I

Desktop Application for simulation of Host functionality for Platform for communication Adapter (PCA)

DOCUMENTATION

@ Ronanki infotech private limited @

CONTENTS

1. INTRODUCTION

1.1 Purpose

The purpose of this document is to outline the software design of the ***Desktop Application for simulation of Host functionality for Platform for communication Adapter (PCA)***. This application can able to generate all messages that host/remote system can generate to communicate to PCA device. This communication shall be over TCP socket. Design and implementation of the software shall be easily portable to other operating systems.

1.2 Scope

This document aims to be a guide for developers to implement the desktop application for simulation of host functionality for PCA. The software will consist of three major components: the application that creates connection between the source PCA and the server, the application can able to sending messages to selected destination PCAs in turn receiving ACK messages from those destinations and disconnecting with the server.

1.3 Document Usage

Although this document will go into some detail about user interface specifications and implementation details, developers are encouraged to use their best judgment when making implementation decisions. As such, the UML diagrams that are contained in this document do not attempt to specify the low level details of each class, i.e. private attributes, user interface events, etc. Instead the document intends to specify enough detail for the classes that a developer can implement it. The authors feel that this should be appropriate for this document.

2. REQUIREMENT SPECIFICATION

2.1 Introduction

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These pre-requisites are known as system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended.

2.2 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accomplished by a hardware compatibility list (HCL), especially in case of operating systems.

Hardware Requirements:

1. VDU: CRT / LCD Monitor
2. Input Devices: Keyboard and Mouse
3. RAM: 512 MB or above
4. Processor: core 2 Duo or above

2.3 Software Requirements

Software requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application.

Software Requirements:

1. Operating system : Windows or Linux
2. Run-time: OS compatible JVM (optional)

3. SYSTEM DESIGN

3.1 Introduction to UML

The Unified Modeling Language is a standardized general-purpose modeling language and nowadays is managed as a de facto industry standard by the Object Management Group (OMG). UML includes a set of graphic notation techniques to create visual models of software-intensive systems.

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as activities, actors, business processes, database schemas, components, programming language statements, and reusable software components.

UML combines techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies.

3.2 Models and Diagrams

It is important to distinguish between the UML model and the set of diagrams of a system. A diagram is a partial graphic representation of a system's model. The model also contains documentation that drive the model elements and diagrams.

UML diagrams represent two different views of a system model:

- Static (or structural) view: emphasizes the static structure of the system using objects, attributes, operations and relationships. The structural view includes class diagrams, object diagrams, component diagrams and deployment diagrams.
- Dynamic (or behavioral) view: emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, collaboration diagrams, use case diagrams, activity diagrams and state machine diagrams.

Different structure diagrams are:

- Class Diagram: represents system class, attributes and relationships among the classes.
- Component Diagram: represents how components are split in a software system and dependencies among the components.
- Deployment Diagram: describes the hardware used in system implementations.
- Object Diagram: represents a complete or partial view of the structure of a modeled system.

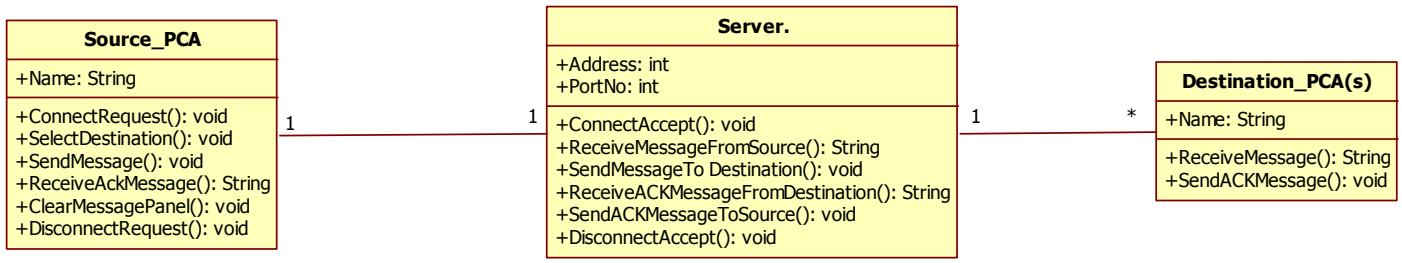
The different behavior diagrams are:

- Activity Diagram: represents step by step workflow of business and operational components.
- Use Case Diagram: describes functionality of a system in terms of actors, goals as use cases and dependencies among the use cases.
- State Machine Diagram: represents states and state transition.
- Collaboration Diagrams: shows the objects and their association with other objects in the system apart from how they interact with each other.
- Sequence Diagram: represents communication between objects in terms of a sequence of messages. Sequence diagram emphasizes time ordering of messages.

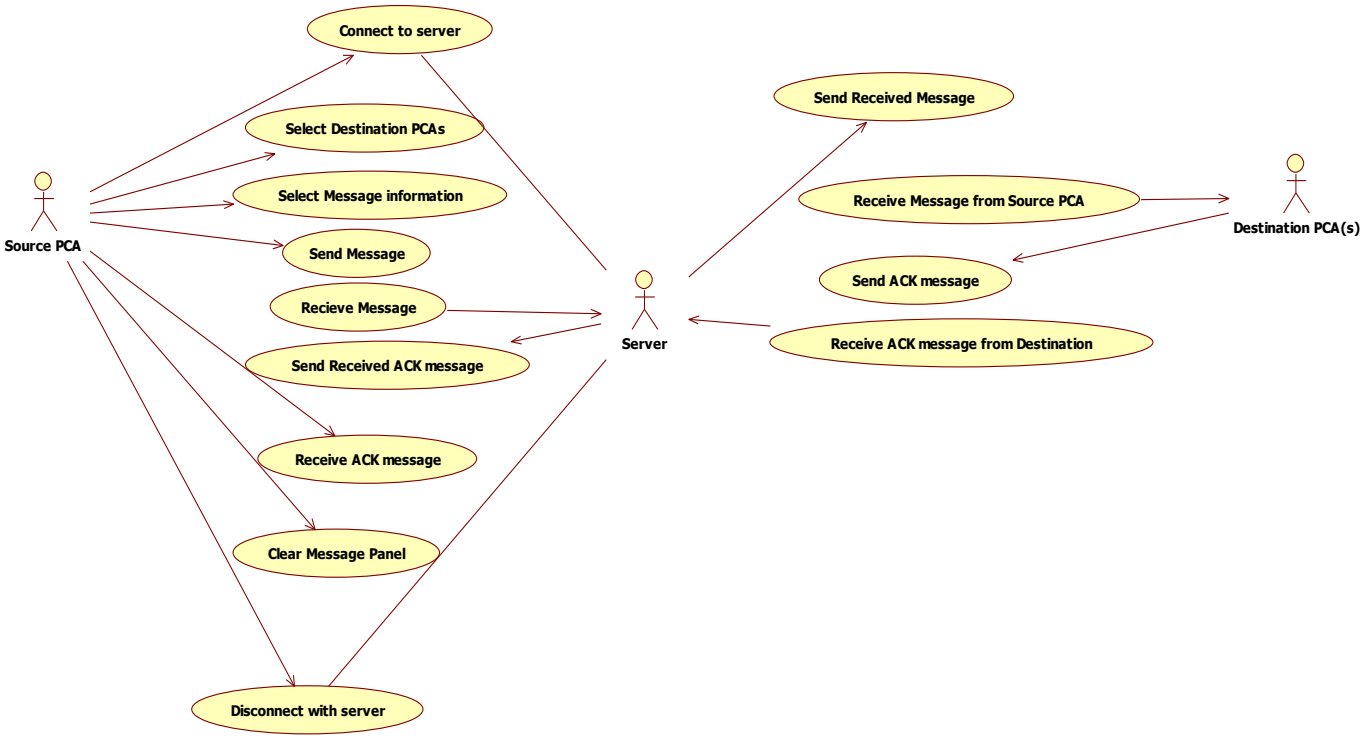
3.3 UML Diagrams of application

For this application I mentioned only class diagram which shows static behavior and usecase and sequence diagrams which shows dynamic behavior.

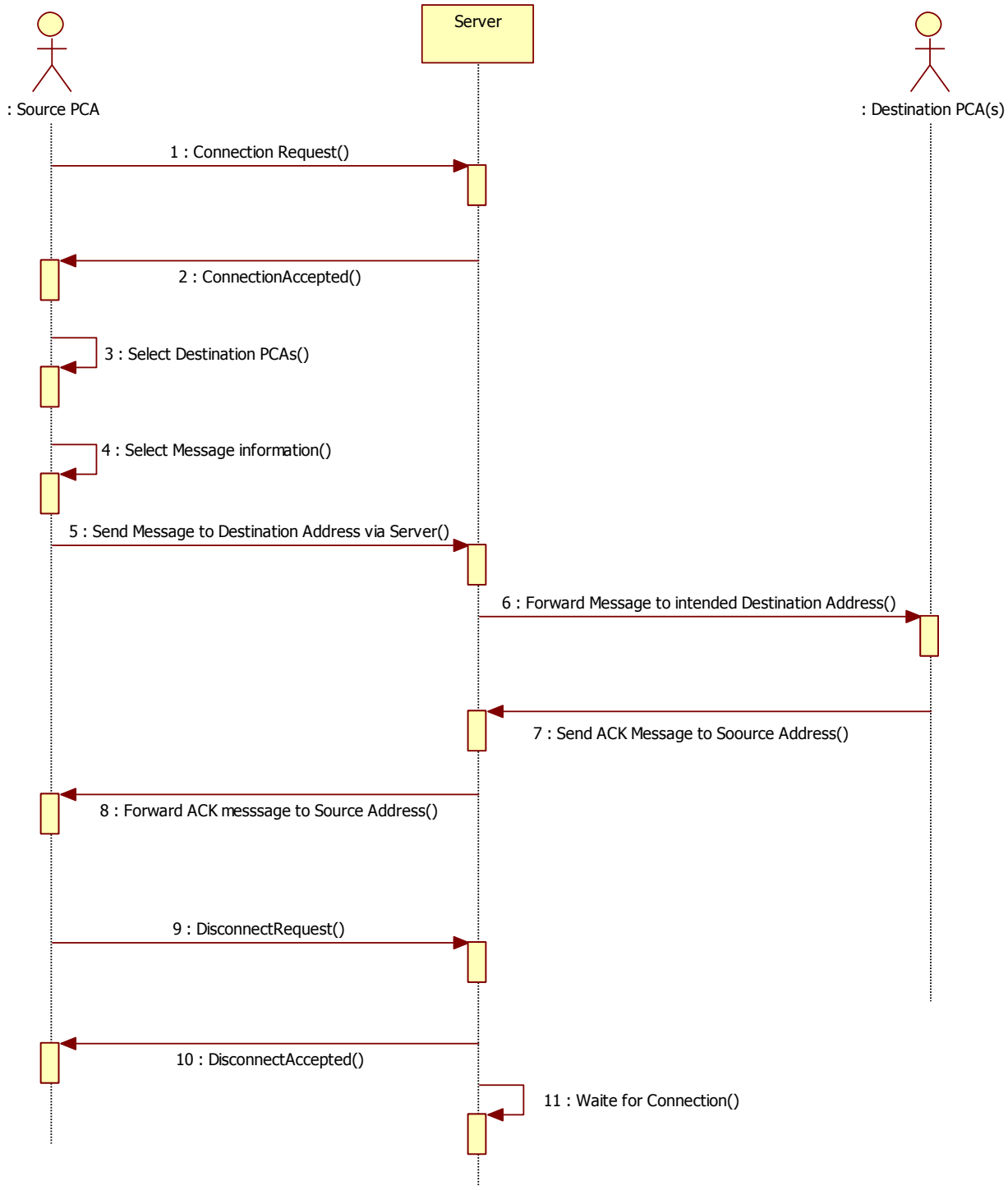
Class Diagram



Usecase Diagram



Sequence Diagram



4. CODING

Main.java:

```
package ronanki.swing;

import javax.swing.*;

import java.awt.event.*;

import java.awt.Desktop;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/**

 * @author K.Tata Rao

 **/

@SuppressWarnings("unused")

public class Main {

   /**

    * @param args the command line argumentsSystem.out.println("match"+mat.group());System.out.println("match"+mat.group());

    */

   public static void main(String[] args) {

      try {

         // Set System L&FPlease Select Your Device From ComboBox

            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

         }
```

```
        catch (ClassNotFoundException e) {
                // handle exception
           }
           catch (InstantiationException e) {
                // handle exception
           }
           catch (IllegalAccessException e) {
                // handle exception
           }
           catch (UnsupportedLookAndFeelException e) {
                // handle exception
           }
              java.awt.EventQueue.invokeLater(new Runnable() {
           public void run() {
              try {
                        final PcahostsimUI pcah = new PcahostsimUI();
                        pcah.setVisible(true);
               } catch (IOException e) {
                        e.printStackTrace();
                                }
           }
        });
        }
}
```

