# **Wireless Auto Test Script User Guide**

[**Wireless Auto Test Script User Guide** 1](#_Toc513102851)

[1. How to get code 2](#_Toc513102852)

[2. Script Architectures 4](#_Toc513102853)

[3. Detail Of Script 5](#_Toc513102854)

[3.1 TestCase.sh 5](#_Toc513102855)

[3.2 Config 8](#_Toc513102856)

[3.3 Scp\_in.exp 10](#_Toc513102857)

[3.4 Scp\_out.exp 11](#_Toc513102858)

[3.5 Single Node 11](#_Toc513102859)

[3.6 Two Nodes 12](#_Toc513102860)

[4. Auto\_Installion 14](#_Toc513102861)

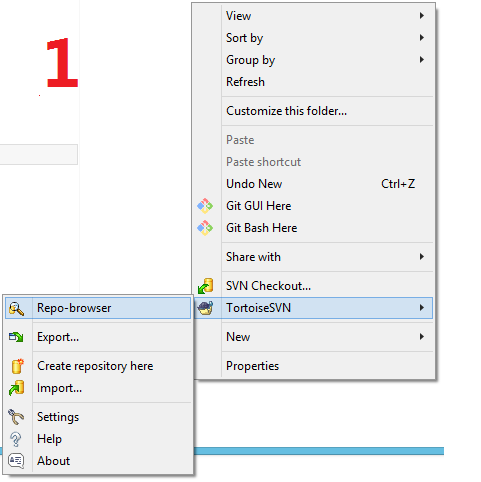
[4.1 Architectures 14](#_Toc513102862)

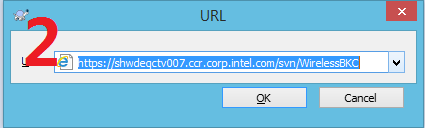
[4.2 run.sh 15](#_Toc513102863)

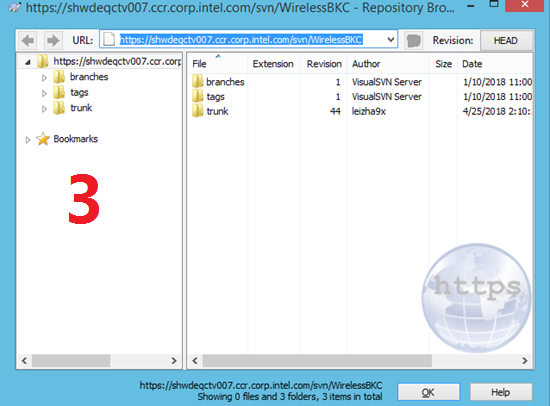
[4.3 Install\_dpdk.sh 16](#_Toc513102864)

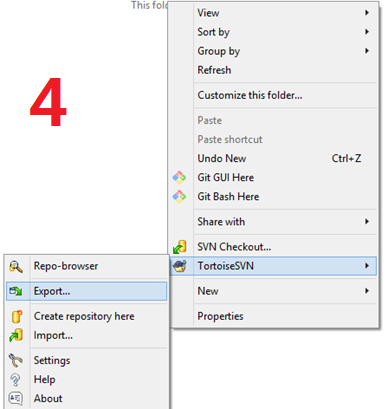
[5. Debug for errors 18](#_Toc513102865)

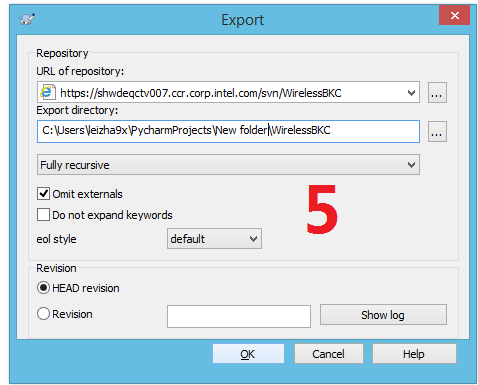
1. **How to get code**
2. Download svn from <https://tortoisesvn.net/downloads.html>
3. Install svn
4. Connect to local network on lab.
5. connect SVN server <https://shwdeqctv007.ccr.corp.intel.com/svn/WirelessBKC> and get code from it.











