Contents

Remote VTY Command Script

Running one or more interactive CLI commands is a common task in network management. Having a script to automate this task becomes a critical tool.

This script automates interactive TELNET or SSH sessions. Any command that can be issued in CLI can be issued via the VTY run command script in an automated fashion. Below are some example uses:

- Run a set of commands on hundreds of devices
- Configuration changes
- Archive configurations and software
- Audits, inventory and discovery
- Automated troubleshooting

Requirements

This script is written in Expect. A current version of Expect is required. Expect and installation requirements for Expect can be found at Expect Homepage

Tested on:

- RHEL 5 running expect 5.43.0-5.1
- Mac OS X 10.6 running Expect 5.44.1.11
- Mac OS X 10.8.5 running Expect 5.45

The script is platform independent and should run on any platform that can run Expect.

VTY Run Command Script

The script supports command line options for configuration. Below is a list of the command line options available:

Note: Run the script using **vty_runcmd.exp <options>**

Required Options

Option	Option Value	Description
-h	Hostname or IP	Enter the hostname or IP address of the device to run the remote command on
-u	username	Enter the username for login or username prompt, normally required for Unix or AAA logins. Use none if no login/username required
-p	password	Enter the password for the firs login/username prompt. This is the first password, non enable mode

Optional Options

Option	Option Value	Description
-e	password	Enter the enable password or secret. This is to enter enable mode.
-t	seconds	Enter the timeout for expect wait in seconds
-m	ssh or telnet	Enter **ssh** or **telnet** for connection method. TELNET is the default. For SSH, the script will automatically enter **yes** for authorized host entries.
-f	filename	Enter the filename to use as the command file. Default is to use STDIN

Automating script to run on multiple hosts

See Automated Remote VTY Command Script for more details on how to run the script via **cron** and how to run it in a loop fashion for multiple hosts.

Steps to install the script

1. Cut and paste the below script into a text file named **vty_runcmd.exp**
2. Edit the first line of the script and update the "**#!/usr/bin/expect --**" command to point to your installed **expect** binary
3. (Unix,Linux,Mac OS only) change the mode so that the script will run by using "**chmod 755 vty_runcmd.exp**"
4. Run the script by using "**./vty_runcmd.exp <options>**"

Script source

```
#!/usr/bin/expect --
##############################################################################
# vty_runcmd.exp
#
#    This is an expect script that will connect to a device using ssh or
#    telnet, then run the specified command.  The output will either be
#    written to an output file or printed to stdout.
#
# Created by Tim Evens (tim@evensweb.com), 5/2009
# Corrected by Sergio Zavala (sergio.zavala at sidetec.com.mx), 12/2010
#
# Copyright Tim Evens, 2009
#
# HISTORY:
#   Tim Evens   5/21/09 Initial program created
#   Sergio Zavala       12/26/10 Connection return code corrected
#   Tim Evens   1/05/10 Connection validated
```

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

#++++++++++++++++++++++++++
# Global vars
#++++++++++++++++++++++++++
set timeout 30
# below matches prompts such as "router#", "router>", "router$"
set prompt "\>\ *$|#\ *$|\\$\ *$"


#-------------------------------------------
# Connect(method, prompt, host, user, password)
#    This function will connect to a host using telnet or ssh.
#
# RETURNS:
#    zero if successful
#    1 = timeout or invalid hostname or method
#    2 = invalid login
#    3 = timeout waiting for login
#    4 = connection failed to host during expect wait
#    9 = unknown error
#-------------------------------------------
proc Connect {method host usr pw} {
   set rval 0
   set usr_chk 0
   set pw_chk 0
   set max_checks 4
   global spawn_id
   global timeout
   global prompt

   puts "Connecting using $method to $host as user $usr"

   # see if we are using ssh
   if { [string compare $method "ssh"] == 0 } {
      set host "$usr@$host"
   }

   # Run command and get connected
   set id [spawn $method $host]
   if { $id <= 0 } {
      puts "ERROR: Failed to connect to host\n"
      set rval 1

