1 Router Auto-Configuration Generator

This script generates router configurations automatically by reading parameters from a comma separated CSV file called “input” and inserting them into a template file called “config.txt”. The template file is used to produce target configurations in the sub-directory “configdir”.

Each record in the CSV “input” file contains the parameters to substitute for a single router configuration. The number of records in the CSV file determines how many target configurations are produced.

It is up to you how many parameters you specify for substitution. Please take care to specify the same number of parameters in the CSV file as there are <<Pn>> substitution flags in the template file. The number of parameters to substitute is passed to the generator script as an argument along with the name of the CSV file.

Two versions of the “generate” script are shown - simple and using variable indirection. Both work equally well so it is up to you which you use.

1.1 Step 1

Cut and paste the “generate” script into a directory on your Linux machine and set it to execute mode:

```
root@tonylaptop:~/# chmod u+x generate
```

1.2 Step 2

Create a sub-directory called “configdir”.

```
root@tonylaptop:~/# mkdir configdir
```

1.3 Step 3

Create the template file “config.txt” using a text editor identifying those parameters that you wish to substitute using the notation <<P1>>, <<P2>>……<<Pn>>. Place the template file in the same directory as the “generate” script. A sample extract of the template file “config.txt” is shown below:

```
hostname <<P1>>
!
enable secret <<P2>>
!
clock timezone GMT 0 0
clock summer-time BST recurring last Sun Mar 2:00 last Sun Oct 2:00
!
no ip bootp server
no ip domain lookup
ip domain name <<P3>>
!
!
username <<P4>> password <<P5>>
!
ip ftp source-interface Loopback<<P6>>
ip tftp source-interface Loopback<<P6>>
!
interface Loopback<<P6>>
 ip address <<P7>> 255.255.255.255
```

```
!
interface GigabitEthernet<<P8>>
 bandwidth 20000
 no ip address
 no ip redirects
 no ip proxy-arp
 duplex auto
 speed auto
 no cdp enable
.
.
etc.
```

1.4 Step 4

Create a comma separated CSV file called “input” from a spreadsheet that contains the parameters you wish to substitute and place it in the same directory as the “generate” script and “config.txt” template file. Each record in the CSV file contains the substitution parameters for a single router configuration.

An extract of a sample “input” CSV file is shown below containing 25 x parameters where the “<<Pn>>” parameter fields are:

P1 = router name	**P6 = loopback no.**	**P11 = VLAN mask**	**P16 = local AS number**	**P21 = SNMP mask**
P2 = enable password	**P7 = loopback IP**	**P12 = 2nd GE interface**	**P17 = neighbour IP**	**P22 = SNMP RO community**
P3 = domain	**P8 = 1st GE interface**	**P13 = 2nd GE VLAN**	**P18 = neighbour AS**	**P23 = SNMP RW community**
P4 = username	**P9 = 1st GE VLAN**	**P14 = VLAN IP**	**P19 = logging host**	**P24 = SNMP trap IP**
P5 = password	**P10 = VLAN IP**	**P15 = VLAN mask**	**P20 = SNMP access list IP**	**P25 = SNMP trap mask**

```
Router1,password,network.co.uk,fred,bloggs,1,192.168.1.1,0/0,100,10.1.1.1,2
55.255.255.0,0/1,200,172.16.1.1,255.255.0.0,65001,10.1.1.2,65010,192.168.1.
10,192.168.1.0,0.0.0.255,public,private,192.168.1.20,trapprivate
Router2,password,network.co.uk,fred,bloggs,1,192.168.1.2,0/0,100,10.1.2.1,2
55.255.255.0,0/1,200,172.16.2.1,255.255.0.0,65002,10.1.2.2,65010,192.168.1.
10,192.168.1.0,0.0.0.255,public,private,192.168.1.20,trapprivate
Router3,password,network.co.uk,fred,bloggs,1,192.168.1.3,0/0,100,10.1.3.1,2
55.255.255.0,0/1,200,172.16.3.1,255.255.0.0,65003,10.1.3.2,65010,192.168.1.
10,192.168.1.0,0.0.0.255,public,private,192.168.1.20,trapprivate
Router4,password,network.co.uk,fred,bloggs,1,192.168.1.4,0/0,100,10.1.4.1,2
55.255.255.0,0/1,200,172.16.4.1,255.255.0.0,65004,10.1.4.2,65010,192.168.1.
10,192.168.1.0,0.0.0.255,public,private,192.168.1.20,trapprivate
Router5,password,network.co.uk,fred,bloggs,1,192.168.1.5,0/0,100,10.1.5.1,2
55.255.255.0,0/1,200,172.16.5.1,255.255.0.0,65005,10.1.5.2,65010,192.168.1.
10,192.168.1.0,0.0.0.255,public,private,192.168.1.20,trapprivate
```

1.5 Step 5

Run the “generate” script specifying the CSV file name “input” and number of parameters to substitute as arguments:

```
root@tonylaptop:~/# ./generate input 25
```

