.16" "10.31.239.1"]

#####
# A list the IP addresses that will pinged for testing along with a desription
# of the IP address.  The second element of each row is a description field.
# This can either be left with ??? or a small descriptive comment can be added.
#####
lappend PING_LIST [list "10.145.25.1" "????"]
lappend PING_LIST [list "10.145.25.2" "?????"]
lappend PING_LIST [list "10.145.25.3" "nj1 eth"]
lappend PING_LIST [list "10.145.25.4" "nj2 eth"]
#lappend PING_LIST [list "10.126.4.102" "nj1 eth"]
#lappend PING_LIST [list "10.126.4.103" "nj2 eth"]
#lappend PING_LIST [list "10.126.4.252" "cnj5 eth"]
lappend PING_LIST [list "10.134.132.1" "cny3"]
lappend PING_LIST [list "10.134.132.2" "cny4"]
lappend PING_LIST [list "10.134.132.3" "ny1"]
lappend PING_LIST [list "10.134.132.4" "ny2"]
lappend PING_LIST [list "200.32.127.132" "nj1 10"]
lappend PING_LIST [list "200.32.127.134" "nj2 10"]
lappend PING_LIST [list "200.32.127.136" "ny1 10"]
lappend PING_LIST [list "200.32.127.138" "ny2 10"]
lappend PING_LIST [list "172.33.1.1" "cny3 eth"]
lappend PING_LIST [list "172.33.1.2" "cny4 eth"]
lappend PING_LIST [list "172.33.1.3" "cnj7 eth"]
lappend PING_LIST [list "172.33.2.1" "cnj3 eth"]
lappend PING_LIST [list "172.33.2.2" "cnj4 eth"]
lappend PING_LIST [list "172.33.2.3" "cnj7 eth"]
lappend PING_LIST [list "10.31.240.253" "jpl ether"]

```

```

lappend PING_LIST [list "200.42.59.16" "jpl L0"]
lappend PING_LIST [list "10.31.239.1" "jpl L10"]
lappend PING_LIST [list "10.128.1.1" "cnj7 L10"]
lappend PING_LIST [list "10.128.2.1" "cnj7 L11"]

#####
#####
#####
# OPTIONAL Skip List
#####
#####
# Add combinations of router, src_address, dest_address for pings
# that you do not want to perform
# Here are some examples based on the data above
# This is useful if you know some combinations are supposed to fail and
# you don't want the summary_log file to report a false alarm
#
# First list element is the router name or IP address. This must
# match the router name or IP address that was specified in the first
# element of TOPO_LIST
#   "*" is a wildcard match and will match any router. This only makes sense
#   to use when also setting the source address to a "*"
# Second element is one of the source IP addresses specified in TOPO_LIST
#   "*" is a wildcard match and will match all sources for a given router
# Third element is one of the destination addresses specified in PING_LIST
#   "*" is a wildcard match and will match all destinations for a given router
# By default, the SKIP_LIST is commented out. If you want to skip
# some ping combinations, then uncomment the lines below and modify the
# router name and IP addresses to fit your environment
#
# Since there is a * in the source field for the third example, this in essence
# is blocking all pings from the cnj7 router to 172.33.1.2
#
# The last example says block all pings to 10.31.239.1 from any router.
# This is basically the same as commenting out a single line in the PING_LIST variable
#####
#lappend SKIP_LIST [list "nj2" "150.32.127.134" "10.128.2.1"]
#lappend SKIP_LIST [list "nj1" "default" "172.33.1.2"]
#lappend SKIP_LIST [list "cnj7" "*" "200.32.127.132"]
#lappend SKIP_LIST [list "*" "*" "10.31.239.1"]

#####
#####
#####
# OPTIONAL Pre-Ping Command List
#####
#####
# Optional feature which will issue specified commands immediately before
# each ping
# 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 two
# 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. TRACEROUTE - This command will perform a traceroute to the
#   address that will be used in the ping.
#   Do not specify an ip address next to this command
#   MUST BE ALL CAPITAL LETTERS
#
# Inserting the KEY words SRC or DST will substitute the
# source/destination of the ping that will be issued for the actual ping
# The key word TRACEROUTE will perform a traceroute to the destination
# Leave these commands commented out if you do not wanted to issue
# any commands following a before the ping is performed.

```