   } else {
      puts "Using Process ID: $id"
   }

   # Start the expect/send process to login
   expect {

      # Below handles the username prompt
      -nocase -re "name:|^login:" {
         send "$usr\r"
         incr usr_chk;

         # continue with expect loop as long as we haven't hit this too many times
         if { $usr_chk < $max_checks } {
            exp_continue
         } else {
            set rval 2
            puts "ERROR: Login retry failed.  Invalid login username"
         }
```

```
        # Below handles the password prompt
        } -nocase -re "word:" {
          send "$pw\r"
          incr pw_chk;

          # continue with expect loop as long as we haven't hit this too many times
          if { $pw_chk < $max_checks } {
             exp_continue
          } else {
             set rval 2
             puts "ERROR: Login retry failed.  Invalid login password"
          }

       # Below handles the yes/no prompt when SSH first connects to a host
       } -nocase -re "\(yes/no\)" {
          send "yes\r"
          exp_continue

       # Below handles the normal prompt to detect when logged in
       } -nocase -re "$prompt" {
          puts "\nSUCCESS: Logged in and ready to send commands\n"

       # Below is for expect timeout waiting for a
       } timeout {
          puts "ERROR: Connection timeout waiting for login prompt"
          set rval 3

       # Below is for when the connect is closed before finishing
       } eof {
          puts "ERROR: Connection to host failed: $expect_out(buffer)"
          set rval 4
       }
    }

    # return with error code
    return $rval
}
# End of Connect ()

#--------------------------------------------
# Usage()
#     This function will print the usage
#--------------------------------------------
proc Usage {} {

    puts "Usage: vty_runcmd.exp <options>"
    puts "\n"
    puts "REQUIRED OPTIONS:"
    puts "   -h <hostname|ip>   = hostname or ip address"
    puts "   -u <username>      = username to login with"
    puts "   -p <password>      = password for login"
    puts "\n"
    puts "Other OPTIONS:"
    puts "   -e <enable password> = Enable password"
    puts "   -t <seconds>         = timeout in seconds"
    puts "   -m <ssh|telnet>      = use either ssh or telnet, default telnet"
    puts "   -f <filename>        = command file, defaults to STDIN"
    puts "\n"
}
# End of Check_ARGS()

#--------------------------------------------
# main()
#
```

```
# RETURNS:
#    0 if successful
#    1 if invalid arg passed
#    2 not enough args (required args not met)
#--------------------------------------------
   set rval 0
   set hostname ""
   set username ""
   set password ""
   set enable_pw ""
   set cmdfile ""
   set method "telnet"

   # Loop through the command line args
   for {set n 0} {$n < $argc} {incr n} {

      set arg [lindex $argv $n]

      # Check the args
      if { [string compare $arg "-u"] == 0} {
         if { $n < $n+1 } {
            incr n
            set username [lindex $argv $n]
         } else {
            set rval 1
            puts "ERROR: Missing ARG for $arg\n"
         }

      } elseif { [string compare $arg "-p"] == 0} {
         if { $n < $n+1 } {
            incr n
            set password [lindex $argv $n]
         } else {
            set rval 1
            puts "ERROR: Missing ARG for $arg\n"
         }

      } elseif { [string compare $arg "-h"] == 0} {
         if { $n < $n+1 } {
            incr n
            set hostname [lindex $argv $n]
         } else {
            set rval 1
            puts "ERROR: Missing ARG for $arg\n"
         }

      } elseif { [string compare $arg "-m"] == 0} {
         if { $n < $n+1 } {
            incr n
            set method [lindex $argv $n]

         } else {
            set rval 1
            puts "ERROR: Missing ARG for $arg\n"
         }