1. Script Architectures

AutoTest script is under trunk folder

Files in AutoTest folder like below:

|  |
| --- |
| .  ├── case  │   ├── common.sh  │   ├── VLANFilter.sh  │   ├── vNIC\_VHOST\_VLANFilter\_AVS\_FVL10G.sh  │   ├── vProcessor\_CPUInfo.sh  .  .  ├── config  │   ├── common\_config.ini  │   ├── vNIC\_VHOST\_VLANFilter\_AVS\_FVL10G.ini  │   ├── vProcessor\_CPUInfo.ini  .  .  ├── Installation  │   ├── install\_dpdk.sh  │   ├── install\_i40evf.sh  │   ├── install\_icc.exp  │   ├── install\_icc.sh  .  .  ├── check\_boot.sh  ├── scp\_in.exp  ├── scp\_out.exp  ├── testCase.sh  ├── tmp  │   ├── tmp  │   ├── vNIC\_SRIOV\_Traffic\_SingleNode\_JumboFrame\_FVL10G  │   └── vNIC\_VHOST\_IperfStress\_2Node\_UDP\_FVL10G  └── tools  ├── cli.py  └── ser.py |

Case: include case scripts and some common scripts

Config: include case configs and common configs

Installation: merge from auto installion scripts

Tmp: store some tmp file for test case

Tools: some tools used while runing test case

Check\_boot.sh: check boot config

Scp\_in.exp: copy scripts and some packages needed into VMs

Scp\_out.exp: copy logs from VMs

Testcase.sh: start script

1. Detail Of Script
   1. TestCase.sh

The usage of testCase.sh is “testcase.sh -c <config file path>”

testCase.sh is the entry point of the script package

The steps of it:

1. Get case config from arguments

|  |
| --- |
| **while** getopts **"c:"** arg;**do  case $**{arg} **in** c)  config=**$**{OPTARG}  **;;** ?)  echo **"arg not found"** echo **"${usage}"** exit 1  **;;  esac done** |

1. Create VMs on the basis of the case config and check what node can create VMs

|  |
| --- |
| nodes=**$***(nova hypervisor-list |grep enabled |awk '{print $4}')* **for** node **in $**{nodes};**do** tmp=**$***(nova hypervisor-show ${node})* total=*`*echo **"${tmp}"** |*grep* -w vcpus\_node |*awk* -F \| **'{print $3}'** |*jq* **'.["0"]'***`* used=*`*echo **"${tmp}"** |*grep* -w vcpus\_used |*awk* **'{print $4}'** |*awk* -F \. **'{print $1}'***`* can\_vms=**$(($((**$total-$used**))**/4**))** node\_vm\_info=$node\_vm\_info**" $node|$can\_vms" done** image\_id=**$***(glance image-list |egrep -i "\s${image\_name}\s" |awk '{print $2}')  # virtio pci-sriov* cmd=**"nova boot --flavor=${flavor} \ --nic net-name=${SSH\_NET} \ --nic net-name=${P0\_NET},vif-model=${vif\_model} \ --nic net-name=${P1\_NET},vif-model=${vif\_model} \ --image ${image\_id} "** declare -a cmds **if [ "${vms}" == "1" ]**;**then** *cmds[${#cmds[\*]}]="${cmd} ${vm\_name}"* **elif [ "${vms}" == "2" ]**;**then  for** i **in** $node\_vm\_info; **do** nod=*`*echo **"$i"** |*awk* -F\| **'{print $1}'***`* **if [ "$nod" == "WFT-2" ]**;**then** continue  **fi** unused\_vms=*`*echo **"$i"** |*awk* -F\| **'{print $2}'***`* **if [ "${hosts}" == "1" ]**;**then  if [[ $**{unused\_vms} **-ge $**{vms} **]]**;**then  while [[ $**{#cmds[\*]} **-lt $**{vms} **]]**;**do** *cmds[${#cmds[\*]}]="$cmd ${vm\_name}\_${#cmds[\*]} --availability-zone nova:${nod}"* **done  fi  elif [ "${hosts}" == "2" ]**;**then  if [[ $**{unused\_vms} **-ge** 1 **]]**;**then** *cmds[${#cmds[\*]}]="$cmd ${vm\_name}\_${nod} --availability-zone nova:${nod}"* **if [[ $**{#cmds[\*]} **-ge $**{vms} **]]**;**then** break  **fi  fi  fi  done  if [[ $**{#cmds[\*]} **-lt $**{vms} **]]**;**then** echo **"not enough hosts found"** exit 1  **fi fi** |

