Apéndice C

Código fuente

Este apéndice contiene el código fuente completo de las clases de ejemplo actualizado a la última version antes de la creación de este documento.

C.1. mx.udlap.kjProtocol.AdminKMessage

.

```
                ...
public class AdminKMessage extends BasicPacket implements
   BasicKannelProtocolMessage {
                ...
        private byte[] message = null;
        private KInteger adm_command = null;
        private KString boxc_id = null;
                ...
        public byte[] getMessage() {

                if (this.message == null) {
                        this.message =
                           DataTypesTools.byteCat(super.length.getBytes(),
                            super.type.getBytes());
                        this.message =
                           DataTypesTools.byteCat(this.message,
                           this.adm_command.getBytes());
```

```
                this.message =
                   DataTypesTools.byteCat(this.message,
                   this.boxc_id.getBytes());
            }
            return this.message;
        }
            ...
        public void setMessage(byte[] data) throws
           PacketParseException {
            KInteger swap = null;
            int index = 0;
            ...
            // Parseamos longitud del paquete (primeros 4
               bytes)
            swap = new KInteger(data[0], data[1], data[2],
               data[3]);
            ...
            // Actualizamos el super atributo length
            super.length = swap;
            ...
            // Parseamos tipo del paqeute (siguientes 4
               bytes)
            swap = new KInteger(data[4], data[5], data[6],
               data[7]);
            ...
            // Actualizamos el super atributo type
            super.type = swap;
            ...
            // Indicamos indice donde se encuentra siguiente
                byte
            index = 8;

            // Parseamos siguiente atributo de este paquete
            this.adm_command =
               DataTypesTools.parseKIntFromByteArray(data,
               index);

            // Indicamos indice donde se encuentra siguiente
                byte
            index += 4;

            // Parseamos siguiente atributo de este paquete
            this.boxc_id =
               DataTypesTools.parseKStringFromByteArray(data,
                index);
```

```
        }
                ...
}
```