PcahostsimUI.java

```
package ronanki.swing;
import java.awt.Color;
import java.awt.Font;
import java.awt.Toolkit;
import java.awt.Window;
import java.awt.event.*;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.FocusEvent;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;
import java.awt.event.TextEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
```

```
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.lang.Thread.State;
import java.lang.reflect.Array;
import java.net.InetAddress;
import java.net.Socket;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.security.Timestamp;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.HashSet;
```

```
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.Executor;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;
import java.util.logging.FileHandler;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;
import java.util.regex.MatchResult;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.print.attribute.TextSyntax;
import javax.swing.AbstractButton;
import javax.swing.ComboBoxModel;
import javax.swing.GroupLayout;
```

```
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JScrollBar;
import javax.swing.SwingUtilities;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import javax.swing.text.Document;
import javax.swing.Timer;


@SuppressWarnings({ "serial", "unused" })
public class PcahostsimUI extends JFrame implements ActionListener,ItemListener,KeyListener
{
	final int RECVBUFFSIZE = 27; //number of bytes that server send as an acknowledgment
	int dd,cbcount=0,cbcount1,cbcount2,cbcount3,cbcount4,counterM,sendcntr,recvcntr;
	String source,destination,message,pcaaddr,pcasel,pcName,pcaselected,selectedpca,time;
	Boolean check,iocheck,checker;
	ArrayList<String> msg = new ArrayList<String>();
	ArrayList<String> msgid = new ArrayList<String>();
	ArrayList<String> msge = new ArrayList<String>();
```

```
ArrayList<String> msgeid = new ArrayList<String>();

ArrayList<String> setkey = new ArrayList<String>();

String[] list = { "SELECT YOUR PCA", "PCA0", "PCA1", "PCA2", "PCA3", "PCA4","PCA5", "PCA6", "PCA7", "PCA8" };

String[] pcaaddress = {"ADD000", "ADD001" , "ADD002" , "ADD003" , "ADD004" , "ADD005" , "ADD006" , "ADD007" , "ADD008" };

ArrayList<JCheckBox> cb ;

protected TCPClients client1;

ConfigBean detail = new ConfigBean();

BufferedWriter bufferedWriter;

ExecutorService executor;

@SuppressWarnings("rawtypes")

Future future;

private int width;

private int height;

Object msgs[];

Object msgsid[];

@SuppressWarnings({ "static-access", "rawtypes" })

public PcahostsimUI() throws IOException {

        setTitle("PCA Host Simulator");

        //initComponents();

        jPanel1 = new javax.swing.JPanel();

        ButtonSendCancel = new javax.swing.JButton();

        ButtonConDiscon = new javax.swing.JButton();

        TextPaneSentMsgs = new javax.swing.JTextArea();

        CbPca0 = new javax.swing.JCheckBox();
```

```
CbPca6 = new javax.swing.JCheckBox();

CbPca4 = new javax.swing.JCheckBox();

CbPca5 = new javax.swing.JCheckBox();

CbPca2 = new javax.swing.JCheckBox();

CbPca3 = new javax.swing.JCheckBox();

TextFieldMsgID = new javax.swing.JTextField();

CbPca1 = new javax.swing.JCheckBox();

LabelTo = new javax.swing.JLabel();

MsgSelCombo = new javax.swing.JComboBox();

CbPca8 = new javax.swing.JCheckBox();

CbPca7 = new javax.swing.JCheckBox();

LabelMessage = new javax.swing.JLabel();

CbSelectAll = new javax.swing.JCheckBox();

TextPaneRcvdMsgs = new javax.swing.JTextArea();

ScrollPaneSentMsgs = new javax.swing.JScrollPane();

ScrollPaneRcvdMsgs = new javax.swing.JScrollPane();

ScrollPaneRcvdMsgs.setVerticalScrollBarPolicy(ScrollPaneRcvdMsgs.VERTICAL_SCROLLBAR_NEVER);
        ScrollPaneSentMsgs.getVerticalScrollBar().setModel(ScrollPaneRcvdMsgs.getVerticalScrollBar().getModel());

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

Clear = new javax.swing.JButton();

MsgSelCombo.setEditable(true);

cb = new ArrayList<JCheckBox>();

cb.add(CbPca0);
```

```
cb.add(CbPca1);
cb.add(CbPca2);
cb.add(CbPca3);
cb.add(CbPca4);
cb.add(CbPca5);
cb.add(CbPca6);
cb.add(CbPca7);
cb.add(CbPca8);
BufferedReader input = null;
java.util.List<String> strings = new ArrayList<String>();
try {
 input = new BufferedReader(new FileReader("Message.txt")); // Reading messages
   from text file
        String line;
        while ((line = input.readLine()) != null) {
              strings.add(line);
        }
}catch (FileNotFoundException e) {
        System.err.println("Error, Message file didn't exist.");
} finally {
        try{
        input.close();
        }catch(NullPointerException ne){
        System.err.println("Make sure that Message.txt contains messages has exist");
        }
```

```
}
final Set<String> lineset = new TreeSet<String>(strings); // Treeset eliminates redundancy and maintain order of messages
//System.out.println(lineset);
TreeMap<Object, Object> tm = new TreeMap<Object, Object>();
Iterator si = lineset.iterator();
while (si.hasNext()) {
Pattern pat = Pattern.compile("\\s*(^[0-9]{3})\\s+([\\w+\\s+$&+,:;=?@#|'<>.-^*()%!]+)\\s*");
Matcher mat = pat.matcher((CharSequence) si.next());
while (mat.find()) {
msgid.add(mat.group(1));
msg.add(mat.group(2));
}
}
Object msgarray[] = msg.toArray();
Object msgidarray[] = msgid.toArray();
for (int i = 0; i <msgidarray.length; i++) {//Here, we can check message duplication, if found gives old message rather new message
if (tm.isEmpty()) {
tm.put(msgidarray[i], msgarray[i]);
} else {
Object[] setkey = tm.keySet().toArray();
for (Object key : setkey) {
int counted = 0;
String valuemap = (String) tm.get(key);
```

```
                    for (int j = 0; j < msgarray.length; j++) {
                        if (valuemap.equals(msgarray[j])) {
                            counted++;
                            if (counted == 1) {
                            } else {
                                msgarray[j] = null;      // if duplicate found, new message replaced by NULL in bufferr..
                                msgidarray[j] = null;
                            }
                        }
                    }
                }
            }
            if (msgidarray[i] != null && msgarray[i] != null)//it can only popup the messages without duplicate and null values...(eliminates null values)
                tm.put(msgidarray[i], msgarray[i]);
        }
        Set set = tm.entrySet();
        Iterator it = set.iterator();
        msge.add("SELECT YOUR MESSAGE");
        msgeid.add("");
        while (it.hasNext()) {
            final Map.Entry me = (Map.Entry) it.next();
            msge.add((String)me.getValue());
            msgeid.add((String)me.getKey());
        }
```

```
msgs = msge.toArray();
msgsid = msgeid.toArray();
dd = tm.size();
MsgSelCombo = new JComboBox(msgs);
MsgSelCombo.addItemListener(this);
// Reading messages and linking of textfield and combobox ending....
TextFieldMsgID.addKeyListener(this);

setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE)
;
this.addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent ev) {
MsgSelCombo.setSelectedItem(msgs[0]);
int windowres = JOptionPane.showConfirmDialog(null,
"Do you want to close the application?", "alert",
JOptionPane.OK_CANCEL_OPTION);
if (windowres == 0) {
if (jLabel2.getText().equals(
"Waiting ACK messages...")) {
JOptionPane.showMessageDialog(null, "Please wait until ACK
messages recieved");
checker = false;
}
else{

if(ButtonConDiscon.getText().equalsIgnoreCase("CONNECT"))
System.exit(0);
```

```
                                        else
if(ButtonConDiscon.getText().equalsIgnoreCase("DISCONNECT")){
                                        executor.shutdown();
                                        disconnect();
                                        System.exit(0);
                                        }
                                }
                        } else {
                        }
                }
                public void windowOpened(WindowEvent ev1) {
                enable(false);// Before CONNECT to our PCA, all other (Except CONNECT Button) GUI components has to be disabled
                enableCBX(false);
        ButtonSendCancel.setEnabled(false);
                }
        });
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
         if(screenSize.width==1366){
        width = ((screenSize.width)*78)/100;
        height = ((screenSize.height)*35)/100;
        }
        else if(screenSize.width==1360){
        width = ((screenSize.width)*79)/100;
        height = ((screenSize.height)*35)/100;
        }
```

```
else if(screenSize.width==1280){
width = ((screenSize.width)*87)/100;
height = ((screenSize.height)*35)/100;
}
else if(screenSize.width==1152){
width = ((screenSize.width)*92)/100;
height = ((screenSize.height)*35)/100;
}
else if(screenSize.width==1024){
        width = ((screenSize.width)*98)/100;
        height = ((screenSize.height)*35)/100;
        }
else if(screenSize.width==800){
        width = ((screenSize.width)*110)/100;
        height = ((screenSize.height)*35)/100;
        }
setMinimumSize(new java.awt.Dimension(width, height));
System.out.println(getMinimumSize()+","+width+",");
jPanel1.setAutoscrolls(true);
Clear.setText("CLEAR");
clearAll();
ButtonSendCancel.setText("SEND");
ButtonConDiscon.setText("CONNECT");
ButtonConDiscon.setPreferredSize(new Dimension(102, 0));
ButtonSendCancel.setPreferredSize(new Dimension(102, 0));
```

```
ButtonConDiscon.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

ButtonSendCancel.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

LabelMessage.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

LabelTo.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

MsgSelCombo.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

CbSelectAll.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

CbPca0.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

CbPca1.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

CbPca2.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

CbPca3.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

CbPca4.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

CbPca5.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

CbPca6.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

CbPca7.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));

CbPca8.setFont(new Font("Tahoma",Font.PLAIN,
((getMinimumSize().width*1)/100)));
```

```
TextFieldMsgID.setFont(new Font("Tahoma",Font.PLAIN, ((getMinimumSize().width*1)/100)));

TextPaneRcvdMsgs.setFont(new Font("Tahoma",Font.PLAIN, ((getMinimumSize().width*1)/100)));

TextPaneSentMsgs.setFont(new Font("Tahoma",Font.PLAIN, ((getMinimumSize().width*1)/100)));

Clear.setFont(new Font("Tahoma",Font.PLAIN, ((getMinimumSize().width*1)/100)));

jLabel1.setFont(new Font("Tahoma",Font.PLAIN, ((getMinimumSize().width*1)/100)));

jLabel2.setFont(new Font("Tahoma",Font.PLAIN, ((getMinimumSize().width*1)/100)));

if (ButtonConDiscon.getActionListeners().length < 1) {//if it is not checked...For each CONNECT & DISCONNECT

    //resultant consequences will executed number of times as CONNECT & DISCONNECT action has been performed

    ButtonConDiscon.addActionListener(this);

}

if (ButtonSendCancel.getActionListeners().length < 1) {//if it is not checked...For each CONNECT & DISCONNECT

    //resultant consequences will executed number of times as CONNECT & DISCONNECT action has been performed

    ButtonSendCancel.addActionListener(this);

}

Clear.addActionListener(this);

CbPca0.setText("PCA0");

CbPca6.setText("PCA6");

CbPca4.setText("PCA4");

CbPca5.setText("PCA5");
```

```
CbPca2.setText("PCA2");
CbPca3.setText("PCA3");
CbPca1.setText("PCA1");
CbSelectAll.setText("Select All");
CbPca8.setText("PCA8");
CbPca7.setText("PCA7");
CbPca0.addActionListener(this);
CbPca1.addActionListener(this);
CbPca2.addActionListener(this);
CbPca3.addActionListener(this);
CbPca4.addActionListener(this);
CbPca5.addActionListener(this);
CbPca6.addActionListener(this);
CbPca7.addActionListener(this);
CbPca8.addActionListener(this);
CbSelectAll.addActionListener(this);
LabelTo.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
LabelTo.setText("To");
LabelTo.setHorizontalTextPosition(javax.swing.SwingConstants.RIGHT);
LabelMessage.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
LabelMessage.setText("Message");
CbPca0.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
CbPca1.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
CbPca2.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
CbPca3.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
```

```
CbPca4.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
CbPca5.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
CbPca6.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
CbPca7.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
CbPca8.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
CbSelectAll.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
Clear.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(
                jPanel1);
jPanel1.setLayout(jPanel1Layout);
GroupLayout.Group hgroup = jPanel1Layout
.createSequentialGroup()
.addComponent(LabelTo,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                (getMinimumSize().width*7)/100,
                javax.swing.GroupLayout.PREFERRED_SIZE)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED);
for(JCheckBox cbox : cb ){
        hgroup.addComponent(cbox);
    hgroup.addGap(((getMinimumSize().width)/100)+1,((getMinimumSize().width)/100)+1,(
(getMinimumSize().width)/100)+1);
}
hgroup.addGap(0,0,0)
```

```
                .addComponent(CbSelectAll,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                (getMinimumSize().width*9)/100,
                                Short.MAX_VALUE);
                jPanel1Layout.setHorizontalGroup(jPanel1Layout.createParallelGroup(
                                                javax.swing.GroupLayout.Alignment.LEADING)
                                                .addGroup(jPanel1Layout.createSequentialGroup()
                                                .addContainerGap()
                                                .addGroup(jPanel1Layout.createParallelGroup(
                                                javax.swing.GroupLayout.Alignment.LEADING)
                                                .addGroup(
                                                javax.swing.GroupLayout.Alignment.TRAILING,
                                                        jPanel1Layout.createSequentialGroup()

        .addComponent(ScrollPaneRcvdMsgs,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                        (getMinimumSize().width*41)/100,
                        Short.MAX_VALUE)
                        .addPreferredGap(
        javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                                .addComponent(ScrollPaneSentMsgs,
                                        javax.swing.GroupLayout.DEFAULT_SIZE,
                                                (getMinimumSize().width*41)/100,
                                                Short.MAX_VALUE))
                                .addGroup(jPanel1Layout.createSequentialGroup()
                                .addComponent(ButtonConDiscon,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                        (getMinimumSize().width*10)/100,
                javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap((getMinimumSize().width)/100,(getMinimumSize().width)/100,(getMinimumSize().wi
dth)/100)

        .addGroup(jPanel1Layout.createParallelGroup(
        javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(hgroup)
```

```
                .addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(LabelMessage,javax.swing.GroupLayout.PREFERRED_SIZE,
        (getMinimumSize().width*7)/100,javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

        .addComponent(TextFieldMsgID,javax.swing.GroupLayout.PREFERRED_SIZE,

        (getMinimumSize().width*5)/100,javax.swing.GroupLayout.PREFERRED_SIZE)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(MsgSelCombo,
        0,(getMinimumSize().width*70)/100,Short.MAX_VALUE)))

        .addGap((getMinimumSize().width)/100,(getMinimumSize().width)/100,(getMinimumSi
ze().width)/100)

        .addComponent(ButtonSendCancel,javax.swing.GroupLayout.PREFERRED_SIZE,

        (getMinimumSize().width*10)/100,javax.swing.GroupLayout.PREFERRED_SIZE))

        .addGroup(jPanel1Layout.createSequentialGroup()
        .addComponent(jLabel1,javax.swing.GroupLayout.PREFERRED_SIZE,


        (getMinimumSize().width*13)/100,javax.swing.GroupLayout.PREFERRED_SIZE)

         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,

                (getMinimumSize().width*71)/100, Short.MAX_VALUE)

        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,

                (getMinimumSize().width*13)/100,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,

        (getMinimumSize().width*62)/100,Short.MAX_VALUE)

        .addComponent(Clear, javax.swing.GroupLayout.PREFERRED_SIZE,

                (getMinimumSize().width*8)/100,
javax.swing.GroupLayout.PREFERRED_SIZE)))

        .addContainerGap()));
```

```
jPanel1Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] { CbPca0, CbPca1, CbPca2, CbPca3, CbPca4,CbPca5, CbPca6, CbPca7,
CbPca8});

        GroupLayout.Group vgroup = jPanel1Layout.createParallelGroup(

                javax.swing.GroupLayout.Alignment.BASELINE)

        .addComponent(LabelTo)

        .addComponent(CbSelectAll);

        for(JCheckBox cbox : cb ){

            vgroup.addComponent(cbox);

        }

        jPanel1Layout.setVerticalGroup(jPanel1Layout      .createParallelGroup(

            javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(jPanel1Layout.createSequentialGroup()

            .addContainerGap()

            .addGroup(jPanel1Layout.createParallelGroup(

    javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(jPanel1Layout.createParallelGroup(

                    javax.swing.GroupLayout.Alignment.TRAILING, false)

            .addGroup(javax.swing.GroupLayout.Alignment.LEADING,

                jPanel1Layout.createSequentialGroup()

            .addGroup(jPanel1Layout.createParallelGroup(
                    javax.swing.GroupLayout.Alignment.BASELINE)

                .addComponent(LabelMessage)

                .addComponent(TextFieldMsgID,
                javax.swing.GroupLayout.PREFERRED_SIZE,

                javax.swing.GroupLayout.DEFAULT_SIZE,
```

javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(MsgSelCombo,

javax.swing.GroupLayout.PREFERRED_SIZE,

javax.swing.GroupLayout.DEFAULT_SIZE,

javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,

javax.swing.GroupLayout.DEFAULT_SIZE,Short.MAX_VALUE)

.addGroup(vgroup))

.addComponent(ButtonConDiscon,
javax.swing.GroupLayout.PREFERRED_SIZE,

53,javax.swing.GroupLayout.PREFERRED_SIZE))

.addComponent(ButtonSendCancel,
javax.swing.GroupLayout.PREFERRED_SIZE,

53,javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(

javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(ScrollPaneSentMsgs,

javax.swing.GroupLayout.DEFAULT_SIZE,169,Short.MAX_VALUE)
.addComponent(ScrollPaneRcvdMsgs, javax.swing.GroupLayout.DEFAULT_SIZE,
169,Short.MAX_VALUE))

.addPreferredGap(
javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(

javax.swing.GroupLayout.Alignment.BASELINE)

```
                .addComponent(jLabel1)
                .addComponent(jLabel2)
                .addComponent(Clear))));
        getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);
        //pack();
    }
    @SuppressWarnings({ })
    //private void initComponents() throws IOException {}// </editor-fold>//GEN-END:initComponents
    @Override // Functioning for CONNECT and DISCONNECT------->Action listener start
    public void actionPerformed(ActionEvent aecon) {
        //System.out.println("source of event listener "+aecon.getSource());
        if(aecon.getSource().equals(ButtonConDiscon)){
            String action = aecon.getActionCommand();
            if (action.equalsIgnoreCase("CONNECT")) {
                                                    check = true;
                                                iocheck = true;
            startup();// It shows list of PCAs from which you have to select one of yours to connect
        ButtonConDiscon.setText("DISCONNECT");
        client1 = new TCPClients(detail.getServeraddess(), detail.getServerport());
        logfile();//whenever you connected, logfile will be created with the current time as name of its
                                    // TCP connection started
                                     // newSingleThreadExecutor() is taken for Thread safe,
         executor = Executors.newSingleThreadExecutor();//because it only executes single thread at any time
```

```
                                        //future = executor.submit(client1.new
ReceivingThread());//creates receiving thread to listen server response until connection is opened
        enableCBX(true);

        ButtonConDiscon.setBackground(Color.LIGHT_GRAY);

        MsgSelCombo.setSelectedItem(msgs[0]);

        TextPaneRcvdMsgs.setEditable(false);

        TextPaneRcvdMsgs.setForeground(new java.awt.Color(51,0, 255));

        TextPaneSentMsgs.setEditable(false);

        TextPaneSentMsgs.setForeground(new java.awt.Color(      153,0,153));

        TextPaneRcvdMsgs.setMinimumSize(new java.awt.Dimension(     54, 20));

        ScrollPaneRcvdMsgs.setViewportView(TextPaneRcvdMsgs);

        TextPaneSentMsgs.setMinimumSize(new java.awt.Dimension(54,20));

        ScrollPaneSentMsgs.setViewportView(TextPaneSentMsgs);

        checkboxselct(); // used to upadte PCA checkboxes after u connected to your PCA

        }

        else if (action.equalsIgnoreCase("DISCONNECT")) {
                        if (jLabel2.getText().equals("Waiting ACK messages...")) {

                JOptionPane.showMessageDialog(null, "Please wait until ACK messages
recieved");

                        check = false;

                        } else {

                                executor.shutdown();

                                disconnect();//it closes socket connection and clears all
components to its factory version

                                        }

                        enableCBX(false);

                        }// ActionEvent for JButton end
```

```
}
else if(aecon.getSource().equals(Clear)){
        TextPaneSentMsgs.setText(null);
        TextPaneRcvdMsgs.setText(null);
}
else if(aecon.getSource().equals(CbSelectAll)){
        selectall();
        checkbox();
}
else if(aecon.getSource().equals(CbPca0) ||
                aecon.getSource().equals(CbPca1) ||
                aecon.getSource().equals(CbPca2) ||
                aecon.getSource().equals(CbPca3) ||
                aecon.getSource().equals(CbPca4) ||
                aecon.getSource().equals(CbPca5) ||
                aecon.getSource().equals(CbPca6) ||
                aecon.getSource().equals(CbPca7) ||
                aecon.getSource().equals(CbPca8)
                ){
        if(selectflag==true){
                CbSelectAll.setSelected(false);
                selectflag = false;
        }
        checkbox();//this method provides functionality that, if atleast single PCA
get selected,then only user able to select Message
```

```
}
else if(aecon.getSource().equals(ButtonSendCancel)){
        String action1 = aecon.getActionCommand();
        if (action1.equalsIgnoreCase("SEND")) {
                cbcount=0;
                sendcntr = 0;
        recvcntr = 0 ;
        checker = true;
        ButtonSendCancel.setText("CANCEL");
        ButtonSendCancel.setBackground(Color.LIGHT_GRAY);
                enable(false);
        for(JCheckBox cbxc : cb){
        if(cbxc.isSelected()) cbcount++;
        }
        cbcount1 = cbcount;
        cbcount2 = cbcount;
        cbcount3 = cbcount;
        cbcount4 = cbcount;
        if(cbcount==1){
        for(JCheckBox cbx : cb)        {
        if(cbx.isSelected()){
         pcasel = cbx.getText();
        TextPaneRcvdMsgs.append("-->>\n                                                To:
"+pcasel+"\n|"+gettime()+"|MSG"+TextFieldMsgID.getText()+"|"+MsgSelCombo.getSelectedIt
em()+"\n");
        TextPaneSentMsgs.append("\n\n\n\n");
        destination = pcadest(cbx.getText());
```

```
message = source + destination + TextFieldMsgID.getText() + MsgSelCombo.getSelectedItem();

jPanel1.setPreferredSize(getMaximumSize());

        }

}

        sendMessage(message,client1);

        future = executor.submit(client1.new ReceivingThread());//creates receiving thread to listen server response until connection is opened

        jLabel2.setText("Waiting ACK messages...");

        }

        else if(cbcount>1){

        TextPaneRcvdMsgs.append("-->>\n To: ");

        for(JCheckBox cbxx : cb)      {

        if(cbxx.isSelected()){

        TextPaneRcvdMsgs.append(cbxx.getText());

        if(--cbcount1>0) TextPaneRcvdMsgs.append(",");

        jPanel1.setPreferredSize(getMaximumSize());
        }

}

        TextPaneRcvdMsgs.append("\n|"+gettime()+"|MSG"+TextFieldMsgID.getText()+"|"+MsgSelCombo.getSelectedItem()+"\n");

        TextPaneSentMsgs.append("\n\n\n\n");

        for(JCheckBox cbxx : cb){

        if(cbxx.isSelected()){

        destination = pcadest(cbxx.getText());

        message = source + destination + TextFieldMsgID.getText() + MsgSelCombo.getSelectedItem();
```

```
        sendMessage(message,client1);

        future = executor.submit(client1.new ReceivingThread());//creates receiving thread to
listen server response until connection is opened

        }

}

jLabel2.setText("Waiting ACK messages...");

}

logfileUpdateS();

Clear.setEnabled(false);

}

if (action1.equalsIgnoreCase("CANCEL")) {
                ButtonSendCancel.setBackground(new JButton().getBackground());

        ButtonSendCancel.setText("SEND");

        clearAll();

        enableCBX(true);

        jLabel2.setText("");

        Clear.setEnabled(true);

}

}

}// Functioning for CONNECT and DISCONNECT------->Action listener end

        //Functioning for Combo box to select message------->item listener start

        public void itemStateChanged(ItemEvent ie) {

                if (ButtonConDiscon.getText() != "CONNECT") {

                        try {

                                for (int b = 0; b <= dd; b++) {

                                        if (MsgSelCombo.getSelectedItem() == msgs[b]) {
```

```
                                        TextFieldMsgID.setText((String) msgsid[b]);
                                }
                        }
                } catch (Exception e) {
                        // TODO: handle exception
                }
        }
        if(MsgSelCombo.getSelectedItem()==msgs[0]){
        ButtonSendCancel.setEnabled(false);
        }
        else{
                enableCBX(false);
                ButtonSendCancel.setEnabled(true);
        }
} //Functioning for Combo box to select message------->item listener end
// Functioning for message id text field to have message id------->key listener start
public void keyTyped(KeyEvent ke) {
        enableCBX(false);
        //ButtonSendCancel.setEnabled(true);
        int id = ke.getID();
        if(id==KeyEvent.KEY_TYPED)
        {
        char c = ke.getKeyChar();
        //System.out.println("key ids:"+ke.getKeyChar()+"hi");
        if(ke.getKeyChar()!='\n')
```

```
                    //MsgSelCombo.setSelectedItem(msgs[0]);
                    ButtonSendCancel.setEnabled(false);
//                  System.out.println("Clicks on ENTER");
            if (!((c >= '0') && (c <= '9') ||
              (c == KeyEvent.VK_BACK_SPACE) ||
              (c == KeyEvent.VK_DELETE))) {
             getToolkit().beep();
             ke.consume();
            }
          }
        }
        public void keyReleased(KeyEvent ke) {
        }
        public void keyPressed(KeyEvent ke) {
            // TODO Auto-generated method stub
            int code = ke.getKeyCode();
            String enterkey = KeyEvent.getKeyText(code);
            if(enterkey=="Enter")
            {
            String msgt = null;
            String n = TextFieldMsgID.getText();
            //System.out.println(n);
            try {
                        for (int i = 0; i <=dd; i++) {   //16 12 2014 .....
                            if (n.equalsIgnoreCase((String) msgsid[i])) {
```

```
                    MsgSelCombo.setSelectedItem(msgs[i]);
                    TextFieldMsgID.setFocusable(true);

                msgt = (String) msgs[i];

                ButtonSendCancel.setEnabled(true);

                }
            }

                if (msgt==null) {

                JOptionPane.showMessageDialog(null, "Message ID you selected not exist, Try another one");

                    MsgSelCombo.setSelectedItem(msgs[0]);
                    TextFieldMsgID.setText(msgsid[0].toString());

                    ButtonSendCancel.setEnabled(false);

                }


            } catch (Exception e) {

                // TODO: handle exception

            }

        }

        }// Functioning for message id text field to have message id------->key listener end

        private void disconnect(){

        clearAll();

        TextPaneRcvdMsgs.setText(null);

        TextPaneSentMsgs.setText(null);

        ButtonConDiscon.setBackground(new JButton()      .getBackground());

        ButtonConDiscon.setText("CONNECT");

        ButtonSendCancel.setText("SEND");
```

```
        jLabel1.setText("");
        jLabel2.setText("");
        for(JCheckBox cbox : cb){
                cbox.setVisible(true);
        }
        enableCBX(false);
        enable(false);
        try {
                if(executor.isShutdown()){
                client1.tcpSocket.shutdownInput();
                client1.tcpSocket.close();
                }
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }

}
// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton ButtonConDiscon;
private javax.swing.JButton ButtonSendCancel;
private javax.swing.JCheckBox CbPca0;
private javax.swing.JCheckBox CbPca1;
private javax.swing.JCheckBox CbPca2;
private javax.swing.JCheckBox CbPca3;
```

```
private javax.swing.JCheckBox CbPca4;
private javax.swing.JCheckBox CbPca5;
private javax.swing.JCheckBox CbPca6;
private javax.swing.JCheckBox CbPca7;
private javax.swing.JCheckBox CbPca8;
private javax.swing.JCheckBox CbSelectAll;
private javax.swing.JLabel LabelMessage;
private javax.swing.JLabel LabelTo;
private javax.swing.JComboBox MsgSelCombo;
private javax.swing.JScrollPane ScrollPaneRcvdMsgs;
private javax.swing.JScrollPane ScrollPaneSentMsgs;
private javax.swing.JTextField TextFieldMsgID;
private javax.swing.JTextArea TextPaneRcvdMsgs;
private javax.swing.JTextArea TextPaneSentMsgs;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JPanel jPanel1;
private javax.swing.JButton Clear;
private String fileName;
protected boolean selectflag;
// End of variables declaration//GEN-END:variables
public void enable(boolean flags) {
        TextFieldMsgID.setEnabled(flags);
        MsgSelCombo.setEnabled(flags);
}
```

```
    public void enableCBX(boolean flags){
            for(JCheckBox cbox : cb){
                    cbox.setEnabled(flags);
            }
            CbSelectAll.setEnabled(flags);
    }
    public void clearAll() {
  CbPca0.setSelected(false);
    CbPca1.setSelected(false);
    CbPca2.setSelected(false);
    CbPca3.setSelected(false);
    CbPca4.setSelected(false);
    CbPca5.setSelected(false);
    CbPca6.setSelected(false);
    CbPca7.setSelected(false);
    CbPca8.setSelected(false);
    CbSelectAll.setSelected(false);
    MsgSelCombo.setSelectedItem(msgs[0]);
    TextFieldMsgID.setText(msgsid[0].toString());
    if(CbSelectAll.isSelected()) selectflag = false;
    }
    public void startup(){
            do{
       pcaselected = (String) JOptionPane.showInputDialog(null,     "Choose your PCA
Device",
```

```
        "Your Device",  JOptionPane.QUESTION_MESSAGE,  null,  list,list[0]);
 if(pcaselected==null)
        System.exit(0);
else if(pcaselected!=null && pcaselected!=list[0])
{
        selectedpca = pcaselected;
        jLabel1.setText("Connected as " + selectedpca);
        for(int p=0;p<list.length;p++){
                if(list[p]==selectedpca){
                        source = pcaaddress[p-1];
                        System.out.println(source);
                }
        }
}
        }while(pcaselected==list[0]);
        for(JCheckBox cbx : cb){
                if(selectedpca==cbx.getText())
                {
                        cbx.setVisible(false);
                        cbx.setEnabled(false);
                }
        }
        }
```