1. Create VMs and waiting it done and store some temp information like mac and IP

|  |
| --- |
| **for ((**i=0;i<**$**{#cmds[\*]};i++**))**;**do** *# create VMs* cmd=**$**{cmds[i]}  tmp=*`*eval *${cmd}`* vm\_id=*`*echo **"${tmp}"**|*egrep* **"\bid\b"** |*awk* -F \| **'{print $3}'**|*sed* **'s/\s//g'***`  sleep* 2  **while [** 1 **]**;**do** *sleep* 3  vm\_info=*`nova show* $vm\_id*`* status=*`*echo **"$vm\_info"** |*egrep* -w **"status"** |*awk* -F \| **'{print $3}'**|*sed* **'s/\s//g'***`* **if [ "$status" == "ACTIVE" ]**;**then** *logger* **"New creating VM status is $status, waiting UP [DEBUG]"** break  **elif [ "$status" == "ERROR" ] || [ "$status" == "" ]**;**then** *logger* **"New creating VM status is $status [FAIL]"** error\_message=*`*echo **"${vm\_info}"** |*grep* **"| fault"** |*awk* -F \| **'{print $3}'***`  logger* **"${error\_message} [FAIL]"** exit 1  **else** *logger* **"New creating VM status is $status, waiting ACTIVE [DEBUG]"  fi  done** *sleep* 5  SSH\_IP=*`*echo **"$vm\_info"** |*grep* **"${SSH\_NET} network"** |*awk* -F \| **'{print $3}'** |*sed* **'s/\s//g'***`* SSH\_MAC=*`*echo **"$vm\_info"** |*grep* **"\"${SSH\_NET}\""**|*awk* -F \| **'{print $3}'** |*jq* **'.nic1.mac\_address'** |*sed* s/\"//g*`* P0\_IP=*`*echo **"$vm\_info"** |*grep* **"${P0\_NET} network"** |*awk* -F \| **'{print $3}'** |*sed* **'s/\s//g'***`* P0\_MAC=*`*echo **"$vm\_info"** |*grep* **"\"${P0\_NET}\""**|*awk* -F \| **'{print $3}'** |*jq* **'.nic2.mac\_address'** |*sed* s/\"//g*`* P1\_IP=*`*echo **"$vm\_info"** |*grep* **"${P1\_NET} network"** |*awk* -F \| **'{print $3}'** |*sed* **'s/\s//g'***`* P1\_MAC=*`*echo **"$vm\_info"** |*grep* **"\"${P1\_NET}\""**|*awk* -F \| **'{print $3}'** |*jq* **'.nic3.mac\_address'** |*sed* s/\"//g*`* **if [ "${SSH\_NET}" == "" ]**;**then** *logger* **"Create VM [FAIL]"** exit 1  **else** *logger* **"Create VM [PASS]"  fi** LOGFILE=**$**{case}\_**$**{DT}\_**$**{i}.LOG  echo **"COUNT=${i} SSH\_IP=${SSH\_IP} SSH\_MAC=${SSH\_MAC} \  P0\_IP=${P0\_IP} P0\_MAC=${P0\_MAC} \  P1\_IP=${P1\_IP} P1\_MAC=${P1\_MAC} \  LOGFILE=${LOGFILE} "** *>> ${TMPFILE}* **done** |

