CureSPIT Project

02/13/2011
Yifan Zhang

1. Overview

1.1 Overview of CureSPIT

CureSPIT aims to help a callee determine whether to accept or not, even in the following 3 cases:
1. The Caller ID is not found in the accepting or denying list at the callee.
2. No Caller ID is set.
3. The Caller ID is not authenticated.

This mechanism of CureSPIT allows a callee to correlate with the previous communication with callers such as an email message or web browsing by checking additional information the callers provide in a SIP INVITE message. This additional information is usually cross-media relations which are provided during the previous communication. The callee who is using services of CureSPIT is provided an account to access a database which stores the cross-media relations which are added during previous cross-media communication with potential callers. The callee can query the database about the additional information to figure out who the caller could be and decide whether to accept the call.

1.2 Overview of my work

Table 1 shows my work, which implemented parts of the mechanism of CureSPIT. The programs include SIP Communicator Extension for CureSPIT, CureSPIT Contact Information Extractor (Firefox Add-on) and CureSPIT IMAPClient.

Table 1: **Progress and work status**

Entity	Program	Functions	Language	Project status
SIP client	SIP Communicator extension (8118) [1] for CureSPIT	Support a new SIP header[2] in INVITE method to convey a Message-ID[3]	Java	Done (using Sender-Ref header[2])
		Show the related information of the Message-ID in the window of		Done. (Modified the value of Message-ID and To header[4] address as

		incoming call alert (This needs to communicate with the database)		the hyperlink to the database)
		Implement the integration between Thunderbird and SIP Communicator		Partly done by implementing the function to make a SIP call with Message-ID using command line interface (the remaining part is to implement a Thunderbird Add-on. If an email message contains a SIP URI, the Add-on can use the command line to make a SIP call with Messaeg-ID using SIP Communicator)
		Deliver the related functions of the Message-ID as a separate module		Not done (because of lack of time)
Web browser	Firefox Add-on [5]	Extract contact information from the HTTP response when getting an OK message	Javascript, XUL	Done
		Insert the contact information to the database(This needs to communicate with the database)	Javascripte	Done
IMAP client	A new program using	Filter new received messages with	Java	Done

	JAVAMAIL API [9]	vCard[10] which also have been responded back, parse the vCard and		
		Post contact information into the database.(This needs to communicate with the database)	Java	Done

2. SIP Communicator extension for CureSPIT

2.1 Specification

This is an extension of SIP Communicator (8118) [11] project.

This CureSPIT extension supports two types of cross-media relations, email-to-call and web-to-call.
To support a cross-media relation, email-to-call, the extension allows a user to input a weak secret of an email message as the sender reference for a new call. The weak secret of an email message should be the value of the Message-ID of the previous email message that the user has received from the
callee beforehand. When the weak secret is input, the INVITE method of a new call includes the value of the weak secret in a new SIP header, Sender-Ref.

To support a web-to-call relation, the SIP Communicator extension for CureSPIT allows a user to input a weak secret of a web transaction as the destination subaddress[12] for a new call. Unlike an email message, a web transaction does not have a transaction ID in HTTP. Thus, a user (caller) should obtain the destination (callee's) subaddress in the previous web transaction. Since the weak secret of a web transaction is input as part of the destination address, the INVITE method of a new call includes the subaddress in the To header as part of the destination address. Thus, no SIP header extension is required to support a web-to-call relation.

2.2 Description

Name: SIP Communicator extension for CureSPIT

Platforms: The code runs on, Linux, Windows NT/2000/XP, Windows 7 with JDK 1.6.

Version: SIP Communicator extension for CureSPIT-1.0

Usage:
In order to run the extension, please install the "Apache Ant" first.

In the command line interface, use command "ant run" to run SIP Communicator. (use command "ant run -Dargs="–h"" to see more options to run the application.)

After the SIP Communicator is launched, first you should login with an SIP account. Then you can enter URI in the first textfield and Message-ID in the second textfield to make a SIP call with mid.

When a callee receive a SIP call with Message-ID and To header address (with subaddress), the Message-ID and the To header address should be the linked to the database. This requires the configuration of user properties.

They are set in the file “SIP Communitator/sip-communicator.properties” in the User Profile path of the OS.

For example, in Windows 7, the file is in the folder “C:\Users\[Username]\AppData\Roaming\SIP Communicator”.

