

# Features “VoIP UserAgent”

Daniele Pellegrini 285240

Guido Soncini 285140

Mattia Ricci 285237

UserAgent	
<code>static InetAddress getAddress()</code>	Return the loopback address
<code>static DatagramSocket getSocketOutgoing()</code>	Get the UDP Socket used to send packets to the mjUA on the port 5080. OUTPUT: <ul style="list-style-type: none"><li>• DatagramSocket used to send UDP packets</li></ul>
<code>static DatagramSocket getSocketIncoming()</code>	Get the UDP Socket used to receive packets from the mjUA on the port 5070. OUTPUT: <ul style="list-style-type: none"><li>• DatagramSocket used to receive UDP packets</li></ul>
<code>static void send(byte[] request)</code>	Send a request in byte to the server mjUA INPUT: <ul style="list-style-type: none"><li>• request: the Request to send, in byte</li></ul>
<code>static DatagramPacket listen() throws SocketTimeoutException</code>	Listen and return for a new DatagramPacket on the socketIncoming DatagramSocket. OUTPUT: <ul style="list-style-type: none"><li>• DatagramSocket received</li></ul> EXCEPTION: <ul style="list-style-type: none"><li>• SocketTimeoutException</li><li>• IOException</li></ul>
<code>@Override void run()</code>	Run a new UserAgent inside a thread, send a Request and read the Response from the mjUA
Request	
<code>static String generateCallID()</code>	Generate and return a random CallID, a string of int

	values of 12 characters length.
<code>static String generateSenderTag()</code>	Generate and return a random Sender Tag,a string of int values of 12 characters length.
<code>static String generateBranch()</code>	Generate a pseudo-random branch of 13 characters length. Every branch must have as the first characters the sequence <i>z9hG4bK</i> , also known as magic number
<code>static void setSenderName(String newValue)</code>	Set the new UserAgent name
<code>static byte[] getInvite()</code>	Get the Invite Request, in byte
<code>static byte[] getAck()</code>	Get the Ack Request, in byte
<code>static byte[] getBye()</code>	Get the Bye Request, in byte
<b>Response</b>	
<code>static void setResponsePacket(DatagramPacket packet)</code>	Set the new Response Packet and the new message received.
<code>static void showMessage()</code>	Show the UserAgent the message received from the mjUA and a few lines of description.
<b>Session</b>	
<code>static void setActive(boolean value)</code>	Set the status of the current Session. INPUT: <ul style="list-style-type: none"> <li>• TRUE the Session is active</li> <li>• FALSE the Session is not active</li> </ul>
<code>static boolean isActive()</code>	Get the status of the current Session
<code>static void addRequest(byte[] newRequest)</code>	Add a new Request (in byte) to the Request list.
<code>static void addResponse(byte[] newRequest)</code>	Add a new Response (in byte) to the Response list.
<code>static void addPacket(DatagramPacket newPacket)</code>	Add a new Packet UDP, sent or received, to the Packets list.
<code>static void clear()</code>	Clear all the lists in the Session.
<code>static void saveRequestFile(String request,</code>	Save the Request sent on a file in

<code>String name)</code>	resources/requests INPUT: <ul style="list-style-type: none"><li>• request: the string to save on a file</li><li>• name: name of the file</li></ul>
<code>static String logMessage()</code>	Return a string containing every packet's data in the Packets list. OUTPUT: <ul style="list-style-type: none"><li>• the Session's logs message</li></ul>
<code>static void save()</code>	Save the Session's logs message in a external file named logs.txt in resources/requests
<b>RTP Packet</b>	
<code>byte[] getHeader()</code>	Get the header of the RTP Packet.
<code>void incrementSequence()</code>	Increment the Sequence value by one unit.
<code>void incrementTimestamp()</code>	Increment the Timestamp value by 160 unit.
<b>Output Audio</b>	
<code>static void setSendingAudio(boolean value)</code>	Set the status of the SendingValue attribute:: INPUT: <ul style="list-style-type: none"><li>• TRUE if the UserAgent is sending audio</li><li>• FALSE if the UserAgent is not sending audio</li></ul>
<code>static boolean isSendingAudio()</code>	Get the status of the SendingAudio attribute
<code>static void sendAudio(byte[] toSend)</code>	Send a RTP Packet (in byte) through Socket to the mjUA. INPUT: <ul style="list-style-type: none"><li>• toSend the RTP Packet, in byte.</li></ul>
<b>AudioThread</b>	
<code>static DatagramSocket getSocketIncoming()</code>	Get the UDP Socket used to receive packets from the mjUA on the port 4070. OUTPUT: <ul style="list-style-type: none"><li>• DatagramSocket used to receive RTP packets</li></ul>

<pre>@Override void run()</pre>	<p>Start a thread that in a loop receives the RTP packets sent by the mjUA (which simple contains silence), edit some of the 160 bytes of the payload with random values and send it back to the mjUA, which receives and play it. The sound will result in a spoiled noise.</p>
<b>AudioSinusoidalThread</b>	
<pre>static double getFrequency()</pre>	<p>Return the value of the Frequency</p>
<pre>static double getAmplitude()</pre>	<p>Return the value of the Amplitude</p>
<pre>static double getTimeIncrementation()</pre>	<p>Return the value of the Time Incrementation</p>
<pre>static void setFrequency(double newValue)</pre>	<p>Set the new value of Frequency attribute INPUT:</p> <ul style="list-style-type: none"> <li>• newValue of Frequency (in double)</li> </ul>
<pre>static void setAmplitude(double newValue)</pre>	<p>Set the new value of Amplitude attribute INPUT:</p> <ul style="list-style-type: none"> <li>• newValue of Amplitude (in double)</li> </ul>
<pre>static void setTimeIncrementation(double newValue)</pre>	<p>Set the new value of Time Incrementation attribute INPUT:</p> <ul style="list-style-type: none"> <li>• newValue of Time Incrementation (in double)</li> </ul>
<pre>@Override void run()</pre>	<p>Start a thread that create a new RTP Header and in a loop calculate a mathematical sine wave of given frequency and amplitude, as a function of x. The wave increments x by a given value. The sinusoid calculated is then compressed using the G711.linear2ulaw of sip.jar library, and fills the entire bytes of the RTP body. At the end of every loop, send the RTP Packet, and increments the Sequence and the Timestamp values. A RTP Packet is sent every 20ms.</p>
<b>AudioFileThread</b>	
<pre>@Override void run()</pre>	<p>Start a thread that create a new RTP Header and take a .wav file in the resources/audio folder to send it to the mjUA. It calculates the number of packets to</p>

be created fitting the entire audio file in different RTP's body of 160 bytes length, then start creating a loop that insert the `AudioInputStream` in every RTP body, and send the packet to the `mjUA`. Once a packet is sent increments the Sequence and the Timestamp value and send the next one after 20ms.