public String pcadest(String pcaName) // method for to get destination address in order to form message, which has to be send

```
{
        for(int q=0;q<list.length;q++){
                if(pcaName.compareToIgnoreCase(list[q])==0){
                        pcaaddr = pcaaddress[q-1];
                }
        }
        return pcaaddr;
}
```

public String pcadestRev(String pcaAdd) // method for to get destination address in order to print PCA which sent ACK

```
{
        for(int q=0;q<pcaaddress.length;q++){
                if(pcaAdd.compareToIgnoreCase(pcaaddress[q])==0){
                        pcName = list[q+1];
                }
        }
        return pcName;
}
public void checkboxselct(){
        for (JCheckBox cbox : cb) {
                if(selectedpca==cbox.getText())
                {
                        cbox.setVisible(false);
                        cbox.setEnabled(false);
                }
```

```
            }
        }
public void selectall(){
            for (JCheckBox cbox : cb) {
                if (CbSelectAll.isSelected()) {
                    selectflag = true;
                    if (cbox.isVisible())
                    cbox.setSelected(true);
                     else
                            cbox.setSelected(false);
                } else {
                    selectflag = false;
                    cbox.setSelected(false);
                }
            }
        }
        public void checkbox(){
            if (CbPca0.isSelected()    || CbPca1.isSelected()
                                    || CbPca2.isSelected()
                                    || CbPca3.isSelected()
                                    || CbPca4.isSelected()
                                    || CbPca5.isSelected()
                                    || CbPca6.isSelected()
                                    || CbPca7.isSelected()
                                    || CbPca8.isSelected()
```

```
                                        || CbSelectAll.isSelected()) {
                        enable(true);
                }
                else{
                        enable(false);
                }
        }
         public void receiving(String innerMessage){
                 String pc = pcadestRev(innerMessage.substring(1, 7));
                 if(cbcount==1){
                                TextPaneRcvdMsgs.append("\n\n\n\n");
                                TextPaneSentMsgs .append("<<--\n From:"+
pc+"\n|"+gettime()+"|"+innerMessage.substring(20, 26)+"|ACK\n");
        clearAll();
                        enable(false);
                        enableCBX(true);
                        jLabel2.setText("");
                        logfileUpdateR(innerMessage);
                        Clear.setEnabled(true);
                        if(check == false){
                                running = false;
                                executor.shutdown();
                                disconnect();
                        }
                        if(checker == false){
```

```
                    executor.shutdown();
                    disconnect();
                    System.exit(0);
                }
            }
            if(cbcount>1){
                sendcntr++;
                TextPaneRcvdMsgs.append("\n\n\n\n");
                TextPaneSentMsgs.append("<<--\n
From:"+pc+"\n|"+gettime()+"|"+innerMessage.substring(20, 26)+"|ACK\n");
                if(cbcount2>1)
                    TextPaneSentMsgs.append("\n");
                    cbcount2--;
                    logfileUpdateR(innerMessage);
                    if(sendcntr==cbcount){
                    clearAll();
                    enable(false);
                    enableCBX(true);
                    jLabel2.setText("");
                    Clear.setEnabled(true);
                    if(check==false){
                        executor.shutdown();
                        disconnect();
                        }
                        if(checker == false){
                                executor.shutdown();
```

```
                                        disconnect();
                                        System.exit(0);
                                    }
                                    }
                            }
        }
public static byte[] string2Byte(String messageStb) {
                byte[] messageB = null;
                messageStb = messageStb.trim();
                try {
                        messageB = messageStb.getBytes("UTF-8");
                } catch (UnsupportedEncodingException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

                byte messageBFinal[]=new byte[(messageB.length+2)];
                messageBFinal[0] = (byte) 0x02 ;
                for(int i=1; i<=messageB.length; i++) {
                        messageBFinal[i]=messageB[i-1];
                }
                messageBFinal[(messageBFinal.length-1)] = (byte)0x03 ;
                return messageBFinal;
        }
        public String gettime(){
```

```
                final Calendar now = Calendar.getInstance();
                final DateFormat dateFormat = new SimpleDateFormat("HH:mm:ss");
                 time = dateFormat.format(now.getTime());
                return time;
        }
        public void sendMessage(String message, TCPClients client1){
        String messageS = message;
        byte messageB[] = PcahostsimUI.string2Byte(messageS);
                client1.sendBytes(messageB);
                recvcntr++;
                if(cbcount==1){
      ButtonSendCancel.setText("SEND");
      ButtonSendCancel.setEnabled(false);
}
                else if(cbcount>1){
                        if(recvcntr==cbcount){
            ButtonSendCancel.setText("SEND");
            ButtonSendCancel.setEnabled(false);
        }
                }
        }

public void logfile(){
   final Logger logger = Logger.getLogger(PcahostsimUI.class .getName());
         boolean makeDir = false;
```

```
Date date = new Date();
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH-mm-ss") ;
String curDate =dateFormat.format(date);
FileHandler fh = null;
String filename = null;
try {
String filepath = null;
if(detail.getLogfilepath()!=""){
String temppath = null;

if(System.getProperty("os.name").contains("Windows")){
temppath = detail.getLogfilepath().replace('/', '\\');
}
else if(System.getProperty("os.name").contains("Linux")){
temppath = detail.getLogfilepath().replace('\\', '/');
}
if(temppath.endsWith("\\"+selectedpca+"logs\\") ||
temppath.endsWith("\\"+selectedpca+"logs") ||
temppath.endsWith("//"+selectedpca+"logs//") ||
temppath.endsWith("//"+selectedpca+"logs") ){
filepath = temppath;
}
else{
filepath = temppath;
if(System.getProperty("os.name").contains("Windows")){
```

```
                    filepath = filepath+"\\"+selectedpca+"logs\\";
                }
                else if(System.getProperty("os.name").contains("Linux")){
                    filepath = filepath+"/"+selectedpca+"logs/";
                        }
                }
                }
            else {
                if(System.getProperty("os.name").contains("Windows")){
                    filepath = new
java.io.File(".").getCanonicalPath()+"\\"+selectedpca+"logs\\";
                }
                else if(System.getProperty("os.name").contains("Linux")){
                    filepath = new
java.io.File(".").getCanonicalPath()+"/"+selectedpca+"logs/";
                    }

        }
            File file = new File(filepath);
                if(file.exists())
                makeDir = file.mkdir();
                else{
                    if(System.getProperty("os.name").contains("Windows")){
                filepath = new
java.io.File(".").getCanonicalPath()+"\\"+selectedpca+"logs\\";                }
                else if(System.getProperty("os.name").contains("Linux")){
```

```
                filepath = new java.io.File(".").getCanonicalPath()+"/"+selectedpca+"logs/";
                }
        System.err.println("Specified path not existed, logs path switched to default path:"+filepath);
            file = new File(filepath);
             makeDir = file.mkdir();
            }
                filename = curDate+".txt";
                 if(System.getProperty("os.name").contains("Windows")){
                     fh = new FileHandler(file+"\\"+filename,true);
                     setFilename(filepath+"\\"+filename);
                    }
                    else if(System.getProperty("os.name").contains("Linux")){
                                fh = new FileHandler(file+"//"+filename,true);
                                setFilename(filepath+"//"+filename);
                                }
                logger.addHandler(fh);
                logger.setUseParentHandlers(false);
                // Set the logger level to produce logs at this level and above.
                logger.setLevel(Level.FINE);
                SimpleFormatter formatter = new SimpleFormatter();
                fh.setFormatter(formatter);
              } catch (SecurityException ex) {
                System.err.println("Please, make sure that you have permissions to create a file/directory");
```

```
                } catch (IOException e) {
                    System.err.println("Please, make sure that your log directory exists");
                }catch(NullPointerException e){
                                System.err.println("logfile path node is missed in conf.xml");
                                System.exit(1);
                            }
                try{
                fh.close();
                }
                catch(NullPointerException e){
                    System.err.println("Please Change your log file path or leave as empty");
                }
              }
            public void setFilename(String filename){
                    this.fileName = filename;
            }
            public String getFilename(){
                    return fileName;
            }
            public void logfileUpdateS(){
                    try {
                    FileWriter fileWriter = new FileWriter(getFilename(), true);
                    bufferedWriter = new BufferedWriter(fileWriter);
                    if(cbcount==1){
```

```
                for(JCheckBox cbx : cb)
                {
            if(cbx.isSelected()){
                 pcasel = cbx.getText();
                                bufferedWriter.append("-->>");
                        bufferedWriter.newLine();
                        bufferedWriter.append(" To: "+pcasel);
                        bufferedWriter.newLine();
    bufferedWriter.append("|"+gettime()+"|MSG"+TextFieldMsgID.getText()+"|"+MsgSelCombo.getSelectedItem());
                        bufferedWriter.newLine();
                }
            }
            }
            else if(cbcount>1){
                bufferedWriter.append("-->>");
                bufferedWriter.newLine();
                bufferedWriter.append(" To: ");
                for(JCheckBox cbxx : cb)
                {
                        if(cbxx.isSelected()){
                                bufferedWriter.append(cbxx.getText());
                        if(--cbcount3>0) bufferedWriter.append(",");
                                }
                }
```

```
                    bufferedWriter.newLine();
        bufferedWriter.append("|"+gettime()+"|MSG"+TextFieldMsgID.getText()+"|"+MsgSelCombo.getSelectedItem());
                    bufferedWriter.newLine();
        }
                bufferedWriter.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                iocheck = false ;
            System.err.println("logs directory missed, please disconnect and reconnect to get logs");
            //System.out.println("\n");
        }catch(NullPointerException ne){
                    System.err.println("You lost Your logging information");
                }
            }
          public void logfileUpdateR(String innerMessage){
            String pc = pcadestRev(innerMessage.substring(1, 7));
            try {
                        FileWriter fileWriter = new FileWriter(getFilename(), true);
                        bufferedWriter = new BufferedWriter(fileWriter);
             if(cbcount==1){
                        bufferedWriter.newLine();
                        bufferedWriter.append("<<--");
                        bufferedWriter.newLine();
```

```
                    bufferedWriter.append(" From:"+ pc);
                    bufferedWriter.newLine();

        bufferedWriter.append("|"+gettime()+"|"+innerMessage.substring(20, 26)+"|ACK");
                    bufferedWriter.newLine();
                }
            if(cbcount>1){
                                bufferedWriter.append("<<--
");
                                bufferedWriter.newLine();
                                bufferedWriter.append("
From:"+pc);
                                bufferedWriter.newLine();

        bufferedWriter.append("|"+gettime()+"|"+innerMessage.substring(20, 26)+"|ACK");
                                bufferedWriter.newLine();
                                if(cbcount4>1)
                                    bufferedWriter.newLine();
                                    cbcount4--;
                 }
            bufferedWriter.close();
           } catch (IOException e) {
                // TODO Auto-generated catch block
                if(iocheck==true)
                System.err.println("logs directory missed, please disconnect and
reconnect to get logs");
                }
```

```
}
public class TCPClients {
    private InetAddress serverAddress;
    private Socket tcpSocket;
    private BufferedReader serverResponse;
    private int serverPort;
    private OutputStream out;
    private DataOutputStream dos ;
    String innerMessage;
    public TCPClients(String host, int port){
        try {
            serverAddress = InetAddress.getByName(host);
            serverPort = port;
            this.tcpSocket = new Socket(serverAddress, serverPort);
            this.serverResponse = new BufferedReader(new
InputStreamReader(tcpSocket.getInputStream()));
           // new PrintWriter(tcpSocket.getOutputStream());
            this.out = tcpSocket.getOutputStream();
            this.dos = new DataOutputStream(out);
        } catch (SocketException e) {
            //e.printStackTrace();
            errorMessages(e);
        } catch (UnknownHostException e) {
            errorMessages(e);
```

```
    } catch (IOException e) {
        errorMessages(e);
    }
}

public void sendBytes(byte[] messagebyte){
synchronized(this) {
    if(!this.tcpSocket.isConnected()) {
        System.out.println("Unable to connect to the Socket");
        return;
    }
    try {
            if(messagebyte.length>2048)
            System.err.println("packet size exceeds the maximum limit 2048
bytes");
            else
    this.dos.write(messagebyte);
    String messageS = this.byte2String(messagebyte);
    innerMessage = messageS;
    this.dos.flush();
    } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
    }
```

```
}
public String byte2String(byte[] messageB) {
      byte[] messageBTemp = new byte[(messageB.length-2)] ;
      for(int i=0; i<(messageB.length-2); i++) {
            messageBTemp[i] = messageB[i+1];
      }
      String messageS = new String(messageBTemp);

      return messageS;
}
public class ReceivingThread implements Callable<String> {
  char[] buffer = new char[RECVBUFFSIZE];
  public String call() {
      while(true){
            try {
                int sign = serverResponse.read(buffer, 0, RECVBUFFSIZE);
                  String message = new String(buffer, 0, sign);
                                    receiving(message);
                  return message;
            }catch (IOException e) {
                  //e.printStackTrace();
                        System.err.println("Problem with the Connection");
            }
       }
      }
}
```

```
                public void errorMessages(Exception e) {
                        String serverAddress1 = serverAddress.getHostName();
                        String serverAddress2 = serverAddress.getHostAddress();
                        System.err.println("########################## ERROR !... ##########################");
                        System.out.println("");
                        String message = e.getMessage();
                        System.err.println(message);
                        if(message.equalsIgnoreCase("Network is unreachable")) {
                                System.err.println("Please Check whether you are connected to the Network");
                        }
                        if(message.equalsIgnoreCase("Connection refused")||message.equalsIgnoreCase("No route to host")||message.equalsIgnoreCase("Connection refused: connect")) {
                        System.err.println("Please check whether your server is running or not");
                        System.err.println("Are the following details correct ? : ");
                        System.err.println("Server HostName: "+serverAddress1+" Address: "+serverAddress2+" Port Number: "+serverPort);
                                }
                                System.exit(1);
                        }
                }
}
```