```
#####
#lappend PRE_PING_CMD_LIST "show ip route DST"
#lappend PRE_PING_CMD_LIST "show ip eigrp neighbors"
#lappend PRE_PING_CMD_LIST "show ip cef DST"
#lappend PRE_PING_CMD_LIST "TRACEROUTE"

#####
#####
#####
# OPTIONAL Diagnostic Command List
#####
#####
# Commands to enter on a router in the event a ping fails
# Inserting the KEY words SRC or DST will substitute the
# source/destination of the failed ping 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 ping.
#####
#lappend DIAG_CMD_LIST "show ip route"
#lappend DIAG_CMD_LIST "show ip route DST"
#lappend DIAG_CMD_LIST "show ip eigrp neighbors"
#lappend DIAG_CMD_LIST "show ip cef DST"
#lappend DIAG_CMD_LIST "TRACEROUTE"
```

4.6.3 Pinger VRF Template File

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

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

#####
# Delay before performing next ping. If there are alot of ping destinations
# it could drive up the CPU on low end routers. Putting in a delay between
# pings should lower the CPU. This value is in milli seconds!!!
# The numbers below were for a Cisco 1750 router running 12.2 code
# 250ms delay throttles CPU usage to 20 - 25%
# 500ms delay throttles CPU usage to 10 - 12%
# Example: 1000ms = 1 second
#
#####
set PING_DELAY_INTERVAL 250

#####
# Data Structure List describing the topology for the pinger utility.
# The first argument in each row is the address/name to telnet to.
# NEW for VRF funtionality. The 2nd argument must be a valid VRF name
# if the VRF_PINGS variable is defined at the top of this file
# The 3rd (plus) argument for each row must be a valid IP interface address
# on that router. This address will be used for the source IP address when
# using extended pings.
# Additional source IP addresses can be added after the 3rd list element of each row
# Each new address must be enclosed in double quotes
# The keyword "default" can also be used as a source IP address. In this
# case the router will use the interface IP address that the ping exits the router.
# For example, the first router that pings will be peformed from is nj1. Pings will
```

```

# be sent from this router first using a source IP Address of 64.145.25.3. The
# addresses that are pinged are contained in the PING_LIST below. After all of the
# addresses in PING_LIST are pinged, Router nj1 will then ping those same IP
# Addresses but will now use a source IP address of 64.32.127.132. After those
# pings are finished nj1 will then use the router's default source IP address
# and ping all of the address in the PING_LIST
#####
lappend TOPO_LIST [list nj1 "vrf1" "64.145.25.3" "64.32.127.132" "default"]
lappend TOPO_LIST [list nj2 "vrf1" "150.145.25.4" "150.32.127.134"]
lappend TOPO_LIST [list cnj3 "vrf2" "200.33.2.1" "192.145.25.1"]
#lappend TOPO_LIST [list cnj5 "vrf2" "28.126.4.252"]
lappend TOPO_LIST [list cnj7 "vrf2" "10.33.1.3" "10.33.2.3" "10.128.1.1" "10.128.2.1"]
lappend TOPO_LIST [list ny1 "vrf3" "10.134.132.3" "200.32.127.136"]
lappend TOPO_LIST [list ny2 "vrf3" "10.134.132.4" "200.32.127.138"]
lappend TOPO_LIST [list cny3 "vrf4" "172.33.1.1" "10.134.132.1"]
lappend TOPO_LIST [list cal "vrf4" "10.31.240.253" "200.42.59.16" "10.31.239.1"]

#####
# A list the IP addresses that will pinged for testing along with a description
# of the IP address. The second element of each row is a description field.
# This can either be left with ??? or a small descriptive comment can be added.
#####
lappend PING_LIST [list "10.145.25.1" "???"]
lappend PING_LIST [list "10.145.25.2" "?????"]
lappend PING_LIST [list "10.145.25.3" "nj1 eth"]
lappend PING_LIST [list "10.145.25.4" "nj2 eth"]
#lappend PING_LIST [list "10.126.4.102" "nj1 eth"]
#lappend PING_LIST [list "10.126.4.103" "nj2 eth"]
#lappend PING_LIST [list "10.126.4.252" "cnj5 eth"]
lappend PING_LIST [list "10.134.132.1" "cny3"]
lappend PING_LIST [list "10.134.132.2" "cny4"]
lappend PING_LIST [list "10.134.132.3" "ny1"]
lappend PING_LIST [list "10.134.132.4" "ny2"]
lappend PING_LIST [list "200.32.127.132" "nj1 10"]
lappend PING_LIST [list "200.32.127.134" "nj2 10"]
lappend PING_LIST [list "200.32.127.136" "ny1 10"]
lappend PING_LIST [list "200.32.127.138" "ny2 10"]
lappend PING_LIST [list "172.33.1.1" "cny3 eth"]
lappend PING_LIST [list "172.33.1.2" "cny4 eth"]
lappend PING_LIST [list "172.33.1.3" "cnj7 eth"]
lappend PING_LIST [list "172.33.2.1" "cnj3 eth"]
lappend PING_LIST [list "172.33.2.2" "cnj4 eth"]
lappend PING_LIST [list "172.33.2.3" "cnj7 eth"]
lappend PING_LIST [list "10.31.240.253" "jp1 ether"]
lappend PING_LIST [list "200.42.59.16" "jp1 L0"]
lappend PING_LIST [list "10.31.239.1" "jp1 L10"]
lappend PING_LIST [list "10.128.1.1" "cnj7 L10"]
lappend PING_LIST [list "10.128.2.1" "cnj7 L11"]

#####
#####
#####
# OPTIONAL Skip List
#####
#####
# Skip List
# Add combinations of router, src_address, dest_address for pings
# that you do not want to perform
# Here are some examples based on the data above
# This is useful if you know some combinations are supposed to fail and
# you don't want the summary_log file to report a false alarm
#
# First list element is the router name or IP address. This must
# match the router name or IP address that was specified in the first
# element of TOPO_LIST
# "*" is a wildcard match and will match any router. This only makes sense