C.2. mx.udlap.kjGateway.SimpleJMSTransport

.

```
1  package mx.udlap.kjGateway;
2
3  import mx.udlap.kjProtocol.packets.SMSPacketMessage;
4  import mx.udlap.kjProtocol.packets.AckMessage;
5  import java.io.Serializable;
6  import java.util.Properties;
7  import javax.jms.Message;
8  import javax.jms.Topic;
9  import javax.jms.TopicConnection;
10 import javax.jms.MessageListener;
11 import javax.jms.TopicConnectionFactory;
12 import javax.jms.TopicSubscriber;
13 import javax.jms.TopicPublisher;
14 import javax.jms.TopicSession;
15 import javax.jms.Session;
16 import javax.jms.ObjectMessage;
17 import javax.naming.NamingException;
18 import javax.jms.JMSException;
19 import javax.naming.Context;
20 import javax.naming.InitialContext;
21 import java.util.LinkedList;
22 import javax.jms.*;
23
24
25 public class SimpleJMSTransport implements JMSTransport,
      MessageListener {
26
27  public static final String prop_ConnectionFactoryName =
       "jmsTransport.ConnectionFactoryName";
28  public static final String prop_InBoundTopicName =
       "jmsTransport.inBoundTopicName";
```

```
29  public static final String prop_OutBoundTopicName =
       "jmsTransport.outBoundTopicName";
30  public static final String prop_JMSTranslatorClass =
       "jmsTransport.JMSTranslatorClass";
31
32  public static final int NOTSTARTED = -1;
33  public static final int STARTED = 0;
34  public static final int JMSFAILED = 1;
35  public static final int JNDIFAILED = 2;
36  public static final int CLASSLOADING = 4;
37  public static final int OTHER = 256;
38
39
40
41  private KjWritingThread writer = null;
42  private Properties props = null;
43  private JMSTranslator jmsTranslator = null;
44  private Serializable inObject = null;
45  private SMSPacketMessage outSMS = null;
46  private int status = NOTSTARTED;
47  private SimpleQueueReceiverThread sqrThread = null;
48
49  /*//////////////////////////////////////////////////
50  // JMS Related attributes
51  private Context jndiContext = null;
52  private TopicConnectionFactory topicConnectionFactory = null;
53  private TopicConnection topicConnection = null;
54  private TopicSession topicSession = null;
55  private Topic inBoundTopic = null;
56  private Topic outBoundTopic = null;
57  private TopicPublisher outBoundPublisher = null;
58  private TopicSubscriber inBoundSubscriber = null;
59  private ObjectMessage outOMessage = null;
60  private ObjectMessage inOMessage = null;
61  //////////////////////////////////////////////////*/
62
63  //////////////////////////////////////////////////
64  // JMS Related attributes
65  private Context jndiContext = null;
66  private QueueConnectionFactory topicConnectionFactory = null;
67  private QueueConnection topicConnection = null;
68  private QueueSession topicSession = null;
69  private Queue inBoundTopic = null;
70  private Queue outBoundTopic = null;
71  private QueueSender outBoundPublisher = null;
```

```
72  private QueueReceiver inBoundSubscriber = null;
73  private ObjectMessage outOMessage = null;
74  private ObjectMessage inOMessage = null;
75  ////////////////////////////////////////////////////
76
77
78  public void start(Properties props) {
79   try {
80    this.props = props;
81
82    // get translator
83    String swap =
        this.props.getProperty(this.prop_JMSTranslatorClass);
84    this.jmsTranslator =
        (JMSTranslator)Class.forName(swap).newInstance();
85
86    // Set up all the JMS connection stuff
87    if (this.props != null){
88     jndiContext = new InitialContext(this.props);
89    }else{
90     jndiContext = new InitialContext();
91    }
92
93    topicConnectionFactory = (QueueConnectionFactory)jndiContext
94       .lookup(this.props.getProperty(prop_ConnectionFactoryName));
95
96    //inBoundTopic =
