#!/bin/ksh

##------------------------------------------------------------------------------

# TITLE :change\_connected\_nodes

# PURPOSE :change connected node using SQL when this changement

# can not be accepted with IDBE(integrated database editor)

# NOTE: Population must be done after this changement

#

# GROUP :IS

# AUTHOR :Alaa Nagy and Eman Aly ( SCC S/W group )

# DATE :17 Aug 2009

#UPDATE HISTORY : >>>> 12 OCT 2009 ( Alaa Nagy )

# create the new node if it isn't in the database

# : >>>> 14 OCT 2010 ( Alaa Nagy )

# check the new\_node is not already connected to

# the desired device

# : >>>> 17 Sep 2011 ( SCC S/W group )

# update the execution of the script as follows:

# > change\_connected\_nodes -h

# > change\_connected\_nodes <device-name>

# > change\_connected\_nodes -p <device-name> <old-node> <new-node>

# > change\_connected\_nodes -a <device-name> <new-node1> <new-node2>

# : >>>> 18 Dec 2011 ( Eman Aly )

# the script allow to you to chose your DataBase server

#------------------------------------------------------------------------------

# @(#) $Header$

# $Log$

#------------------------------------------------------------------------------

#################################################

# FUNCTION AREA

################################################

. /aedc/etc/work/aedc/SCC/aedc\_SCC\_functions

function usage {

echo "

Usage: `tput bold` aedc\_change\_connected\_nodes `tput sgr0`

for continus execution

`tput bold` aedc\_change\_connected\_nodes -h `tput sgr0`

for display help

`tput bold` aedc\_change\_connected\_nodes <device-name> `tput sgr0`

to execute the script once for a device

`tput bold` aedc\_change\_connected\_nodes -p <DB\_server\_name> <device-name> <old-node> <new-node> `tput sgr0`

to change one device's node by a new one

`tput bold` aedc\_change\_connected\_nodes -a <DB\_server\_name> <device-name> <new-node1> <new-node2> `tput sgr0`

to apply 2 nodes to the device

Note: DB\_server\_name can be either `tput bold`AEDC\_SYSTEM`tput sgr0` or `tput bold`AEDC\_TRAINER`tput sgr0`

"

}

function read\_dev {

echo " "

read dev\_nm1?" Enter Group\_name.device\_name <CR to exit> >>> "

if [[ -z ${dev\_nm1} ]]

then

exit

else

echo ${dev\_nm1} |sed 's/D://;s/P://'|read dev\_nm

fi

}

function dsply {

echo "

if exists (select C0202\_dev\_id from T0202\_devices

where (rtrim(C0003\_group\_name)+'.'+rtrim(C0202\_dev\_name))='${dev\_nm}' )

begin

print ' '

print 'The nodes connected to the device ${dev\_nm} and there sides are'

print ' '

set nocount on

select C0201\_node\_name , C0203\_side from T0201\_node , T0203\_terminal

where T0201\_node.C0201\_node\_id =T0203\_terminal.C0201\_node\_id\_A

and T0203\_terminal.C0202\_dev\_id = (

select C0202\_dev\_id from T0202\_devices

where (rtrim(C0003\_group\_name)+'.'+rtrim(C0202\_dev\_name))='${dev\_nm}' )

end

else

begin

print ' '

print ' >>>> THERE IS NO SUCH DEVICE <<<<'

end

go

" | isql -Udbu -Pdbudbu -S$sev

}

function chk\_new\_nod\_exist {

case ${new\_node} in

${node1}|${node2})

oldnod=1

;;

esac

tstcount=`echo "

if not exists (select \* from T0201\_node where C0201\_node\_name='${new\_node}')

print 'not exists'

go

" | isql -Udbu -Pdbudbu -S$sev |wc -l`

}

function creat\_new\_node {

MaxId=`echo "

select C0201\_node\_id from T0201\_node

order by C0201\_node\_id

go

" | isql -Udbu -Pdbudbu -S$sev |tail -3|head -1`

NewId=`expr ${MaxId} + 1`

echo "

set nocount on

insert into T0201\_node

values ( ${NewId} ,'${new\_node}' )

go

" | isql -Udbu -Pdbudbu -S$sev

}

function chng\_conct\_node {

echo "

set nocount on

update T0203\_terminal set

C0201\_node\_id\_A =(select C0201\_node\_id from T0201\_node where C0201\_node\_name='${new\_node}' ) ,