```

```

#   to use when also setting the source address to a "*"
# Second element is the vrf name as specified in TOPO_LIST
#   "*" is a wildcard match and will match all vrfs for a given router
# Third element is one of the source IP addresses specified in TOPO_LIST
#   "*" is a wildcard match and will match all sources for a given router
# Forth element is one of the destination addresses specified in PING_LIST
#   "*" is a wildcard match and will match all destinations for a given router
# By default, the SKIP_LIST is commented out.  If you want to skip
# some ping combinations, then uncomment the lines below and modify the
# router name and IP addresses to fit your environment
#
# Since there is a * in the source field for the third example, this in essence
# is blocking all pings from the cnj7 router to 172.33.1.2
#
# The last example says blocking all pings to 10.31.239.1 from any router, withing any vrf.
# This is basically the same as commenting out a single line in the PING_LIST variable
#####
#lappend SKIP_LIST [list "nj2" "vrf1" "150.32.127.134" "10.128.2.1"]
#lappend SKIP_LIST [list "nj1" "vrf1" "default" "172.33.1.2"]
#lappend SKIP_LIST [list "cnj7" "vrf2" "*" "200.32.127.132"]
#lappend SKIP_LIST [list "*" "*" "*" "10.31.239.1"]

#####
#####
#####
# OPTIONAL Pre-Ping Command List
#####
#####
# Optional feature which will issue specified commands immediately before
# each ping
# 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 two
# 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. TRACEROUTE - This command will perform a traceroute to the
#   address that will be used in the ping.
#   Do not specify an ip address or vrf name next to this command
#   MUST BE ALL CAPITAL LETTERS
#
# Inserting the KEY words SRC, DST, or VRF will substitute the
# source/destination of the ping that will be issued for the actual ping
# The key word TRACEROUTE will perform a traceroute to the destination
# Leave these commands commented out if you do not wanted to issue
# any commands following a before the ping is performed.
#####
#lappend PRE_PING_CMD_LIST "show ip route VRF DST"
#lappend PRE_PING_CMD_LIST "show ip eigrp neighbors"
#lappend PRE_PING_CMD_LIST "show ip cef DST"
#lappend PRE_PING_CMD_LIST "TRACEROUTE"

#####
#####
#####
# OPTIONAL Diagnostic Command List
#####
#####
# Commands to enter on a router in the event a ping fails
# Inserting the KEY words SRC, DST or VRF will substitute the
# source/destination or vrf of the failed ping 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 ping.
#####
#lappend DIAG_CMD_LIST "show ip route vrf VRF"
#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"
#lappend DIAG_CMD_LIST "TRACEROUTE"
```