ConfigBean.java

```
package ronanki.swing;

import java.io.File;
```

```
import java.io.IOException;
import java.io.Serializable;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.DOMException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
@SuppressWarnings("serial")
public class ConfigBean implements Serializable {
	static Element eElement;
	private String address,path;
	private int port;
	String[] server = new String[3];
	public ConfigBean(){
		try {
		File fXmlFile = new File("conf.xml");
			DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
			DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
			Document doc = dBuilder.parse(fXmlFile);
			doc.getDocumentElement().normalize();
```

```
                NodeList nList = doc.getElementsByTagName("Attributes");
                for (int temp = 0; temp < nList.getLength(); temp++) {
                        Node nNode = nList.item(temp);
                        if (nNode.getNodeType() == Node.ELEMENT_NODE) {
                                eElement = (Element) nNode;
                server[temp] =
eElement.getElementsByTagName("value").item(0).getTextContent();
                        }
                }
                setServeraddress(server[0]);
                setServerport(Integer.parseInt(server[1]));
                setLogfilepath(server[2]);
        } catch (DOMException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (ParserConfigurationException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (SAXException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                //e.printStackTrace();
                System.err.println("Could not locate the conf file,Please verify conf.xml");
                System.exit(1);
```

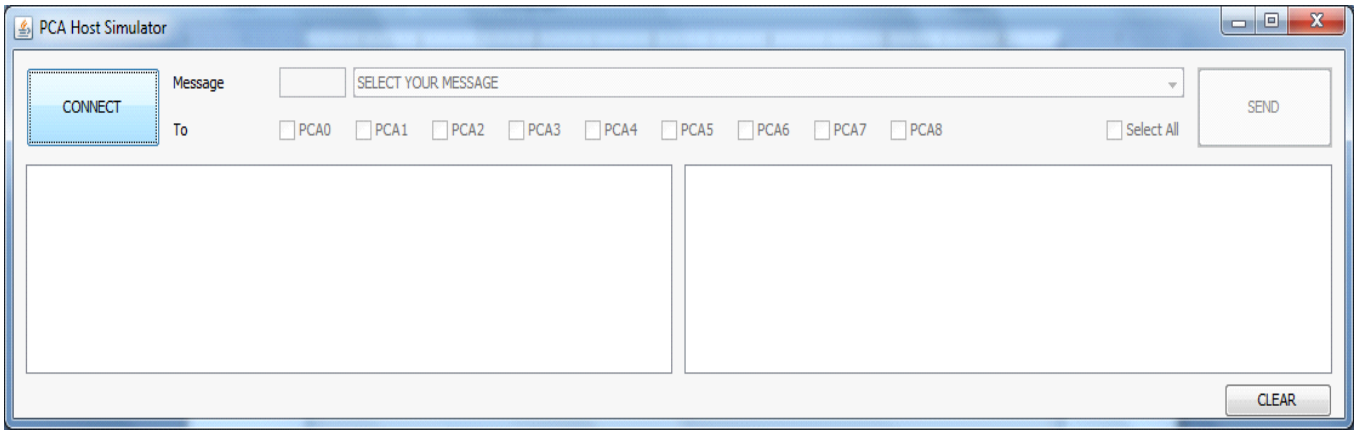
```
                }
                catch(ArrayIndexOutOfBoundsException ae){
                System.err.println("Please do not add more nodes, conf.xml should contain only three nodes");
                System.exit(1);
                }
                catch(NumberFormatException ne){
                        System.err.println("serverPort or serverAddress node missed in conf.xml");
                        System.exit(1);
                }
        }
        public void setServeraddress(String addr){
                this.address = addr;
        }
        public String getServeraddess(){
                return address;
        }
        public void setServerport(int port){
                this.port = port;
        }
        public int getServerport(){
                return port;
        }
        public void setLogfilepath(String path){
                this.path = path;
```

```
        }
        public String getLogfilepath(){
                return path;
        }
}
```

5. SCREEN CHARTS

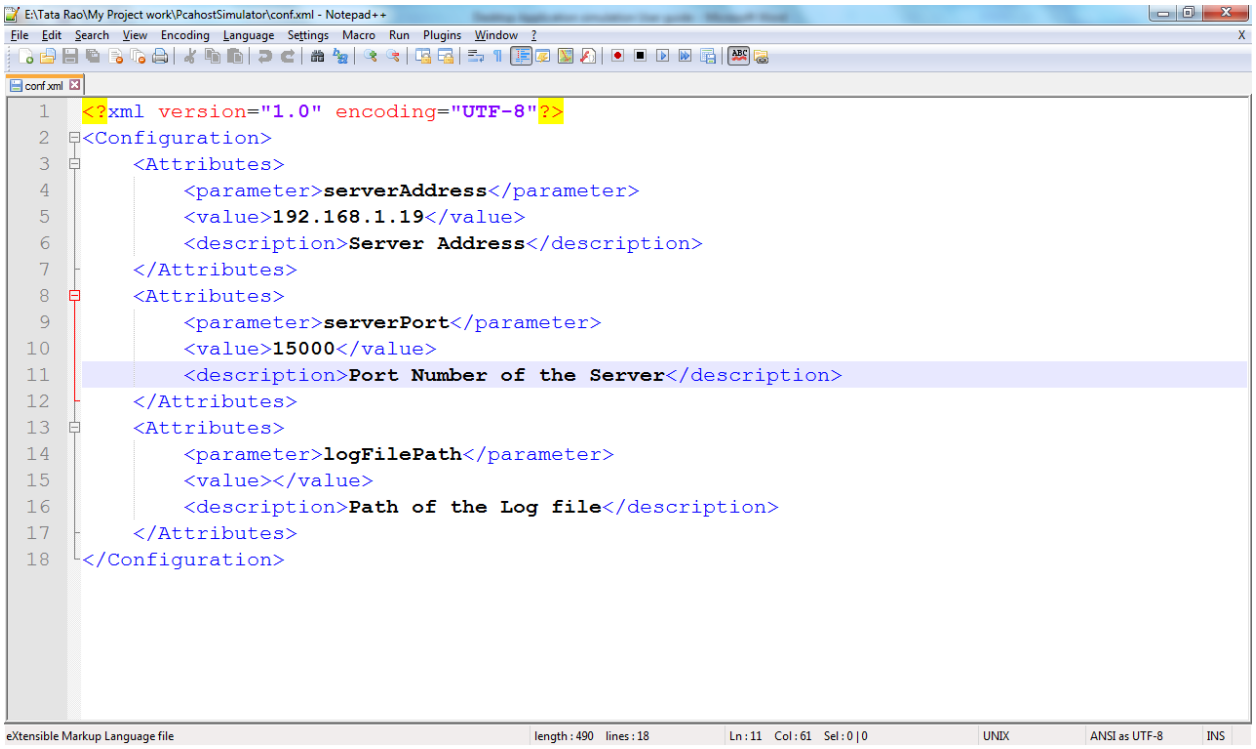
5.1 Startup

At initial stage, only CONNECT button was enabled, so that user can come to know he/she has to connect his device with server.



