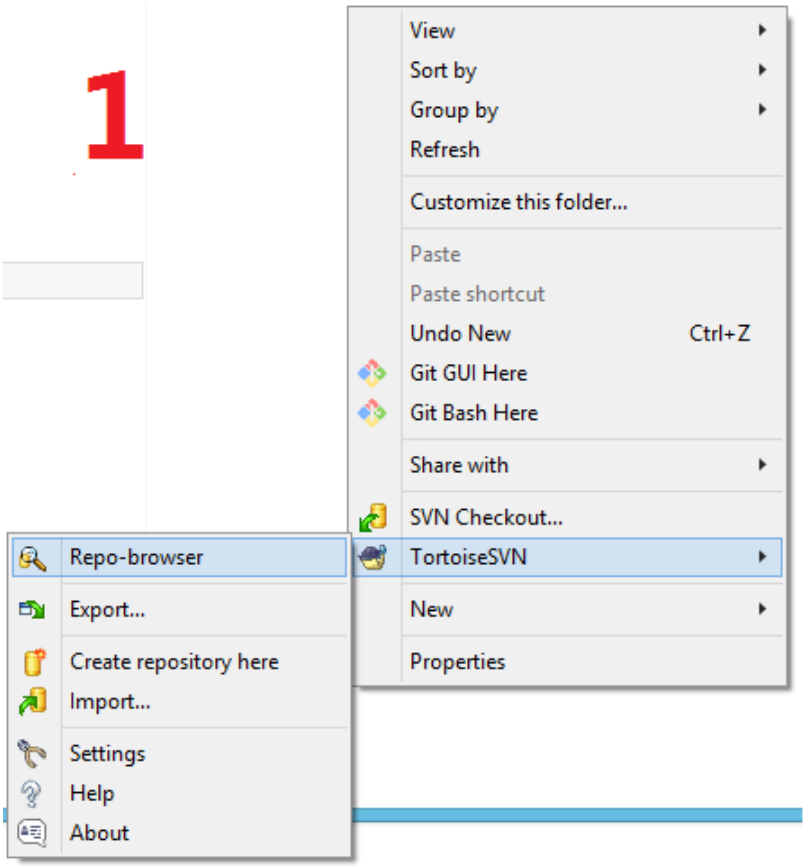
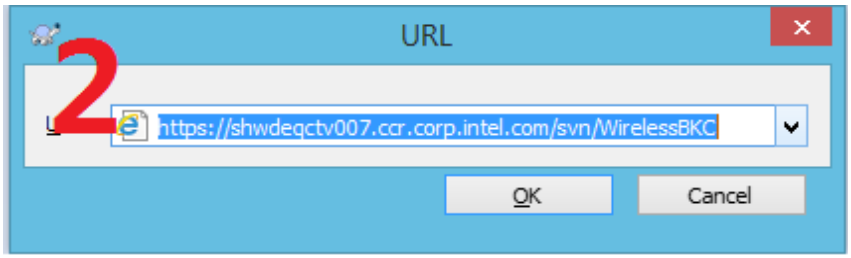
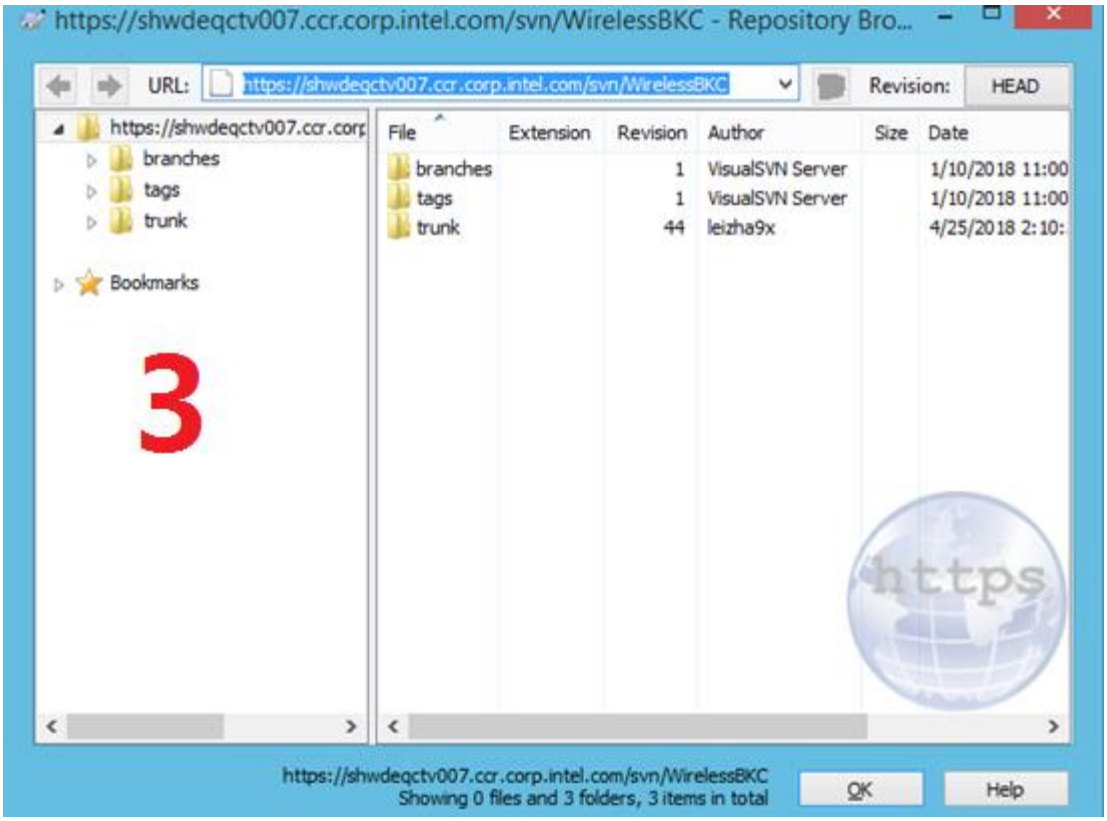
Wireless Auto Test Script User Guide