4.6.4 Tracer Template File

```
#####
# 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

#####
# Note, the yes/no field is whether or not the traceroute should be successful.
# 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"]

#####
# 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"]

#####
# 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 specified.
#####
#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 specified.
#####
#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"

```

4.6.5 Tracer VRF Template File

```

#####
# 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

#####
# 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"]

#####
# 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"]

#####
# 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 specified. Do not
#   enter the vrf name, the script will automatically 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 specified.
#####
#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"

```

4.6.6 Cisco Password Change Template File

```

# 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

```

4.6.7 BGP Attribute Checker Template File

```

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

# Define BGP route prefixes to be tested

# Scenario: Normal Steady State

#####
# 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          mask          yes/no next_hop          metric pref origin a
#####
lappend PREFIX_LIST_1 [list "10.134.134.0" "255.255.255.0" "yes" "0.0.0.0" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_1 [list "10.45.45.1" "255.255.255.255" "yes" "0.0.0.0" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_1 [list "10.25.25.1" "255.255.255.255" "yes" "172.100.20.2" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_1 [list "10.40.40.1" "255.255.255.255" "yes" "172.16.20.5" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_1 [list "10.30.30.1" "255.255.255.255" "yes" "192.158.10.21" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_1 [list "10.60.60.1" "255.255.255.255" "yes" "192.158.10.21" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_1 [list "10.50.50.1" "255.255.255.255" "no" "192.158.10.21" "N/A" "100" "IGP" "1

#####
# For router: SJ2
# Variables          subnet          mask          yes/no next_hop          metric pref origin a
#####
lappend PREFIX_LIST_2 [list "10.134.134.0" "255.255.255.0" "yes" "0.0.0.0" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_2 [list "10.45.45.1" "255.255.255.255" "yes" "172.100.20.1" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_2 [list "10.25.25.1" "255.255.255.255" "yes" "0.0.0.0" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_2 [list "10.40.40.1" "255.255.255.255" "yes" "172.100.20.1" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_2 [list "10.30.30.1" "255.255.255.255" "yes" "172.100.20.1" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_2 [list "10.60.60.1" "255.255.255.255" "yes" "172.100.20.1" "N/A" "100" "IGP" "1
lappend PREFIX_LIST_2 [list "10.50.50.1" "255.255.255.255" "no" "192.158.10.2" "N/A" "100" "IGP" "1

#####
# For router: 192.168.1.40 (3745)
# Variables          subnet          mask          yes/no next_hop          metric pref origin a
#####
lappend PREFIX_LIST_3 [list "10.134.134.0" "255.255.255.0" "yes" "192.158.10.101" "N/A" "100" "IGP" "1
"]
lappend PREFIX_LIST_3 [list "10.45.45.1" "255.255.255.255" "yes" "192.158.10.101" "N/A" "100" "IGP" "1
"]
lappend PREFIX_LIST_3 [list "10.25.25.1" "255.255.255.255" "yes" "192.158.10.101" "N/A" "100" "IGP" "1
"]