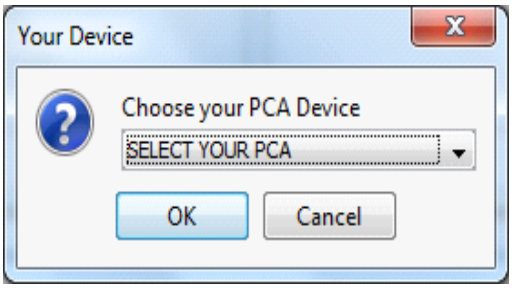
Once the user started the application, he/she allowed to connecting with the other PCA devices as his/her wish. Before the connection he/she should make sure that the configuration file must be provided with server address and port number and log file path (log file path can be empty also as you wish). Configuration file (***conf.xml***) located in the directory level where application installed.

Conf.xml

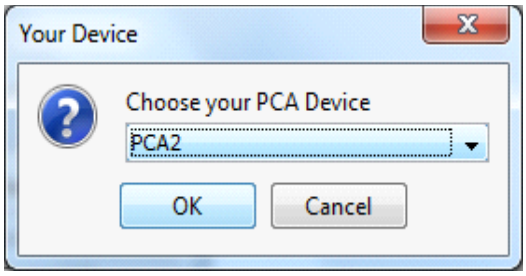


5.2 Connecting PCA

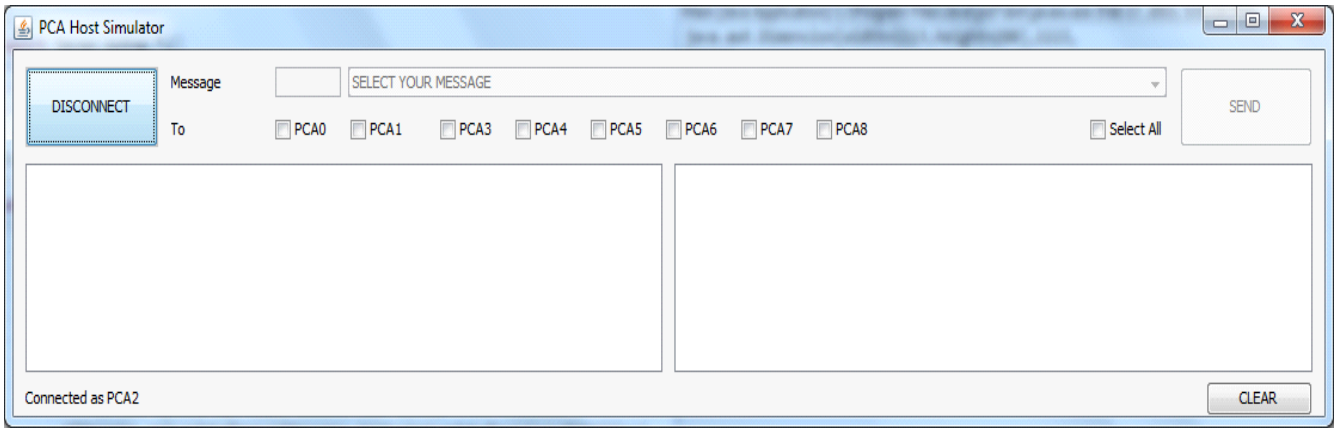
When user clicked on CONNECT button, he/she able to Select PCA device from a PCA list displayed by a InputDialogBox as shown in bellow.



Here, InputDialogBox provides two optional buttons (OK and Cancel). Click on Cancel closes the GUI window and click on OK allows the user to connect with server, if the user has been selected on of the PCA device from list. Suppose user clicks the OK button without selecting PCA, nothing can be happen and same dialog box shown again and again until either he/she select a PCA or clicks on Cancel.



Once PCA selected, and then clicked on Ok, user could be connected as selected PCA, the same status can be shown at left-bottom label and CONNECT button state change to DISCONNECT. Connection status is visible to user, until clicked on DISCONNECT state of button. See bellow image.



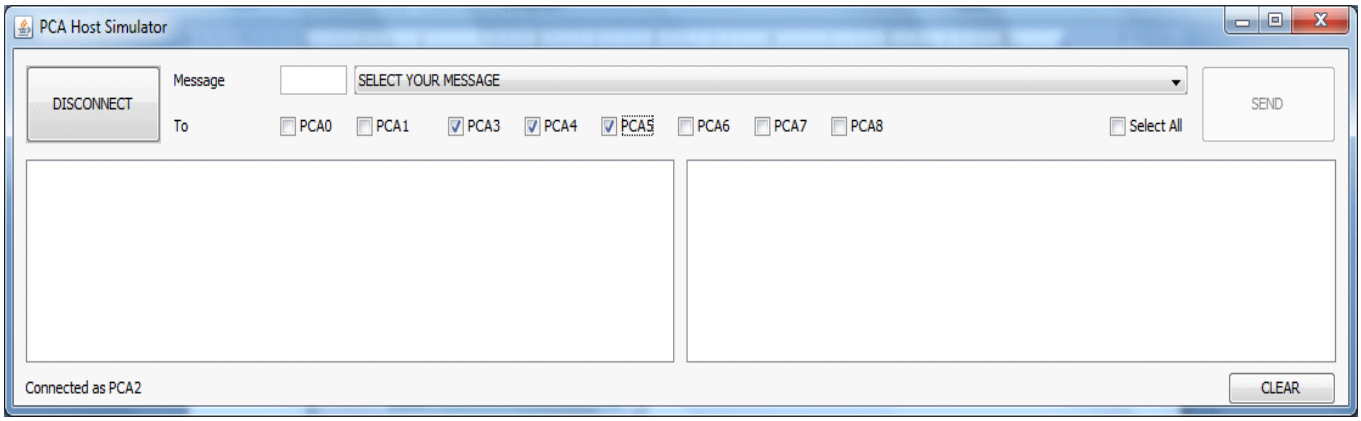
When user connected as one of the PCA, then checkboxes getting enabled as shown as above. Thereby user comes to know, he/she has to select the destination PCAs.

5.3 Destination PCAs selection

User should select at least one PCA checkbox as destination. “Select All” checkbox toggled for select all PCAs and deselect all PCAs.

Note: Until user select at least one PCA checkbox, Message components such as Message ID textfield, and Message combobox will be in disable state.

After selecting PCA(s), MessageID Textfield and Message combobox get enabled and allows the user to select MessageId and Message as shown bellow.



5.4 Selecting a Message

User can select message either by entering message id in text filed or by selecting a message from combobox. (Combo box populates the messages which are given in text file named Message.txt)

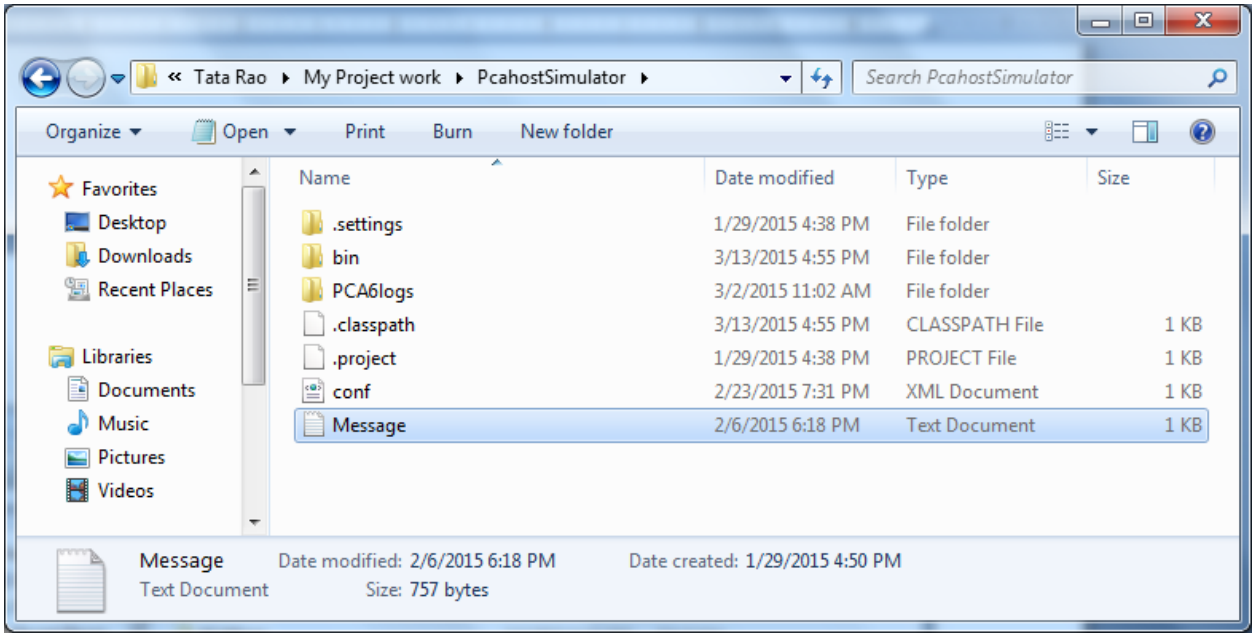
Message text file format
<msgid><space><message as ASCII>

Note: Messages which follows the above format can only accepted.

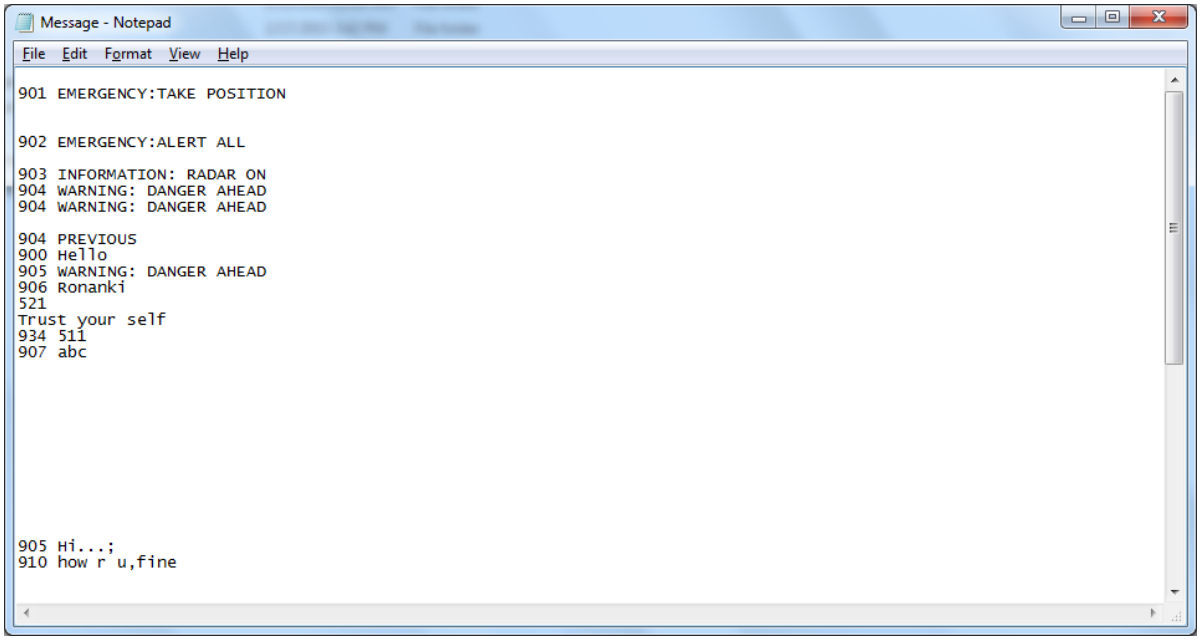
Example:

901 EMERGENCY:TAKE POSITION
902 EMERGENCY:ALERT ALL

Message.txt file located at application installed directory level. Represented in below figure.