1. Copy scripts and packages to VMs and execute case script

|  |
| --- |
| **for** file **in $**{files};**do** *logger* **"AutoTest/scp\_in.exp ${transfer\_node} ${netns} ${ip} ${file} [DEBUG]"** *AutoTest/scp\_in.exp* **$**{transfer\_node} **$**{netns} **$**{ip} **$**{file} |*tee* -a **$**{LOGFILE\_PATH}  *sleep* 2 **done** *#copy AutoTest script to VM logger* **"AutoTest/scp\_in.exp ${transfer\_node} ${netns} ${ip} AutoTest ${case\_shell} [DEBUG]"** *AutoTest/scp\_in.exp* **$**{transfer\_node} **$**{netns} **$**{ip} AutoTest **$**{case\_shell} |*tee* -a **$**{LOGFILE\_PATH} |

1. Copy out logs from VMs while test finished

|  |
| --- |
| *AutoTest/scp\_out.exp* **$**{transfer\_node} **$**{netns} **$**{ip} **$**{LOGFILE} |*tee* -a **$**{LOGFILE\_PATH} |

* 1. Config

Define configs for case vms and test scripts

For example vNIC\_AVP\_IperfStress\_SingleNode\_TCP\_FVL10G.ini

|  |
| --- |
| # VM config hosts=1 vms=2 #virtio pci-sriov avp vif\_model=avp flavor=Pktgen  # case case\_shell=./AutoTest/case/vNIC\_AVP\_IperfStress\_SingleNode\_TCP\_FVL10G.sh  # test data type=tcp  # add common config common\_configs="config/common\_config.ini ../config/common\_config.ini" for common\_config in ${common\_configs};do  if [ -f ${common\_config} ];then  . ${common\_config}  break  fi done |

hosts=1 # create vms on single node   
vms=2 # create two VMs

vif\_model=avp # mean 1 ssh network + 2 \* avp  
flavor=Pktgen # will be covered by common\_config.ini

# if you want to set different flavor only for this case , please add flavor=<you want> after last line

case\_shell=./AutoTest/case/vNIC\_AVP\_IperfStress\_SingleNode\_TCP\_FVL10G.sh

#test script will be executed while login in VM

type=tcp # test data will used while running test script

common\_config.ini

|  |
| --- |
| #image image\_name=flexran-new  #vm flavor=pktgen vm\_name=`basename ${case\_shell} .sh` case=${vm\_name}  #log file TMPFILE=tmp/$case vm\_ip=`ifconfig |grep "192.168.0." |awk '{print $2}'` if [ $? == 0 ];then  LOGFILE=`cat ${PWD}/../tmp/${case} |grep "=${vm\_ip}\ " |sed 's/ /\n/g' |grep "LOGFILE=" |awk -F = '{print $2}'`  LOGFILE\_PATH=~/AutoTest/log/${LOGFILE} fi |

image\_name=flexran-new # change this while image changes

flavor=pktgen # change this while flavor changes

vm\_ip=`ifconfig |grep "192.168.0." |awk '{print $2}'`

#when test case needs two nodes, get vm\_ip while running test script in vm, then check this vm is first or second, will see detail in 2 nodes test

LOGFILE=`cat ${PWD}/../tmp/${case} |grep "=${vm\_ip}\ " |sed 's/ /\n/g' |grep "LOGFILE=" |awk -F = '{print $2}'`

LOGFILE\_PATH=~/AutoTest/log/${LOGFILE} # record log file path , used in common.sh

* 1. Scp\_in.exp

Expect script used to login VM

Called in testCase.sh

AutoTest/scp\_in.exp ${transfer\_node} ${netns} ${ip} AutoTest ${case\_shell}

1. Copy autotest packages or others to node

|  |
| --- |
| spawn scp -r $files wrsroot@$host0:~ set timeout 200 expect "\*password:\*" { send "$wrsrootpswd\r" } expect eof |

1. Login node

|  |
| --- |
| spawn ssh $host0 expect "\*password:\*" { send "$wrsrootpswd\r" } |

1. Clear tmp information

|  |
| --- |
| expect ":~" { send "echo $wrsrootpswd | sudo -S sed -i '/$host1/d' /root/.ssh/known\_hosts\r" } sleep 1 send "sudo sed -i '/^\*/d' /etc/ssh/ssh\_known\_hosts\r" sleep 1 |