```

```

"]
lappend PREFIX_LIST_3 [list "10.40.40.1" "255.255.255.255" "no" "192.158.10.101" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_3 [list "10.30.30.1" "255.255.255.255" "yes" "0.0.0.0" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_3 [list "10.60.60.1" "255.255.255.255" "yes" "192.158.10.26" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_3 [list "10.50.50.1" "255.255.255.255" "yes" "192.158.10.26" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_3 [list "10.70.70.1" "255.255.255.255" "yes" "192.158.10.101" "N/A" "100" "IGP" "1"

#####
# For router: 192.168.1.27
# Variables          subnet          mask          yes/no next_hop      metric pref origin a
#####
lappend PREFIX_LIST_4 [list "10.134.134.0" "255.255.255.0" "yes" "192.158.10.42" "N/A" "100" "IGP" "1"
]
lappend PREFIX_LIST_4 [list "10.45.45.1" "255.255.255.255" "yes" "192.158.10.42" "N/A" "100" "IGP" "1"
]
lappend PREFIX_LIST_4 [list "10.25.25.1" "255.255.255.255" "yes" "192.158.10.42" "N/A" "100" "IGP" "1"
]
lappend PREFIX_LIST_4 [list "10.40.40.1" "255.255.255.255" "no" "192.158.10.42" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_4 [list "10.30.30.1" "255.255.255.255" "yes" "192.158.10.34" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_4 [list "10.60.60.1" "255.255.255.255" "yes" "0.0.0.0" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_4 [list "10.50.50.1" "255.255.255.255" "yes" "192.158.10.34" "N/A" "100" "IGP" "1"

#####
# For router: Stuper (VPN_A)
# Variables          subnet          mask          yes/no next_hop      metric pref origin a
#####
lappend PREFIX_LIST_5 [list "10.134.134.0" "255.255.255.0" "yes" "172.16.20.6" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_5 [list "10.45.45.1" "255.255.255.255" "yes" "172.16.20.6" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_5 [list "10.25.25.1" "255.255.255.255" "yes" "175.20.20.5" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_5 [list "10.40.40.1" "255.255.255.255" "yes" "175.20.20.5" "N/A" "100" "IGP" "1"

#####
# For router: Stuper (VPN_BB)
# Variables          subnet          mask          yes/no next_hop      metric pref origin a
#####
lappend PREFIX_LIST_6 [list "10.30.30.1" "255.255.255.255" "yes" "182.30.30.65" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_6 [list "10.60.60.1" "255.255.255.255" "yes" "182.30.30.65" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_6 [list "10.50.50.1" "255.255.255.255" "no" "182.30.30.65" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_6 [list "10.70.70.1" "255.255.255.255" "yes" "182.30.30.65" "N/A" "100" "IGP" "1"

#####
# For router: Paulper (VPN_A)
# Variables          subnet          mask          yes/no next_hop      metric pref origin a
#####
lappend PREFIX_LIST_7 [list "10.134.134.0" "255.255.255.0" "yes" "175.20.20.6" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_7 [list "10.45.45.1" "255.255.255.255" "yes" "175.20.20.6" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_7 [list "10.25.25.1" "255.255.255.255" "yes" "182.30.30.101" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_7 [list "10.40.40.1" "255.255.255.255" "yes" "182.30.30.41" "N/A" "100" "IGP" "1"

#####
# For router: Paulper (VPN_BB)
# Variables          subnet          mask          yes/no next_hop      metric pref origin a
#####
lappend PREFIX_LIST_8 [list "10.30.30.1" "255.255.255.255" "yes" "175.20.20.6" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_8 [list "10.60.60.1" "255.255.255.255" "yes" "175.20.20.6" "N/A" "100" "IGP" "1"
lappend PREFIX_LIST_8 [list "10.50.50.1" "255.255.255.255" "no" "175.20.20.6" "N/A" "100" "IGP" "1"

#####
# 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. Does this router have a vrfs? yes for PERs. No for CE routers.
#    This basically states whether the script needs to enter
#    "show ip bgp" or "show ip v v vpn_name bgp"
# 3. VPN name on PER router. This is only relevant if the 2nd field is "yes"
# 4. The associated Prefixes to check on this router