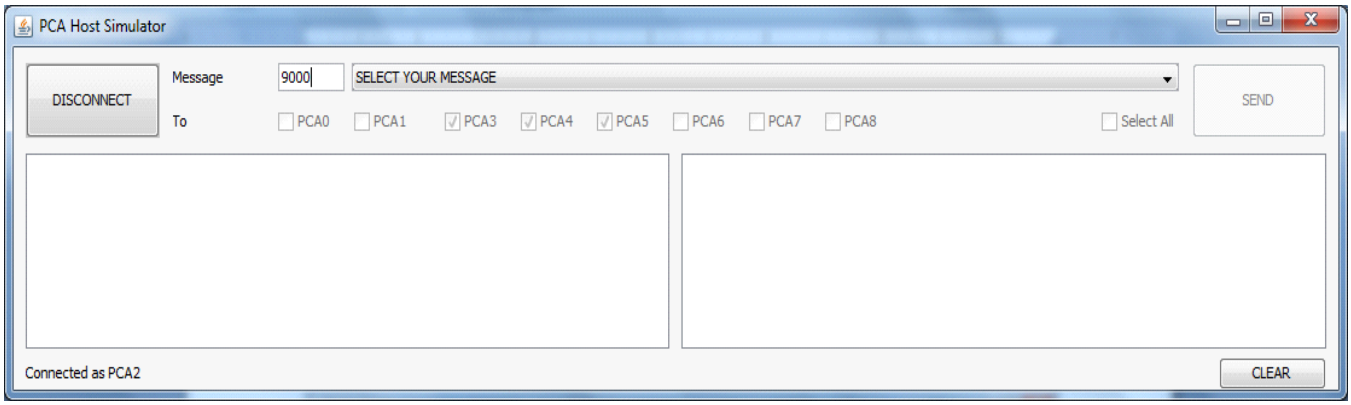


Message.txt



Selecting a message by entering message id:

User can able to enter message id in text field as shown below.

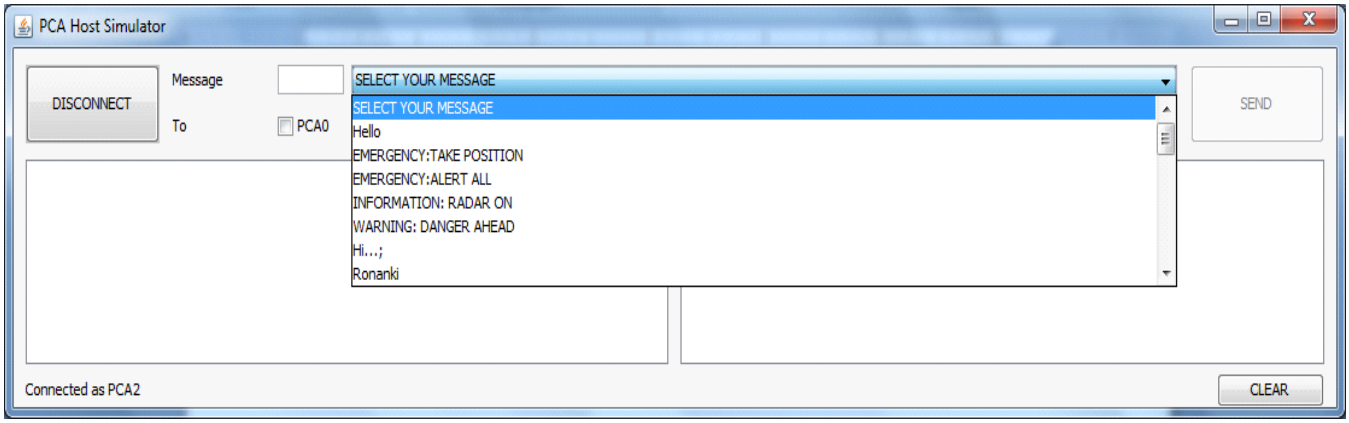


If the message id entered by the user is not exists in the text file, then an alert box will be displayed to alert the user as shown below. If the user clicks on OK, then the text field cleared and the focus of curser remains in text field to suggest the user for reentering of message id.

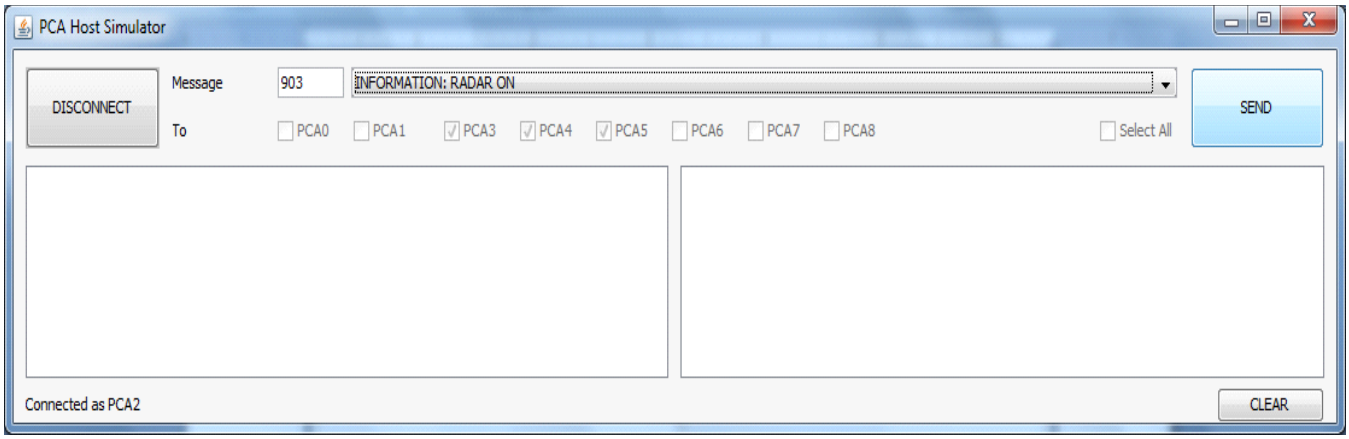
If the entered message id existed, then the corresponding message populated in combo box and SEND button will enable to allow sending of message.

Selecting a message from combo box:

User can able to select a message from combo box as shown below.



After selecting message from combo box, the corresponding message id placed in text field and SEND button will enable to allow sending of message as shown below.



Note: The SEND button disabled until you selected a message.

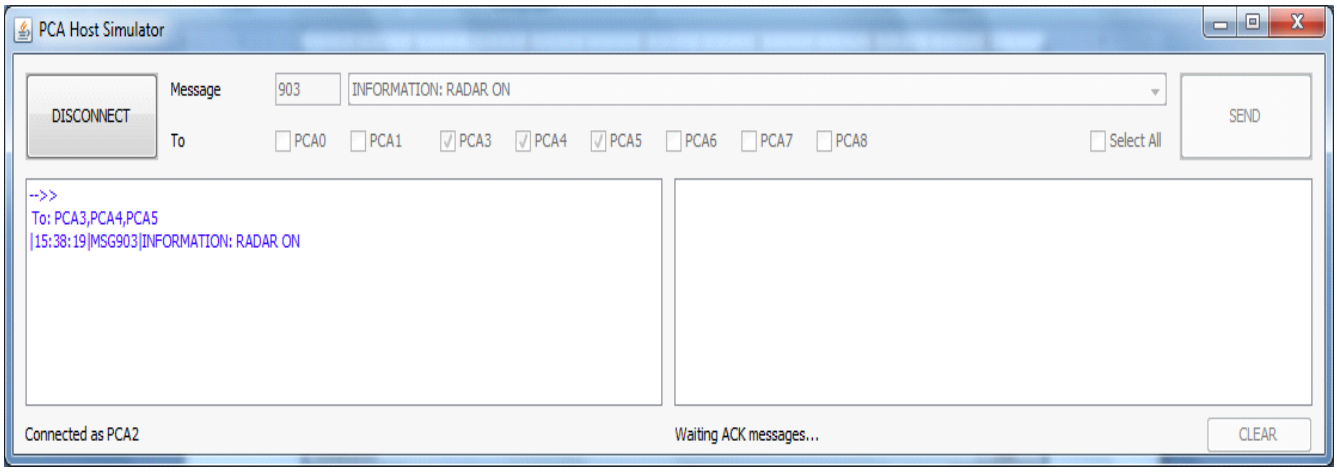
5.5 Sending a Message

After user selected a message SEND button enabled and able to send message to selected PCAs. Once clicked on SEND button message has been sent to all destination PCAs and SEND button get disabled until all acknowledgments received (this disable of SEND button controls the resending of message multiple times by mistakenly).

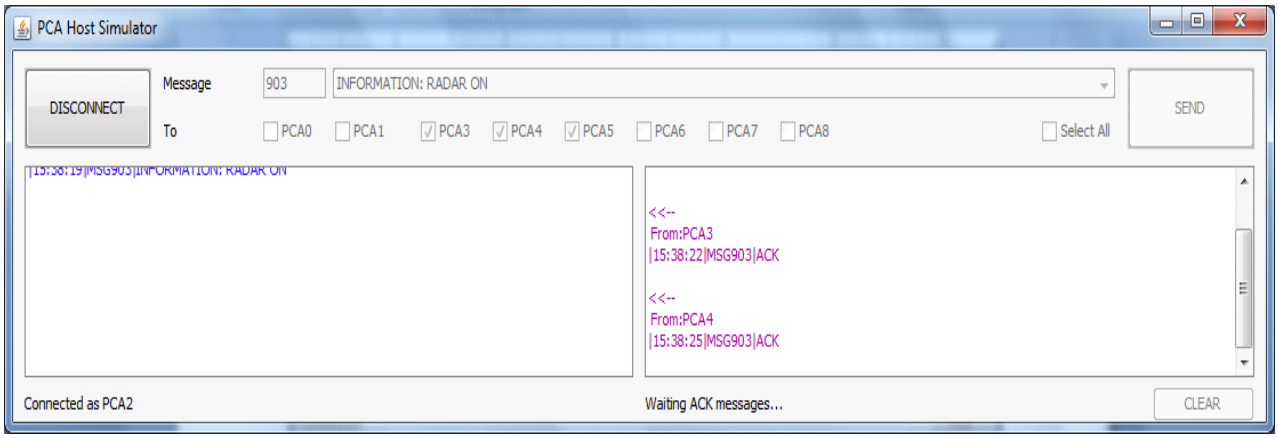
Note:
1. Check boxes and message components already disabled, which prevents alteration of destination PCAs selection and message content while sending.
2. Because of short span of time taken to send a message, changing of SEND button state to CANCEL and CANCEL to SEND state is not observed by us.

5.6 Displaying all outgoing messages and incoming messages

After message sent, **Message Panel-1** displays all messages outgoing from the connected PCA. These messages include direction indicator, destination, timestamp, msg id and initial part of message. See bellow snap.

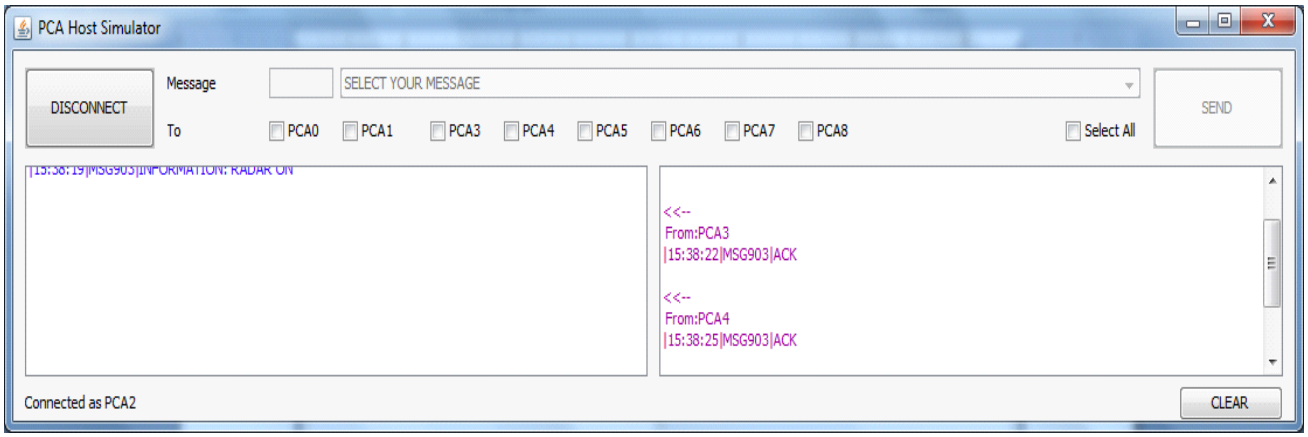


Message Panel 2 displays all incoming ACK messages to the connected PCA. Displayed messages include direction indicator, source, timestamp, msg id and initial part of message. See below snap.



Note: Label right-bottom displays waiting status for ACK messages until it gets all ACK messages from all selected PCA destinations, to which Source PCA sent message.