      } elseif { [string compare $arg "-t"] == 0} {
         if { $n < $n+1 } {
            incr n
            set timeout [lindex $argv $n]
         } else {
            set rval 1
            puts "ERROR: Missing ARG for $arg\n"
         }
```

```
   } elseif { [string compare $arg "-f"] == 0} {
      if { $n < $n+1 } {
         incr n
         set cmdfile [lindex $argv $n]
      } else {
         set rval 1
         puts "ERROR: Missing ARG for $arg\n"
      }

   } elseif { [string compare $arg "-e"] == 0} {
      if { $n < $n+1 } {
         incr n
         set enable_pw [lindex $argv $n]
      } else {
         set rval 1
         puts "ERROR: Missing ARG for $arg\n"
      }
   }
}
# End of arg check

# make sure we found the amount of args expected
if { [llength $hostname] > 0 && [llength $method] > 0 &&
       [llength $username] > 0 && [llength $password] > 0 } {
   # Print out the args found
   puts "hostname = $hostname, user = $username, pw = $password, method = $method"

} else {
   set rval 2
   puts "ERROR: Missing required args, must have -h, -u, -p\n"
   Usage
}

# ------------------
# Now that we have the correct ARGS and we know what to do, lets proceed to
#     connect, run the commands, then exit.
# ------------------

# make sure we have not encountered any errors
if { $rval <= 0 } {

   if { [llength $cmdfile] <= 0 } {
      puts "Enter the send text (type 'end' on last line to finish):"
      expect_user -nocase -re "(.*)\nend\n"
      set send_text $expect_out(1,string)

   } else {
      puts "Using $cmdfile for send text"
      # set cmdfile_fd [open $cmdfile r]
      if { [catch {set cmdfile_fd [open $cmdfile r]} err_msg] } {
         puts stderr "Could not open $cmdfile for reading\n$err_msg"
         exit 1
      }

      # read in the file info - warning there is a limit on the size
      set send_text [read $cmdfile_fd 10000]

      # close open file
      close $cmdfile_fd
   }

   # connect and check return status before proceeding
   if { [Connect "$method" "$hostname" "$username" "$password"] > 0 } {
```

```
   # stop here,  no need to print an error since Connect func does that
   exit 1
}


# If we have an enable password, lets try to send it
if { [llength $enable_pw] > 0} {
   puts "***Using enable mode"
   send "enable\r"
   expect {
      -timeout 3
      # Below handles the password prompt
      -nocase -re "word:" {
         send "$enable_pw\r"
         exp_continue

      # Below handles the normal prompt to detect when logged in
      } -re "#\ *$" {
         puts "--SUCCESS on enable mode--"

      # Below is for expect timeout waiting for a
      } timeout {
         puts "ERROR: Enable password timeout"
         set rval 3

      # Below is for when the connect is closed before finishing
      } eof {
         puts "ERROR: Connection to host failed: $expect_out(buffer)"
         set rval 4
      }
   }
}

# Loop through the send_text and send one line at a time
foreach line [split $send_text \n] {
   # Make sure to exclude empty lines
   if { [llength $line] > 0 } {
      send "$line\r"

      # Start the expect/send process to login
      expect {
         # Below handles the yes/no prompts
         -nocase -re "\(yes/no\)" {
            send "yes\r"
            exp_continue

         # Below handles the y/n prompts
         } -nocase -re "\(yes/no\)" {
            send "yes\r"
            exp_continue

         # Below handles the y/n prompts
         } -nocase -re "--more--" {
            send " "
            exp_continue

         # Below handles the normal prompt to detect when logged in
         } -nocase -re "$prompt" {
            puts "\n--SUCCESS for normal login prompt--\n"
         }
      }

   }
}
```

```
      # Now that we are done, send an exit
      puts "*** Finished with script"
      send "exit\r"
      sleep 1
   }

## END OF SCRIPT #################################################################
```

Examples

ASA show version using TELNET