1.6 Step 6

Retrieve the target configurations from the “configdir” sub-directory. Each configuration is named “config-<num>.txt” where the maximum value of <num> is dependent on the number of records in the “input” CSV file. A sample extract of the configuration file “config-1.txt” is shown below with each replaced parameter highlighted in bold red:

```
config-1.txt
hostname Router1
!
```

```
enable secret password
!
clock timezone GMT 0 0
clock summer-time BST recurring last Sun Mar 2:00 last Sun Oct 2:00
!
no ip bootp server
no ip domain lookup
ip domain name network.co.uk
!
!
username fred password bloggs
!
ip ftp source-interface Loopback1
ip tftp source-interface Loopback1
!
interface Loopback1
 ip address 192.168.1.1 255.255.255.255
!
interface GigabitEthernet0/0
 bandwidth 20000
 no ip address
 no ip redirects
 no ip proxy-arp
 duplex auto
 speed auto
 no cdp enable
!
interface GigabitEthernet0/0.100
 encapsulation dot1Q 100
 ip address 10.1.1.1 255.255.255.0
.
.
etc.
```

2 Script Contents

2.1 Generate (simple version)

With this simple version it is necessary to use a text editor to change the number of parameters to match those specified in the CSV and template configuration files. The parameters to change are highlighted in blue.

The “input” CSV file is read and the template file is copied to sub-directory “configdir” creating a target configuration “config-<num>.txt” file for each record in the CSV file.

The “input” CSV file is re-read and the “sed” command used to carry out global substitutions for each parameter in each of the target configuration files.

```
#!/bin/bash
#
# Script to read CSV file "input" as input to generate template
# configurations in subodirectory "configdir"
#

# Check that build data "input" file parameter $1 has been passed
#
if [ -z "$1" ]; then echo "Input file not supplied"; exit; fi

# Check that the number of parameters to replace $2 has been passed
#
if [ -z "$2" ]; then echo "Number of parameters not supplied"; exit; fi

# Change the field separator
```

```
#
IFS=","

inpfile=$1

# Pre-create the output config files based on the number of records
# in the file "input" ready to process with the "sed" command
#
x=1
while read line
do
 outfile="configdir/config-$x.txt"
 cp config.txt $outfile
 x=$[$x+1]
done < $inpfile

# Read the configuration parameters from file "input" and replace
# the parameters in each config file in sub-directory "configdir"
#

x=1
cat $inpfile | while read \
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 \
P11 P12 P13 P14 P15 P16 P17 P18 P19 P20 \
P21 P22 P23 P24 P25

do
 outfile="configdir/config-$x.txt"
 sed -i "s|<<P1>>|$P1|g" "$outfile"
 sed -i "s|<<P2>>|$P2|g" "$outfile"
 sed -i "s|<<P3>>|$P3|g" "$outfile"
 sed -i "s|<<P4>>|$P4|g" "$outfile"
 sed -i "s|<<P5>>|$P5|g" "$outfile"
 sed -i "s|<<P6>>|$P6|g" "$outfile"
 sed -i "s|<<P7>>|$P7|g" "$outfile"
 sed -i "s|<<P8>>|$P8|g" "$outfile"
 sed -i "s|<<P9>>|$P9|g" "$outfile"
 sed -i "s|<<P10>>|$P10|g" "$outfile"
 sed -i "s|<<P11>>|$P11|g" "$outfile"
 sed -i "s|<<P12>>|$P12|g" "$outfile"
 sed -i "s|<<P13>>|$P13|g" "$outfile"
 sed -i "s|<<P14>>|$P14|g" "$outfile"
 sed -i "s|<<P15>>|$P15|g" "$outfile"
 sed -i "s|<<P16>>|$P16|g" "$outfile"
 sed -i "s|<<P17>>|$P17|g" "$outfile"
 sed -i "s|<<P18>>|$P18|g" "$outfile"
 sed -i "s|<<P19>>|$P19|g" "$outfile"
 sed -i "s|<<P20>>|$P20|g" "$outfile"
 sed -i "s|<<P21>>|$P21|g" "$outfile"
 sed -i "s|<<P22>>|$P22|g" "$outfile"
 sed -i "s|<<P23>>|$P23|g" "$outfile"
 sed -i "s|<<P24>>|$P24|g" "$outfile"
 sed -i "s|<<P25>>|$P25|g" "$outfile"
 x=$[$x+1]
done
```

2.2 Generate (using indirection)

This script is similar to the simple version except that it uses variable indirection to substitute each parameter as each target configuration is copied, which is more efficient.

With this version it is necessary to use a text editor to change only the parameters highlighted in blue.

The “input” CSV file is read and the template file copied to sub-directory “configdir” creating a target configuration “config-<num>.txt” file for each record in the CSV file.

A while loop carries out a global substitution of each parameter in the file before the next record of the CSV file is read.

```
#!/bin/bash
#
# Script to read CSV file "input" as input to generate template
# configurations in subodirectory "configdir"
#

# Check that build data "input" file parameter $1 has been passed
#
if [ -z "$1" ]; then echo "Input file not supplied"; exit; fi

# Check that the number of parameters to replace $2 has been passed
#
if [ -z "$2" ]; then echo "Number of parameters not supplied"; exit; fi

# Change the field separator
#
IFS=","

inpfile=$1
numparm=$2

# Read the configuration parameters from file "input" and replace
# the parameters in the config file as each is generated
#

x=1
while read \
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 \
P11 P12 P13 P14 P15 P16 P17 P18 P19 P20 \
P21 P22 P23 P24 P25

do
 outfile="configdir/config-$x.txt"
 cp config.txt $outfile
 x=$[$x+1]
 y=1
 while [ $y -le $numparm ]
 do
  B="<<P$y>>"
  C="P$y"
  eval C=\$$C
  sed -i "s|$B|$C|g" "$outfile"
  y=$[$y+1]
 done
done < $inpfile
```