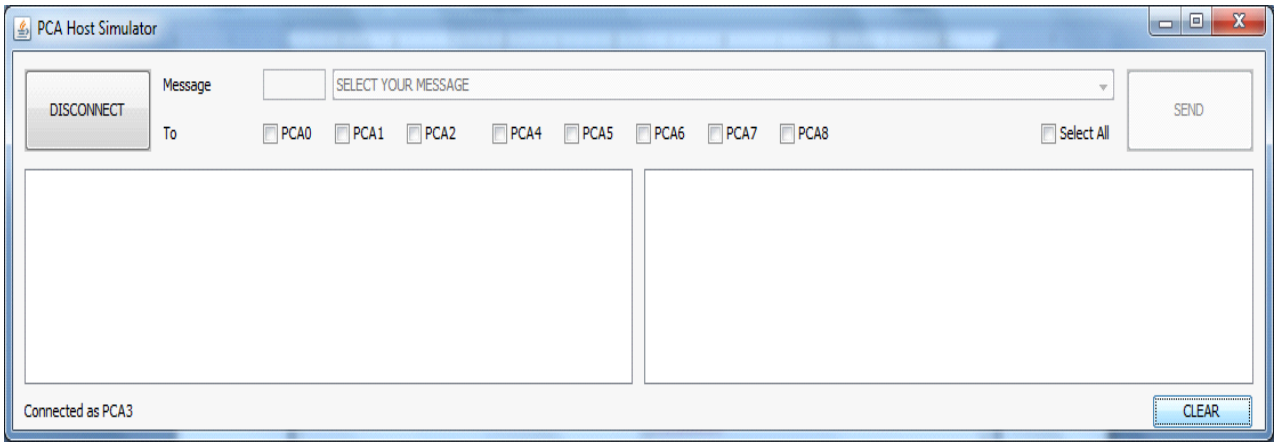
Once receiving all ACK messages get displayed, checkboxes automatically enabled. So that, again you can able to send messages.



5.7 Clearing message panels

Clear button disabled until outgoing and incoming messages displayed into message panels. Using clear button user can able to clear the message panels.

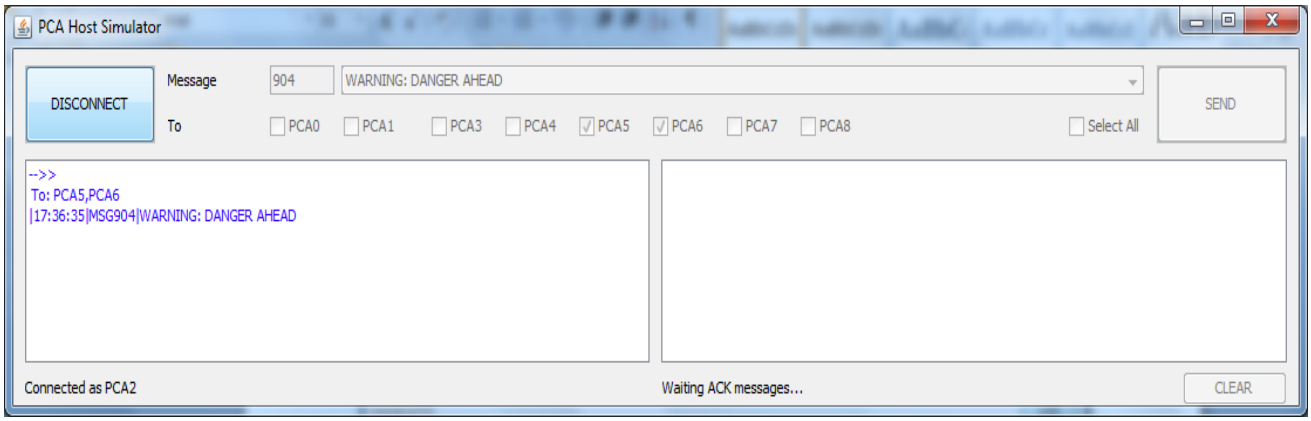
After clearing message panels, GUI looks as below:

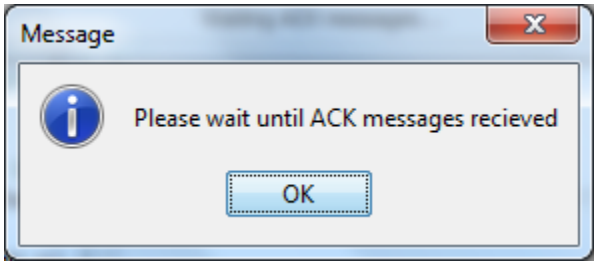


5.8 Disconnecting PCA

When user pressed DISCONNECT button, if there is no pending messages to send or receive, then TCP socket connection terminated and DISCONNECT changes to CONNECT state and clears both message panes.

When user pressed DISCONNECT button, if there is pending messages to send or receive, then an alert box will appear and informs about pending messages to receive and waits to disconnect until all pending messages has been received.

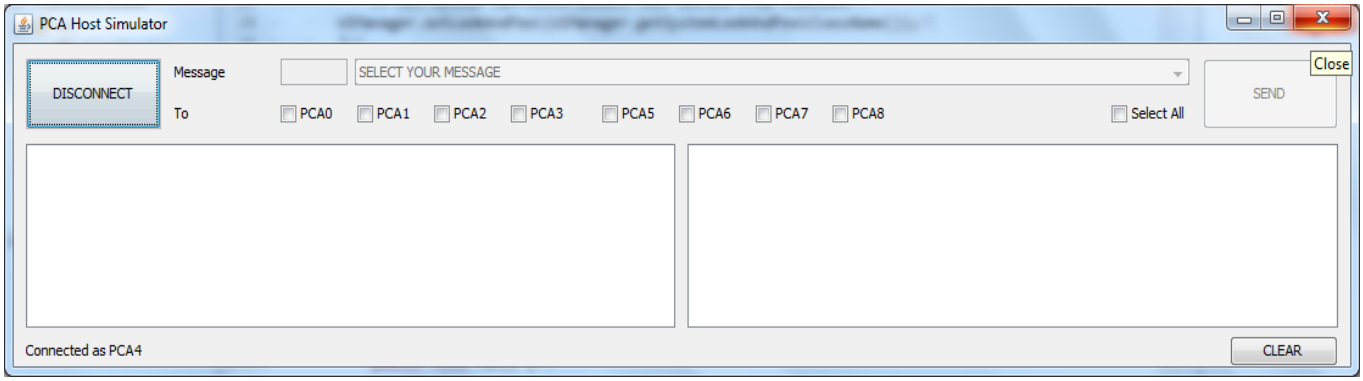


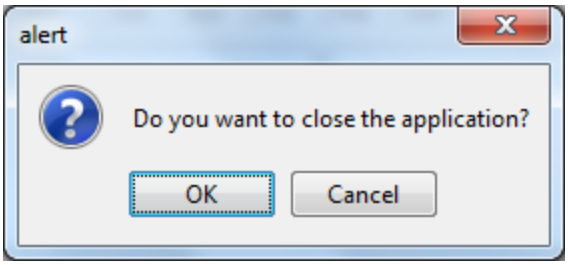


After all pending messages receive has been done, PCA disconnected automatically.

5.9 Closing GUI window

When user clicks on close button of window, an alert box displayed which takes conformity of user on closing window. See the below snap.





Note:

1. Closing window by user will disconnects PCA , if there is still a connection.
2. When the user tries to close the window, while acknowledgements are at pending then GUI will stay until all incoming messages (ACK) received then disconnected itself and GUI closed automatically.

5.10 Message Logging

All incoming and outgoing messages can be logged to text file. Each file should have connection timestamp as part of filename. A separate file is created every time a connect is performed. This log file is created in the path, which was provided into the conf.xml file.

Message log file format shown below:

```
-->>
 To: PCA7
|18:06:11|MSG315|EMERGENCY: TAKE POSITION
<<--
 From: PCA7
|18:07:03|MSG315|ACK
-->>
 To: PCA5, PCA3
|18:12:31|MSG335|EMERGENCY: ALERT ALL
<<--
From: PCA5
|18:12:55|MSG335|ACK
<<--
From: PCA3
|18:13:12|MSG335|ACK
```

Note:

1. Logs folder name itself represents user selected PCA logs.
2. If logs path not provided or not available in desktop terminal, PCA logs folder will be created in default path where application installed.

6. TESTING

PCA Communication – Test Cases				
TEST CASES REPORT				
Id	**TESTER**	**Test Case**	**Status**	**Comments**
1	santosh	Server Ip address and port number details are provided into conf.xml file. If any of these values are wrong, Connection was refused after few seconds	Cleared	
2	santosh	Log file path also provided into conf.xml file. If it is not provided, the logfile path created at relative path	Cleared	
3	santosh	When you open your application, you will see UI part of all components	Cleared	
4	santosh	If the user clicked on “window close” button, it will shows a dialog box that you really want to close the UI. If yes, it closes the window. if Cancel, it will continues	Cleared	
5	santosh	At start up,only CONNECT button was enabled, so user able to understand connect his device.	Cleared	
6	santosh	When User pressed on CONNECT button, you are able to Select PCA device from a PCA list displayed by a InputDialogBox	Cleared	
7	santosh	Dialog box display two buttons Ok and Cancel. If you click Cancel, the UI window will get closed. If you clicked on Ok button without selecting PCA from list, the same dialog box displayed continuously until you selected a PCA.	Cleared	
8	santosh	Checkbox labels updated based on the selection.	Cleared	
9	santosh	Once you selected your PCA, and then clicked on Ok, you are now connected to respected PCA, the same status can be shown at left-bottom label and CONNECT button state change to DISCONNECT. Connection status is visible to you, until you clicked on DISCONNECT state of button.	Cleared	

10	santosh	When your PCA get connected, TCP socket connection started and receiving thread started to receive response from other side.	Cleared	
11	santosh	When your PCA get connected, you will able to select destination PCAs (one or more) displayed as checkboxes. “Select All” checkbox toggled for select all PCAs and deselect all PCAs. Once you select all PCAs by SelectAll checkbox, then if you want to deselect any checkbox, all checkboxes get deselected and you have to select again as your wish.	Cleared	
12	santosh	Until you select atleast one PCA checkbox, Message components such as Message ID textfield, and Message combobox will be in disable state.	Cleared	
13	santosh	After selecting PCA(s), all checkboxes disabled,MessageID Textfield and Message combobox get enabled and allow you to select MessageId and Message.	Cleared	
14	santosh	All messages stored in a text file named as Message.txt and populated to the combobox when application starts. User shall edit the text file to add/remove/modify the messages.	Cleared	
15	santosh	User was able to type a number in MessageID Textfield corresponding to the message of his choice when you pressed enter the same selected in the combo box next to it.	Cleared	
16	santosh	If the entered number is not existed in your text file, it displays an alert box and clears both Message components to default state when we click OK button. And you now able to reselect your message.	Cleared	
17	santosh	User was able to select a message from combobox of his choice,then the corresponding message id displayed in textfield.	Cleared	
18	santosh	The SEND button disabled until you selected a message. After you selected a message SEND button enabled and you are able to send message to selected PCAs	Cleared	
19	santosh	SEND button toggled as send and cancel button. Starting state is Send and changes to CANCEL as soon send is started. Soon after send is completed, changes back to SEND. If SEND is pressed, it initiates sending packet over tcp socket.	Cleared	

20	santosh	When sending initiated, Message Id Textfield and combobox get disabled (Checkboxes already disabled). So that you can avoid the changing of message and changing of destinations while data is sending and receiving ACKs.	Cleared	
21	santosh	After SEND button clicked, Message Panel-1 displays all messages outgoing from the connected PCA. These messages displayed in required format (Mean to say,includes direction indicator, destination, timestamp , msg id and initial part of message).	Cleared	
22	santosh	Message Panel 2 displays all incoming ACK messages to the connected PCA. Displayed messages includes direction indicator, source, timestamp , msg id and initial part of message.	Cleared	
23	santosh	Label right-bottom displays waiting status for ACK messages until it gets all ACK messages from all selected PCA destinations, to which Source PCA sent message.	Cleared	
24	santosh	Both message panels were synchronized with each other for scroll and other operations.	Cleared	
25	santosh	Clear button disabled until sending and receiving messages displayed into message panels in one send.	Cleared	
26	santosh	Clear button clears both panels. (Just you get feeling of clear screen as in DOS cmd screen or UNIX).	Cleared	
27	santosh	Once receiving all ACK messages get displayed, checkboxes automatically enabled. So that again you can able to send messages.	Cleared	
28	santosh	When you pressed DISCONNECT button, if there is no pending messages to send or receive, then TCP socket connection terminated and DISCONNECT changes to CONNECT state.And clear both message panes. After connect, button changes to disconnect and vice versa.	Cleared	
29	santosh	New separate Log text file created for each connection	Cleared	
30	santosh	Each Log text file had connection timestamp as part of filename.	Cleared	
31	santosh	All incoming and outgoing messages logged to log text file	Cleared	

		in windows not in linux		
32	santosh	we should have 3 bytes of message Id .But if we assign more than 3 or less than 3 bytes to ID .it wl takes and send message to device	Cleared	
33	santosh	If we send message ID with more than 3 digits. Responce Acknolegment message will come back with 3 digit ID only	Cleared	
34	santosh	we want to make GUI look the same on screen resolutions in different operating systems	Cleared	