        (Topic)jndiContext.lookup(this.props.getProperty(prop_InBoundTopicNam
97    //outBoundTopic =
        (Topic)jndiContext.lookup(this.props.getProperty(prop_OutBoundTopicNa
98
99    inBoundTopic =
        (Queue)jndiContext.lookup(this.props.getProperty(prop_InBoundTopicNam
100   outBoundTopic =
        (Queue)jndiContext.lookup(this.props.getProperty(prop_OutBoundTopicNa
101
102   //topicConnection =
        topicConnectionFactory.createTopicConnection();
103   //topicSession = topicConnection.createTopicSession(false,
        Session.AUTO_ACKNOWLEDGE);
104
105   topicConnection =
        topicConnectionFactory.createQueueConnection();
106   topicSession = topicConnection.createQueueSession(true,
        Session.AUTO_ACKNOWLEDGE);
```

```
107
108
109
110     //outBoundPublisher =
            topicSession.createPublisher(outBoundTopic);
111     //inBoundSubscriber =
            topicSession.createSubscriber(inBoundTopic);
112
113     outBoundPublisher = topicSession.createSender(outBoundTopic);
114     inBoundSubscriber =
            topicSession.createReceiver(inBoundTopic);
115
116     inBoundSubscriber.setMessageListener(this);
117     //sqrThread = new SimpleQueueReceiverThread(this,
            inBoundSubscriber);
118     //sqrThread.start();
119     topicConnection.start();
120
121     status = STARTED;
122    }catch(JMSException e ){
123     status = JMSFAILED;
124    }catch(NamingException e ){
125     status = JNDIFAILED;
126    }catch(ClassNotFoundException e ){
127     status = CLASSLOADING;
128    }catch(InstantiationException e ){
129     status = CLASSLOADING;
130    }catch(IllegalAccessException e ){
131     status  = CLASSLOADING;
132    }catch(Throwable e){
133     status = OTHER;
134    }
135   }
136
137   public int getStatus() {
138    return status;
139   }
140
141
142   public void empty(Object ack){
143    System.out.println(ack);
144   }
145
146   public void onMessage(Message parMessage){
147    inOMessage = null;
```

```
148   outSMS = null;
149   try{
150    if(parMessage instanceof ObjectMessage){
151     inOMessage = (ObjectMessage) parMessage;
152     // Mensaje listo para traducir y enviar a Kannel
153     outSMS =
          this.jmsTranslator.objectToKannel((Object)inOMessage.getObject());
154     //System.out.println("Sending: " + outSMS.hexDump());
155     writer.send(outSMS);
156     topicSession.commit();
157    }else{
158     System.out.println("Otro␣tipo␣de␣mensaje␣recibido:");
159     System.out.println(parMessage.getClass().getName());
160    }
161
162   }/*catch( JMSException e){
163    System.out.println("JMSException");
164   }*/catch( Throwable tjms){
165    System.out.println("Otra␣excepcion");
166   }
167
168  }
169
170
171
172  public void addKjWritingThread(KjWritingThread writer){
173   this.writer = writer;
174  }
175
176  public void gotMOMessage(SMSPacketMessage smsMessage){
177   // Read from Kannel
178   // Write this message to JMS topic
179   outOMessage = null;
180   inObject = this.jmsTranslator.kannelToObject(smsMessage);
181   try{
182
183    //System.out.println("Recibido:\n" + smsMessage.hexDump());
184    outOMessage = topicSession.createObjectMessage();
185    outOMessage.setObject(inObject);
186    //outBoundPublisher.publish(outOMessage);
187    outBoundPublisher.send(outOMessage);
188    topicSession.commit();
189    // Enviar ACK
190
191
```

```
192
193     //System.out.println("writing ack");
194     writer.rawWrite((new AckMessage(AckMessage.ACK_STAT_SUCCESS,
           smsMessage)).getMessage());
195     //System.out.println("ack written");
196    }catch(Exception e){
197     System.out.println("Message␣publishing␣failed:␣" + e);
198    }
199
200   }
201
202   public void gotMTMessage(Object obj){
203    // Read from a topic
204    // Write this message to kannel link
205
206   }
207 }
```