1. Copy packages to VM

|  |
| --- |
| send "sudo ip netns exec ${netns} scp -r /home/wrsroot/`basename $files` root@$host1:~\r" expect "Password:" {send "$wrsrootpswd\r"} expect "\*(yes/no)" {send "yes\r"} set timeout 200 expect "\*root\*" {send "$rootpswd\r"} |

1. If case\_shell is none, then exit,

else run check\_boot.sh to check boot config is right or not

|  |
| --- |
| send "sudo ip netns exec ${netns} ssh root@$host1\r" expect "Password:" {send "$wrsrootpswd\r"} expect "\*(yes/no)" {send "yes\r"} expect "\*root\*" {send "$rootpswd\r"} sleep 3 send "chmod -R 777 /root/AutoTest\r" send "AutoTest/check\_boot.sh\r" |

1. After reboot, login VM and run case shell

Until screen show “"--check finished--”

|  |
| --- |
| set timeout 100000  send "${case\_shell}\r" expect {  -re "--check finished--" {  send "exit\r"  }  -re "--check continue--" {  send "exit\r"  sleep 30  set timeout 3  send "sudo ip netns exec ${netns} ssh root@$host1\r"  expect "Password:" {send "$wrsrootpswd\r"}  expect "\*(yes/no)" {send "yes\r"}  expect "\*root\*" {send "$rootpswd\r"}  sleep 3  send "${case\_shell}\r"  set timeout 100000  exp\_continue  } } |

* 1. Scp\_out.exp

1. Ssh node
2. Copy logfile from VM to node
3. Copy logfile from node to control
   1. Single Node

Example:　DriverReset.sh

1. up the eth1 and eth2
2. Check link status

|  |
| --- |
| **for** eth **in** eth1 eth2;**do** *ifconfig* **$**{eth} up  *sleep* 2  *logger* **"`ifconfig ${eth} |grep ${eth}` [INFO]"** eth\_info=*`ifconfig* **$**{eth} |*grep* **$**{eth} |*awk* -F \< **'{print $2}'** |*awk* -F \> **'{print $1}'***`* **if [ "${eth\_info}" == "UP,BROADCAST,RUNNING,MULTICAST" ]**;**then** *logger* **"${eth} up status check [PASS]"  else** *logger* **"${eth} up status check [FAIL]"** ERROR=**$((**$ERROR+1**))  fi** *logger* **"`ethtool ${eth}` [INFO]"** link=*`ethtool* **$**{eth} |*grep* **"Link detected:"** |*awk* **'{print $3}'***`* **if [ "${link}" == "yes" ]**;**then** *logger* **"${eth} up Link Detected ${link} [PASS]"  else** *logger* **"${eth} up Link Detected ${link} [FAIL]"** ERROR=**$((**$ERROR+1**))  fi   if [ "${IS\_SRIOV}" == "1" ]**;**then** speed=*`ethtool* **$**{eth} |*grep* **"Speed"** |*awk* **'{print $2}'***`* **if [ "${speed}" == "40000Mb/s" ]**;**then** *logger* **"${eth} speed check [PASS]"  else** *logger* **"${eth} speed check [FAIL]"** ERROR=**$((**$ERROR+1**))  fi  fi done** |

1. Down the eth1 and eth2
2. Check link status

|  |
| --- |
| **for** eth **in** eth1 eth2;**do** *ifconfig* **$**{eth} down  *sleep* 2  *logger* **"`ifconfig ${eth} |grep ${eth}` [INFO]"** eth\_info=*`ifconfig* **$**{eth} |*grep* **$**{eth} |*awk* -F \< **'{print $2}'** |*awk* -F \> **'{print $1}'***`* **if [ "${eth\_info}" == "BROADCAST,MULTICAST" ]**;**then** *logger* **"${eth} down status check [PASS]"  else** *logger* **"${eth} down status check [FAIL]"** ERROR=**$((**$ERROR+1**))  fi** *logger* **"`ethtool ${eth}` [INFO]"** link=*`ethtool* **$**{eth} |*grep* **"Link detected:"** |*awk* **'{print $3}'***`* **if [ "${link}" == "no" ]**;**then** *logger* **"${eth} down Link Detected ${link} [PASS]"  else** *logger* **"${eth} down Link Detected ${link} [FAIL]"** ERROR=**$((**$ERROR+1**))  fi done** |