If the linkpath of the Message-ID is “https://almond.cs.columbia.edu/phprestsql/index.php” and the SenderRef property is “email_secret”, we can add the property “net.java.sip.communicator.impl.gui.main.call.ReceivedCallDialog.linkPath=https\://almond.cs.columbia.edu/phprestsql/index.php” and “net.java.sip.communicator.impl.gui.main.call.ReceivedCallDialog.linkPath.SenderRef=email_secret” to the file “C:\Users\[Username]\AppData\Roaming\SIPCommunicator\sip-communicator.properties”

See the modification of file “sip-communicator.properties” in Appendix A.

Description:

One kind of the additional information is Message-ID, which is the ID of an email message. The destination address of the email message with a Message-ID and whether it's accepted are stored in the database.

To allow the callers to provide additional information (which is Message-ID in this extension) when making a call, a GUI entry is added for the caller to enter the Message-ID in the GUI interface and an additional parameter is added in the command line interface. Then, a new SIP header (Sender-Ref) in the INVITE method is added to convey the information.

To allow that the callee to see the Message-ID, the value of Message-ID is shown in the window of the incoming call alert.

To show that the callee can query the database using the Message-ID, the value of Message-ID is modified as a hyperlink to the record of the email-secret (using Message-ID to identify the record) in the database.

Also, the callee can use another additional information the callers may provide like subaddress[12]. The subaddress is used to identify the category of the callers. It is included in the To header address, using a delimiter to separate it with the original To header address. For example, Alice's address is "Alice@gmail.com". She can add the subaddress "student" to the address and post it to her students. With delimiter[12] "+", the whole To header address is "Alice+student@gmail.com". This means when Alice receives a call with this kind of To header address, she will know it is one of her students calling.

To show that the callee can use subaddress, the To header address is added in window of the incoming call alert and modified as a hyperlink to the record of url-secret (using To header address to identify the record) in the database. Also, making a SIP call with Message-ID in the command line is added. The format is "-r <mid> <url-to-call>"

Features:

* Enabled the caller to enter the Message-ID in the GUI interface.;
* Included a new SIP header (Sender-Ref) in the INVITE method;
* Displayed the value of Message-ID in the incoming call alert;
* Modified the value of Message-ID as a hyperlink to the record of the email-secret (using Message-ID to identify the record) in the database;
* Added the To header address in the incoming call alert and modify it as a hyperlink to the record of url-secret (using To header address to identify the record) in the database;
* Made a SIP call with Message-ID in the command line;

Implementation:
See detailed implementation in Appendix B.

3. CureSPIT Contact Information Extractor

3.1 Specification

To support a web-to-call relation by collecting contact information of potential callers. A Firefox Add-on is used to extract the contact information from HTTP response if the header of the response contains the META tag and the attribute of HTTP-EQUIV="Correspondence-URIS"[13]. Usually the contact information is the Correspondence-URI which includes mailbox / SIP-URI / SIPS-URI / telephone-URI.

After obtaining the contact information, the Add-on can insert the information into the database based on the preference of the user.

3.2 Application Structure

content
----insert.xul
(show a window to ask the user whether to insert the contact information in the database and call function insertObject.insert() in “overlay.js”)
----options.xul
(define the user preference for the user to enter username and password)
----overlay.js
(setActivateStatus.settrue(): change the attribute "isActivated" in “modules/global.js” to true
setActivateStatus.settrue(): change the attribute "isActivated" in “modules/global.js” to false
com.extract.overlay.show(): show the contact information in one page
insertObject.insert(): insert the contact information to the database using the attribute "contactArray", "typeArray" in “modules/global.js”, the attribute "extensions.extract.preflogin" in “options.xul”.)

----overlay.xul
(overlay the statusbar as the GUI for this application. Call function setActivateStatus.settrue (),setActivateStatus.setfalse(),com.extract.overlay.show() in “overlay.js”)
modules
----global.js
locale
chrome.manifest
install.rdf

3.3 Usage and description

Name: CureSPIT Contact Information Extractor

Platforms: Firefox-3.6.*

Version: CCIE(CureSPIT Contact Information Extractor)- 1.0

Usage:
The Add-on can update the database if it can connect to the database.

1. At first, drag the file "extractor.xpi" to the Firefox to install the Add-on

2. The Add-on is embedded in the statusbar of Firefox. In order to use the Add-on, the user should right click under the panel of the status bar.

3. There will be 3 buttons, "activate", "deactivate" and "show contact information". In the default situation, the Add-on is not activated. The user can click the "activate", then click the "show contact information" to see the contact information. Then the Add-on will ask the user whether to insert the information into the database, if user clicks "yes", the Add-on will insert it into the database.