C.3. mx.udlap.kjGateway.SimpleMessage

.

```
1  package mx.udlap.kjGateway;
2
3  import mx.udlap.kjProtocol.packets.SMSPacketMessage;
4  import java.io.Serializable;
5
6  public class SimpleMessage implements Serializable{
7
8   private String sender = "";
9   private String receiver = "";
10  private String udhData = "";
11  private String msgData = "";
12
13  public SimpleMessage(){}
14
15  public SimpleMessage(String sender, String receiver, String
       udhData, String msgData){
16   setSender(sender);
17   setReceiver(receiver);
```

```
18    setUdhData(udhData);
19    setMsgData(msgData);
20   }
21
22   public void setSender(String sender) {
23    this.sender = sender;
24   }
25   public void setReceiver(String receiver) {
26    this.receiver = receiver;
27   }
28   public void setUdhData(String udhData) {
29    this.udhData = udhData;
30   }
31   public void setMsgData(String msgData) {
32    this.msgData = msgData;
33   }
34   public String getSender() {
35    return sender;
36   }
37   public String getReceiver() {
38    return receiver;
39   }
40   public String getUdhData() {
41    return udhData;
42   }
43   public String getMsgData() {
44    return msgData;
45   }
46
47
48  }
```

C.4. mx.udlap.kjGateway.SimpleService

.

```
1  package mx.udlap.kjGateway;
2
3  import mx.udlap.kjProtocol.packets.SMSPacketMessage;
4  import java.io.Serializable;
```

```
5  import java.io.InputStreamReader;
6  import java.io.FileInputStream;
7  import java.io.File;
8  import java.io.IOException;
9  import java.util.Properties;
10 import javax.jms.Message;
11 import javax.jms.Topic;
12 import javax.jms.TopicConnection;
13 import javax.jms.MessageListener;
14 import javax.jms.TopicConnectionFactory;
15 import javax.jms.TopicSubscriber;
16 import javax.jms.TopicPublisher;
17 import javax.jms.TopicSession;
18 import javax.jms.Session;
19 import javax.jms.ObjectMessage;
20 import javax.naming.NamingException;
21 import javax.jms.JMSException;
22 import javax.naming.Context;
23 import javax.naming.InitialContext;
24 import javax.jms.*;
25
26 public class SimpleService implements MessageListener {
27
28  public static final String prop_ContentFile =
       "simpleService.contentFile";
29
30  private String defaultMessage = "Contenido␣no␣encontrado.";
31  private Properties properties = null;
32  private Properties contenido = null;
33  private SimpleQueueReceiverThread sqrThread = null;
34  private long stress = 0;
35  private long stressTime = 0;
36
37  /*//////////////////////////////////////////////////
38  // JMS Related attributes
39  private Context jndiContext = null;
40  private TopicConnectionFactory queueConnectionFactory = null;
41  private TopicConnection queueConnection = null;
42  private TopicSession queueSession = null;
43  private Topic inBoundTopic = null;
44  private Topic outBoundTopic = null;
45  private TopicPublisher outBoundPublisher = null;
46  private TopicSubscriber inBoundSubscriber = null;
47  private ObjectMessage outOMessage = null;
48  private ObjectMessage inOMessage = null;
```

```
49  ////////////////////////////////////////////////////*/
50
51  ////////////////////////////////////////////////////
52  // JMS Related attributes
53  private Context jndiContext = null;
54  private QueueConnectionFactory queueConnectionFactory = null;
55  private QueueConnection queueConnection = null;
56  private QueueSession queueSession = null;
57  private Queue inBoundTopic = null;
58  private Queue outBoundTopic = null;
59  private QueueSender outBoundPublisher = null;
60  private QueueReceiver inBoundSubscriber = null;
61  private ObjectMessage outOMessage = null;
62  private ObjectMessage inOMessage = null;
63  ////////////////////////////////////////////////////
64
65
66  public SimpleService(Properties props) throws NamingException,
67        JMSException {
68   this.properties = props;
69
70   try {
71   this.contenido = new Properties();
72   this.contenido.load(new FileInputStream(
73      new File(this.properties.getProperty(prop_ContentFile))));
74
75   this.contenido.list(System.out);
76
77   }catch(IOException e){
78    System.out.println("Error␣cargando␣contenido...");
79    this.contenido = null;
80   }
81
82
83   // Set up all the JMS connection stuff
84   if (this.properties != null){
85    jndiContext = new InitialContext(this.properties);
86   }else{
87    jndiContext = new InitialContext();
88   }
89
90   queueConnectionFactory = (QueueConnectionFactory)jndiContext
91      .lookup(this.properties.getProperty(SimpleJMSTransport.prop_Connection
92
```

```
93    inBoundTopic =
         (Queue)jndiContext.lookup(this.properties.getProperty(SimpleJMSTranspo
94    outBoundTopic =
         (Queue)jndiContext.lookup(this.properties.getProperty(SimpleJMSTranspo
95
96    queueConnection =
         queueConnectionFactory.createQueueConnection();
97    queueSession = queueConnection.createQueueSession(true,
         Session.AUTO_ACKNOWLEDGE);
98
99    outBoundPublisher = queueSession.createSender(outBoundTopic);
100   inBoundSubscriber = queueSession.createReceiver(inBoundTopic);
101
102   inBoundSubscriber.setMessageListener(this);
103   //sqrThread = new SimpleQueueReceiverThread(this,
         inBoundSubscriber);
104   //sqrThread.start();
105   queueConnection.start();
106
107  }
108
109
110  public void start(){
111
112
113
114   System.out.println("Para␣terminar␣presiona␣*␣␣ENTER");
115   InputStreamReader inputStreamReader = new
         InputStreamReader(System.in);
116   char last = '\0';
117   while (!(last == '*')){
118    try{
119     last = (char) inputStreamReader.read();
120    }catch(IOException e){
121     System.out.println("I/O␣Error");
122    }
123
124   }
125
126  }
127
128  public void onMessage(Message parMessage){
129   inOMessage = null;
130   outOMessage = null;
131   SimpleMessage mesg = null;
```

```
132   String swap = null;
133   try{
134    if(parMessage instanceof ObjectMessage){
135     inOMessage = (ObjectMessage) parMessage;
136     // Mensaje listo para traducir y enviar a Kannel
137     mesg = (SimpleMessage)inOMessage.getObject();
138
139     swap = mesg.getMsgData().trim().toUpperCase();
140     if (this.contenido != null){
141      swap = this.contenido.getProperty(swap);
142
143      if(swap != null){
144       System.out.println(swap);
145       outOMessage = queueSession.createObjectMessage();
146       outOMessage.setObject(new
             SimpleMessage(mesg.getReceiver(),
147            mesg.getSender(),
148            "",
149            swap));
150       //System.out.println("llego1");
151       outBoundPublisher.send(outOMessage);
152       queueSession.commit();
153       //System.out.println("llego");
154      }
155     }
156
157    }else{
158     System.out.println("Otro tipo de mensaje recibido:");
159     System.out.println(parMessage.getClass().getName());
160    }
161
162   }catch( JMSException e){
163    System.out.println("JMSException");
164   }catch( Throwable tjms){
165    System.out.println("Otra excepcion: " + tjms);
166   }
167
168  }
169
170  public static void main(String arg[]) throws Exception {
171   System.out.println("Starting SimpleService");
172   Properties props = new Properties();
173   props.load(new FileInputStream(
174     new File(
175     new String(System.getProperty("user.dir") +
```

```
176       System.getProperty("file.separator") +
177       "kjGateway.cfg"))));
178
179     SimpleService serv = new SimpleService(props);
180     serv.start();
181   }
182
183 }
```