1. Check test status

|  |
| --- |
| **if [ "${ERROR}" ==** 0 **]**;**then** *logger* **"${case} --check finished-- [PASS]" else** *logger* **"${case} --check finished-- [FAIL]" fi** |

* 1. Two Nodes

Expamle: VLANFilter.sh

1. Get VM ip and check which one it is

|  |
| --- |
| vm\_ip=*`ifconfig* |*grep* **"192.168.0."** |*awk* **'{print $2}'***`* vm\_info=*`cat* ../tmp/**$**{case} |*grep* **"SSH\_IP=$vm\_ip"** |*sed* **'s/ /\n/g'***`* count=*`*echo **"${vm\_info}"** |*grep* COUNT= |*awk* -F \= **'{print $2}'***`* |

1. If this VM is first one, run python ser.py background

|  |
| --- |
| **if [ "$count" ==** 0 **]**;**then** *python* ../tools/ser.py &  *logger* **"${case} --check finished-- [DEBUG]"** |

1. If this VM is second one, run test case and check pass or not

|  |
| --- |
| **elif [ "$count" ==** 1 **]**;**then** other\_info=*`cat* ../tmp/**$**{case} |*grep* -v **"SSH\_IP=$vm\_ip"** |*sed* **'s/ /\n/g'***`* other\_eth1\_ip=*`*echo **"$other\_info"** |*grep* **'P0\_IP='** |*awk* -F \= **'{print $2}'***`  ping* -I eth1 $other\_eth1\_ip -c 4  **if [** $? **==** 0 **]**;**then** *logger* **"ping eth1 $other\_eth1\_ip [PASS]"  else** *logger* **"ping eth1 $other\_eth1\_ip [FAIL]"** ERROR=**$((**ERROR+1**))  fi** other\_eth2\_ip=*`*echo **"$other\_info"** |*grep* **'P1\_IP='** |*awk* -F \= **'{print $2}'***`  ping* -I eth2 $other\_eth2\_ip -c 4  **if [** $? **==** 0 **]**;**then** *logger* **"ping eth2 $other\_eth2\_ip [PASS]"  else** *logger* **"ping eth2 $other\_eth2\_ip [FAIL]"** ERROR=**$((**ERROR+1**))  fi** other\_ssh\_ip=*`*echo **"$other\_info"** |*grep* **'SSH\_IP='** |*awk* -F \= **'{print $2}'***`* cmd=**"ifconfig eth1 |grep inet6 |awk '{print \$2}'"** other\_eth1\_ipv6\_address=*`python ../tools/cli.py -i* **$**{other\_ssh\_ip} *-c* **"${cmd}"***`  ping6* -I eth1 $other\_eth1\_ipv6\_address -c 4  **if [** $? **==** 0 **]**;**then** *logger* **"ping6 eth1 $other\_eth1\_ipv6\_address [PASS]"  else** *logger* **"ping6 eth1 $other\_eth1\_ipv6\_address [FAIL]"** ERROR=**$((**ERROR+1**))  fi** cmd=**"ifconfig eth2 |grep inet6 |awk '{print \$2}'"** other\_eth2\_ipv6\_address=*`python ../tools/cli.py -i* **$**{other\_ssh\_ip} *-c* **"${cmd}"***`  ping6* -I eth2 $other\_eth2\_ipv6\_address -c 4  **if [** $? **==** 0 **]**;**then** *logger* **"ping6 eth2 $other\_eth2\_ipv6\_address [PASS]"  else** *logger* **"ping6 eth2 $other\_eth2\_ipv6\_address [FAIL]"** ERROR=**$((**ERROR+1**))  fi   if [ "${ERROR}" ==** 0 **]**;**then** *logger* **"${case} --check finished-- [PASS]"  else** *logger* **"${case} --check finished-- [FAIL]"  fi fi** |

other\_ssh\_ip=*`*echo **"$other\_info"** |*grep* **'SSH\_IP='** |*awk* -F \= **'{print $2}'***`* cmd=**"ifconfig eth1 |grep inet6 |awk '{print \$2}'"** other\_eth1\_ipv6\_address=*`python ../tools/cli.py -i* **$**{other\_ssh\_ip} *-c* **"${cmd}"***`*