```
[network@network bin]$ ./vty_runcmd.exp -h 172.16.0.33 -u tim -p userpass -e enablepass
hostname = 172.16.0.33, user = tim, pw = userpass, method = telnet
Enter the send text (type 'end' on last line to finish):
show version
end
Connecting using telnet to 172.16.0.33 as user tim
spawn telnet 172.16.0.33
Using Process ID: 13811
Trying 172.16.0.33...
Connected to  FP-N1-7-FWSM-P1 (172.16.0.33).
Escape character is '^]'.
-=====================================================================-
| Unauthorized access is prohibited!                                  |
|                                                                     |
-=====================================================================-


User Access Verification

Warning: Authentication is enabled for system context. Use admin context credentials.

Username: tim
Password: ********
Type help or '?' for a list of available commands.
FP-N1-7-FWSM-P1/PartnerFW>
SUCCESS: Logged in and ready to send commands

***Using enable mode
enable
Password: ********
FP-N1-7-FWSM-P1/PartnerFW# --SUCCESS on enable mode--
show version

FWSM Firewall Version 4.0(6) <context>
Device Manager Version 6.1(5)F

Compiled on Mon 22-Jun-09 02:33 by fwsmbld

FP-N1-7-FWSM-P1 up 55 days 11 hours
failover cluster up 116 days 12 hours

Hardware:   WS-SVC-FWM-1
The Running Activation Key is not set, using default settings:

Licensed features for this user context:
Failover                    : Active/Active
VPN-DES                     : Enabled
VPN-3DES-AES                : Enabled
GTP/GPRS                    : Disabled
```

```
BGP Stub                    : Disabled
Service Acceleration        : Disabled

Configuration last modified by tim at 23:42:12.528 UTC Mon Nov 1 2009
FP-N1-7-FWSM-P1/PartnerFW#
--SUCCESS for normal login prompt--

*** Finished with script
[network@network bin]$
```

ASA show version using SSH - automatically accepts RSA key

```
[network@network ~]$ bin/vty_runcmd.exp -m ssh -h 172.16.0.33 -u tim -p userpass -e enablepass
hostname = 172.16.0.33, user = tim, pw = userpass, method = ssh
Enter the send text (type 'end' on last line to finish):
show ver
end
Connecting using ssh to 172.16.0.33 as user tim
spawn ssh tim@172.16.0.33
Using Process ID: 14855
The authenticity of host '172.16.0.33 (172.16.0.33)' can't be established.
RSA key fingerprint is 1d:81:48:7e:ec:5a:ba:e1:f5:e5:a3:92:41:a8:6a:61.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.0.33' (RSA) to the list of known hosts.
tim@172.16.0.33's password:
-=====================================================================-
| Unauthorized access is prohibited!                                  |
|                                                                     |
-=====================================================================-
Type help or '?' for a list of available commands.
FP-N1-7-FWSM-P1/PartnerFW>
SUCCESS: Logged in and ready to send commands

***Using enable mode
enable
Password: ********
FP-N1-7-FWSM-P1/PartnerFW# --SUCCESS on enable mode--
show ver

FWSM Firewall Version 4.0(6) <context>
Device Manager Version 6.1(5)F

Compiled on Mon 22-Jun-09 02:33 by fwsmbld

FP-N1-7-FWSM-P1 up 55 days 11 hours
failover cluster up 116 days 12 hours

Hardware:   WS-SVC-FWM-1
The Running Activation Key is not set, using default settings:

Licensed features for this user context:
Failover                    : Active/Active
VPN-DES                     : Enabled
VPN-3DES-AES                : Enabled
GTP/GPRS                    : Disabled
BGP Stub                    : Disabled
Service Acceleration        : Disabled

Configuration last modified by enable_15 at 06:05:31.674 UTC Mon Nov 1 2009
FP-N1-7-FWSM-P1/PartnerFW#
--SUCCESS for normal login prompt--

*** Finished with script
```

```
[network@network ~]$
```