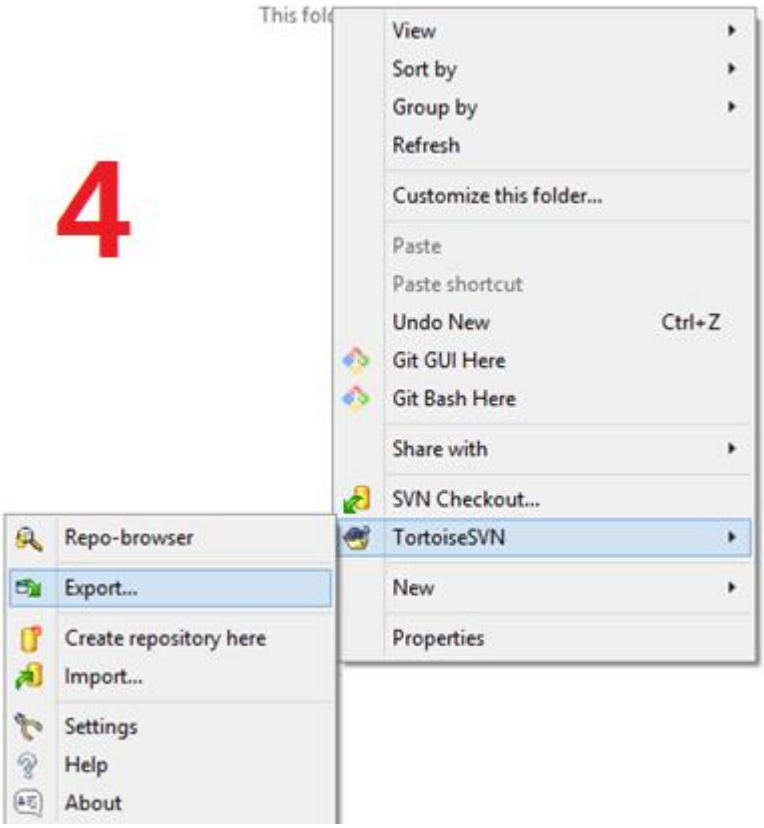
1. How to get code

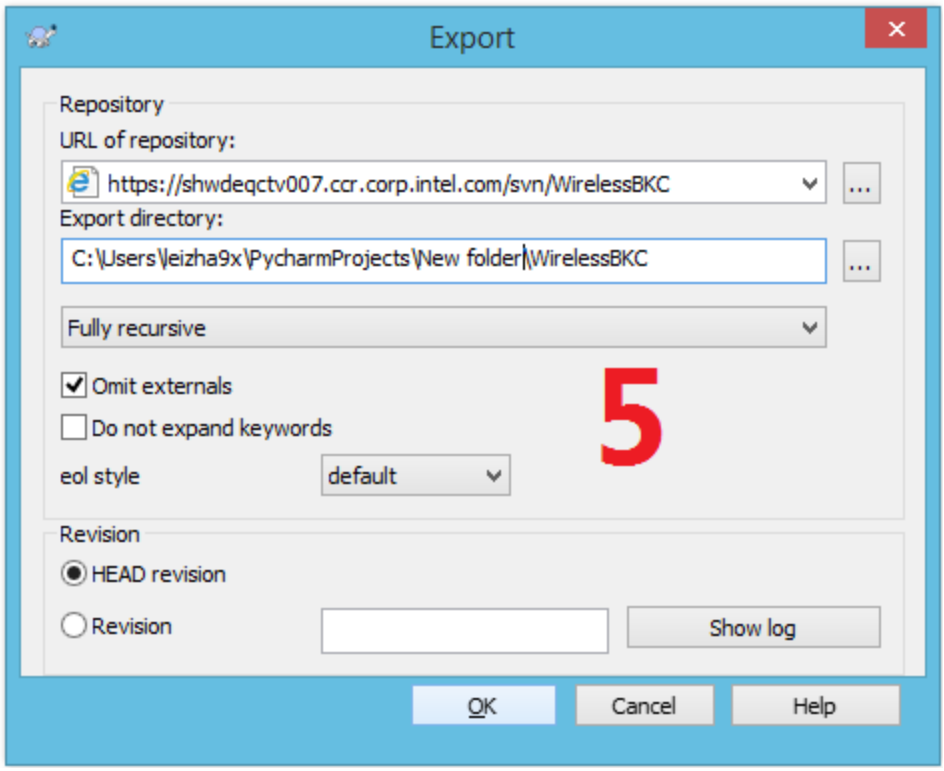
1. Download svn from https://tortoisesvn.net/downloads.html
2. Install svn
3. Connect to local network on lab.
4. connect SVN server https://shwdeqctv007.ccr.corp.intel.com/svn/WirelessBKC and get code from it.






https://shwdeqctv007.ccr.corp.intel.com/svn/WirelessBKC - Repository Bro...
URL:
https://shwdeqctv007.ccr.corp.intel.com/svn/WirelessBKC
Revision:
HEAD
https://shwdeqctv007.ccr.corp
branches
tags
trunk
Bookmarks
3
File
Extension
Revision
Author
Size
Date
branches 1 VisualSVN Server 1/10/2018 11:00
tags 1 VisualSVN Server 1/10/2018 11:00
trunk 44 leizha9x 4/25/2018 2:10:
https
https://shwdeqctv007.ccr.corp.intel.com/svn/WirelessBKC
Showing 0 files and 3 folders, 3 items in total
OK
Help


4
View
Sort by
Group by
Refresh
Customize this folder...
Paste
Paste shortcut
Undo New Ctrl+Z
Git GUI Here
Git Bash Here
Share with
SVN Checkout...
TortoiseSVN
New
Properties
Repo-browser
Export...
Create repository here
Import...
Settings
Help
About



2. 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

3. Detail Of Script

3.1 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
```

2. 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
```

3. 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
```

4. 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}
```

5. Copy out logs from VMs while test finished

```
AutoTest/scp_out.exp ${transfer_node} ${netns} ${ip} ${LOGFILE}
|tee -a ${LOGFILE_PATH}
```

3.2 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_FVL
10G.sh

# test data
```

```
type=tcp

# add common config
common_configs="config/common_config.ini ../config/common_config.i
ni"
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

3.3 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
```

2. Login node

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

3. 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
```

4. 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"}
```

5. 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"
```

6. 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
            }
}
```

3.4 Scp_out.exp

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

3.5 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
```

3. Down the eth1 and eth2
4. 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
```

5. Check test status

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

3.6 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}'`
```

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

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

3. 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

4. Auto_Installion

4.1 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
```

4.2 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

4.3 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.

5. 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
2. 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
3. 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.
4. Copy log out fail
   Check log create and created correctly. If not, check tmp file created correctly after VM creating.