```

```

#                               rtr vrf? vpn_name prefix list
#####
lappend RTR_LIST [list "br1" "no" "VPN_A" "$PREFIX_LIST_1"]
lappend RTR_LIST [list "sj2" "no" "VPN_A" "$PREFIX_LIST_2"]
lappend RTR_LIST [list "192.168.1.40" "no" "VPN_A" "$PREFIX_LIST_3"]
lappend RTR_LIST [list "192.168.1.27" "no" "VPN_A" "$PREFIX_LIST_4"]
lappend RTR_LIST [list "stuper" "yes" "VPN_A" "$PREFIX_LIST_5"]
lappend RTR_LIST [list "stuper" "yes" "VPN_BB" "$PREFIX_LIST_6"]
lappend RTR_LIST [list "paulper" "yes" "VPN_A" "$PREFIX_LIST_7"]
lappend RTR_LIST [list "paulper" "yes" "VPN_BB" "$PREFIX_LIST_8"]

```

4.6.8 Generic Send Commands Template File

```

#####
# 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 "# "

#####
# 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 "%"

```

4.6.9 Cisco Send Commands RCF

```

#####
# 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

```

4.6.10 Cisco Config Command Sender RCF

```

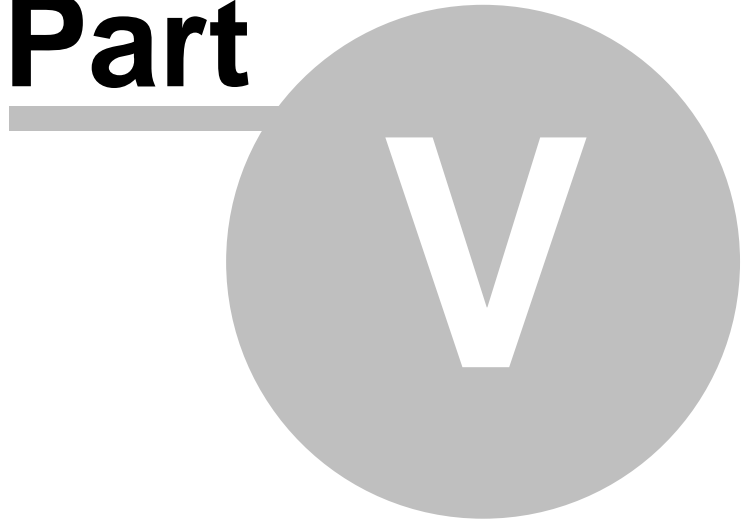
# Sample template file for config_devices_rcf script
# Lines that begin with a # are comments
# Format of file is router:filename
# Where filename contains a list of configuration
# commands that will only be sent to that router

nyrtr1:nyrtr1.cmds
10.1.1.1:cmd_file1.cmds

```

Caveats

Part



5 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:

- ***config_devices***
- ***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
```

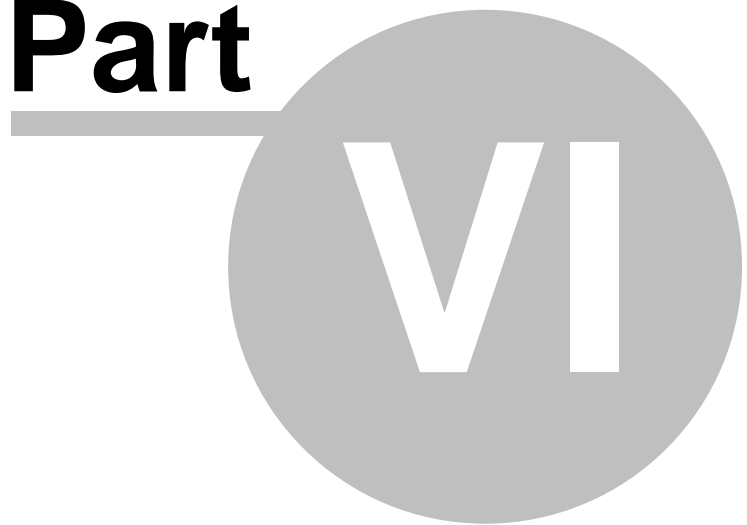
Below shows a sample output in configuration mode when there is not a prompt in config mode:

```
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 ***config_devices*** script to do this. With the ***config_devices*** 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

Part



6 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.

Endnotes 2... (after index)

Back Cover