C0201\_node\_id\_B =(select C0201\_node\_id from T0201\_node where C0201\_node\_name='${new\_node}' ) ,

C0201\_node\_id\_C =(select C0201\_node\_id from T0201\_node where C0201\_node\_name='${new\_node}' )

where C0202\_dev\_id = (select C0202\_dev\_id from T0202\_devices

where (rtrim(C0003\_group\_name)+'.'+rtrim(C0202\_dev\_name))='${dev\_nm}' )

and C0201\_node\_id\_A=(

select C0201\_node\_id from T0201\_node where C0201\_node\_name='${node}')

go

use dbcontrol

go

update T0002\_dbstatus set C0002\_by\_pass\_propagation=1

go" | isql -Udbu -Pdbudbu -S$sev

$SCCROOT/aedc\_propgation\_check\_view

echo "`date +"%Y/%m/%d:%H:%M:%S"` Change connected node of ${dev\_nm} from ${node} to ${new\_node}" >> $SCCERR/dbedit\_vs\_pop.log

}

function read\_node {

#==========================================================

# Entering The Node To Be Change and The New Node

#==========================================================

echo ""

read node?"Enter node to be change <CR to exit> >> "

if [[ -z ${node} ]]

then

exit

fi

while true

do

read new\_node?"Enter the new\_node name <CR to abort> >> "

if [[ -z ${new\_node} ]]

then

break

fi

echo " "

#==========================================================

# Checking The Existance Of New Node

#==========================================================

oldnod=0

chk\_new\_nod\_exist

if [ $oldnod -eq 1 ]

then

echo "This node is already connected to the device ${dev\_nm}"

else

if [ $tstcount -eq 2 ]

then

echo "the node ${new\_node} is not exists in the database"

#==========================================================

# Create it ?

#==========================================================

read chang?" do you want to create a new node name ${new\_node}? (y/n) >>> "

if [ ${chang} = y ]

then

echo " "

echo " SQL IS GOING TO CREATE THIS NEW NODE "

creat\_new\_node

else

break

fi

fi

#==========================================================

# Update the connected node

#==========================================================

chng\_conct\_node

echo '

the new nodes connected to the device ${dev\_nm} and there sides became

'

dsply

echo '

NOTE: Population must be done after this changement'

break

fi

done

}

function sequance {

node=$1

new\_node=$2

oldnod=0

chk\_new\_nod\_exist

if [ $oldnod -eq 1 ]

then

echo "This node is already connected to the device ${dev\_nm}"

exit 2

else

if [ $tstcount -eq 2 ]

then

creat\_new\_node

fi

chng\_conct\_node

fi

}

function main {

#===========================================================

# Display Nodes Of The Device Name

#===========================================================

# clear

#select\_db\_serv

dsply > $SCCTMP/temp\_rep

# cat $SCCTMP/temp\_rep

lines=`wc -l $SCCTMP/temp\_rep|awk '{print $1}'`

if [[ ${lines}>2 ]]

then

node1=`tail -3 $SCCTMP/temp\_rep|head -1|awk '{print $1}'`

node2=`tail -2 $SCCTMP/temp\_rep|head -1|awk '{print $1}'`

if [[ -z $1 ]]

then

cat $SCCTMP/temp\_rep

read\_node

fi

case $1 in

-p)

sequance $2 $3

;;

-a)

if [ $2 = $3 ]

then

echo " Device cannot have 2 similar connected nodes"

exit 2

else

sequance $node1 $2

sequance $node2 $3

fi

;;

\*)

#usage

echo "Type :

$0 -h

to get help"

;;

esac

else

cat $SCCTMP/temp\_rep

fi

rm -f $SCCTMP/temp\_rep

}

#####################################

# START OF THE SCRIPT

#####################################

case $# in

0)

while true

do

select\_db\_serv

read\_dev

main

read contin?" Hit <CR> to continue!"

done

;;

1)

if [ $1 = '-h' ]

then

usage

echo "`tput bold` exit 2 `tput sgr0`:

for error of having 2 similar connected nodes ( at -p,-a options )

"

else

dev\_nm=$1

select\_db\_serv

main

fi

;;

5)

sev=$2

dev\_nm=$3

main $1 $4 $5

;;

\*)

echo "

Error: MISSING PARAMETERS"

usage

;;

esac