This 3 lines connect to first VM and get its ipv6 ip

1. Auto\_Installion
   1. Architectures

Installion folder is merged from Auto\_Installion, but has any different between them.

|  |
| --- |
| .  ├── add\_patch.sh  ├── bind\_port.sh  ├── common\_fun.sh  ├── config\_env.sh  ├── config.sh  ├── extlinux.conf  ├── extra  │   └── start.sh  ├── flexran  │   ├── auto\_extract.sh  │   ├── create\_re\_bin.sh  │   └── r\_buildall.sh  ├── install\_dpdk.sh  ├── install\_flexran.sh  ├── install\_gtest.sh  ├── install\_icc.exp  ├── install\_icc.sh  ├── install\_license.sh  ├── install\_pktgen.sh  ├── install\_qat.exp  ├── install\_qat.sh  ├── install\_rpm.sh  ├── patch  │   ├── dpdk-dev-usertools-add-support-for-AVP-device.patch  │   ├── l2fwd  │   │   └── main.c  │   ├── l2fwd-crypto  │   │   └── main.c  │   └── l3fwd  │   └── main.c  ├── pktgen.exp  ├── README.md  ├── rpm  │   ├── dos2unix-7.3.4-5.3.1.x86\_64.rpm  │   ├── expect-5.45-14.el7\_1.x86\_64.rpm  │   └── tcl-8.5.13-8.el7.x86\_64.rpm  └── run.sh |

* 1. run.sh

**1 Install SUT (One key to install all drivers, utilities and BKC testing tools with required patches, step1.1-1.4)  
 1.1 Install RPM dependency Only  
 1.2 Install ICC with licenses Only  
 1.3 Install DPDK Only  
 1.4 System configuration to update extlinux.conf  
2 Install PKTGEN client  
3 Bind Ethernet to DPDK for BKC testing  
4 Install Flexran  
0. Exit  
Input you choice:"**

|  |
| --- |
| **while [** 1 **]**; **do** export ADD\_PATCH=1  read -p **"$choose\_info"** choose  *stop\_trap\_signal* **case "$choose" in  "0"**)  break  **;;  "1"**)  cd **$**{SCRIPT\_FOLDER}  . ./install\_rpm.sh  cd **$**{SCRIPT\_FOLDER}  . ./install\_icc.sh  cd **$**{SCRIPT\_FOLDER}  . ./install\_dpdk.sh *# cd ${SCRIPT\_FOLDER} # . ./install\_qat.sh* cd **$**{SCRIPT\_FOLDER}  . ./config\_env.sh  **;;  "1.1"**)  cd **$**{SCRIPT\_FOLDER}  . ./install\_rpm.sh  **;;  "1.2"**)  cd **$**{SCRIPT\_FOLDER}  . ./install\_icc.sh  **;;  "1.3"**)  cd **$**{SCRIPT\_FOLDER}  . ./install\_dpdk.sh  **;;  "1.4"**)  cd **$**{SCRIPT\_FOLDER}  . ./config\_env.sh  **;;  "2"**)  cd **$**{SCRIPT\_FOLDER}  . ./install\_pktgen.sh  cd **$**{SCRIPT\_FOLDER}  . ./config\_env.sh  **;;  "4"**)  cd **$**{SCRIPT\_FOLDER}  . ./install\_flexran.sh  **;;  "3"**)  cd **$**{SCRIPT\_FOLDER}  . ./bind\_port.sh  **;;  "\*"**)  echo **"Your choose is not match, please try again."  esac** *start\_trap\_signal* **done** |

Get your choose and run corresponding script

* 1. Install\_dpdk.sh

1. Extract dpdk package and set some value for installion

|  |
| --- |
| **[** ! **-d $**{INSTALL\_FOLDER} **] &&** mkdir **$**{INSTALL\_FOLDER} cd **$**{DPDK\_FOLDER} dpdk\_pkg=`ls -F **$**{INSTALL\_FOLDER} |grep dpdk.\*/**$**` dpdk\_tar=`ls -F |grep dpdk.\*[^/]**$**` **[ "$dpdk\_pkg" == "" ] &&** tar -xvf $dpdk\_tar -C **$**{INSTALL\_FOLDER} dpdk\_pkg=`ls -F **$**{INSTALL\_FOLDER} |grep **"dpdk.\*/$"**` export RTE\_SDK=**$**{INSTALL\_FOLDER}/**$**{dpdk\_pkg} *add\_bashrc* **"export RTE\_SDK=${INSTALL\_FOLDER}/${dpdk\_pkg}"** export RTE\_TARGET=x86\_64-native-linuxapp-icc *add\_bashrc* **"export RTE\_TARGET=x86\_64-native-linuxapp-icc"** |

2. Add patch and replace any main.c, add\_patch.sh

|  |
| --- |
| **if [ $**{ADD\_PATCH} **==** 1 **]**;**then** . **$**{SCRIPT\_FOLDER}/add\_patch.sh **fi** |

|  |
| --- |
| cp **$**{SCRIPT\_FOLDER}/patch/\*.patch . **for** patch **in** `ls \*.patch`;**do** patch -p 1 *< $patch* **done** *#cp ${SCRIPT\_FOLDER}/patch/l2fwd/main.c examples/l2fwd/* cp **$**{SCRIPT\_FOLDER}/patch/l2fwd-crypto/main.c examples/l2fwd-crypto/ *# check whether network is virtio* lspci |grep **"00:04.0 Eth"** *> /dev/null* **if [** $? **==** 0 **]**;**then** echo **"network is virtio will fix l3fwd main.c"** cp **$**{SCRIPT\_FOLDER}/patch/l3fwd/main.c examples/l3fwd/ **fi** |

3. install dpdk, l3fwd, l2fwd-crypto

|  |
| --- |
| make install T=x86\_64-native-linuxapp-icc **if [ -f** x86\_64-native-linuxapp-icc/app/testpmd **]**;**then** echo **"DPDK is installed successfully." else** echo **"DPDK is installed failed."** exit 1 **fi**  modprobe uio lsmod |grep igb\_uio *>> /dev/null* **||** insmod x86\_64-native-linuxapp-icc/kmod/igb\_uio.ko  cd **$**{INSTALL\_FOLDER}/**$**{dpdk\_pkg}examples/l3fwd **if [ -f** build/l3fwd **]**;**then** echo **"l3fwd has already installed." else** make  **if [ -f** build/l3fwd **]**;**then** echo **"l3fwd is installed successfully."  else** echo **"l3fwd is installed failed."** exit 1  **fi fi**  cd **$**{INSTALL\_FOLDER}/**$**{dpdk\_pkg}examples/l2fwd-crypto **if [ -f** build/l2fwd-crypto **]**;**then** echo **"l2fwd-crypto has already installed." else** make  **if [ -f** build/l2fwd-crypto **]**;**then** echo **"l2fwd-crypto is installed successfully."  else** echo **"l2fwd-crypto is installed failed."** exit 1  **fi fi** |

Other script is similar with this, detail information need to see the script.

Single step debug is supported for Auto\_Installion, you can run ./install\_dpdk.sh for install dpdk.

1. Debug for errors

If you get some error while installing, you can use bash –x <script> to see detail information.

1. Creating VM error

Check config/common\_config.sh and config/<case>.ini match with system or not. Detail see 3.2

1. Login in vm error

It may login time out after copy a large file, it always happened while login “WFT-2”, need to move DHAP agent from WFT-2 to others.

It may login time out after reboot vm, sometimes vm reboot speends a long time. Default set 30 seconds, detail see scp\_in.exp

1. Case test fail

Check it is case fail or bug for this test. Sometimes update any packages will cause fail, need to fix test case script.

1. Copy log out fail

Check log create and created correctly. If not, check tmp file created correctly after VM creating.