Description:

CureSPIT Contact Information Extractor is used to extract the contact information when the client is browsing web. It is from HTTP response header, when it contains the META tag and the attribute of HTTP-EQUIV="Correspondence-URIS". Usually the contact information is the Correspondence-URI which includes mailbox / SIP-URI / SIPS-URI / telephone- URI. After getting the contact information, it will update the database of the CureSPIT client.

Features:
* Added a user preference window for the CureSPIT client to enter username and password;
* Added the function to activate or deactivate the Add-on;
* Added the function to extract contact information and update the database;

Source Code

Please see the folder “Source Code”.

4. CureSPIT ImapClient

4.1 Specification

The CureSPIT IMAPClient supports email-to-cal cross-media relations. It collects contact information in previous email messages and inserts it into the database. See the detailed pseudocode in Appendix C.

4.2 Usage and description

Name: CureSPIT ImapClient

Platforms: The code runs on, Linux, Windows NT/2000/XP, Windows 7 and Mac with JDK 1.6.

Version: CureSPIT ImapClient-1.0

Usage:
The application works only if it can connect to the database, because it must retrieve the email configuration from the database.

1. In order to run the application, please copy the files in folder lib to %JAVA_HOME%/jre/lib/ext. Those files are external jar of the application.

2. If the database demands the certificate, please download the certificate file from the database and copy it to %JAVA_HOME%/jre/lib/security.

3. Before running the application, please modify the file "email-config.txt" to indicate the username of CureSPIT username and the description of the email mailbox.

4. Use command "java Imapclient" to run the application.

Description:

This CureSPIT ImapClient extracts the contact information from the vCard of email messages and updates the database. VCard usually contains the detail information of the sender. A CureSPIT client will provide his/her username and ImapClient can use the username to get the username and password of the client's email address. For this purpose, the ImapClient periodically checks the email mailbox. Each time, it will fetch the messages of which the receive_date is between the time it last fetched and current time. Then it will filter the messages which have been answered. At present, there are two ways to filter the messages which have been responded based on which kind the mail box is. For AOL mailbox, ImapAnswerFlag[14] is used to determine whether the message has been responded. However, for Gmail mail box, the ImapAnsweredFlag is not set on the IMAP server. Thus, this program determines whether the message has been responded or not as follows:

(1) The messages in Sent mail folder should be scanned. And the To and CC addresses of a message in Sent mail are stored in an array.
(2) If the From address of a message in other folders is in the array, it indicates that the message has been responded.

Next, ImapClient gets the messages which are attached with vCards and extract contact information from vCards. At last, it will update the database.

Features:
* CureSPIT client read the file "email-config.txt" to as the configuration of user's mailbox.
* Added the function to fetch messages which have been responded from all folders using two different ways periodically;
At present, there are two ways to filter the messages which have been responded based on which kind the mail box is.
For mailbox like AOL, ImapAnswerFlag is used to determine whether the message has been responded.
For mailbox like Gmail, to determine whether the message has been responded or not, the messages in Sent mail can be made use of.
* Added the function to extract contact information from vCard and update database;

Source Code

Please see the folder "Source Code"

5. Conclusion: Problems what I encountered

During the 1st half of the fall semester, I was working on SIP Communcator extension for CureSPIT. I tried to understand the every functionality of the software by reading the source code which turned out to be not feasible. This took lots of efforts and time. Because besides the function making a SIP call, there are plenty of other functions. Studying other functions costs time and efforts. With the instructions of Kumiko, I learned to read the API documentation and focused on how to add a new header in the INIVTE method. Finally, I figured out how to do this.

In the 2nd half of the fall semester, I continued to work on SIP Communicator to implement the function "make a call in command line" and "show the hyperlink to the database". Also, I began to work with the RESTful database [8]. I managed to implement the Firefox Add-on and the IMAPClient using JAVAMAIL API. But they both had to communicate with the RESTful database ("getting or posting data"). I didn't figure it out how to implement this until after the end of the fall semester. I tried to use the default method from the instructions of the RESTful database [8], which doesn't support JSON and has escape string problem. This is why I couldn't finish it until after the end of fall semester. But by reading PHP code of the RESTful database interface and fixing it to support JSON object, I managed to finish it in winter break.

During the winter break, I succeeded in implementing the communication with the RESTful database. Then I wrote the functionality documents of the program I implemented or modified.

Although I've not implemented many functions, I spent time to understanding existing software what I had to modify. I learned how modifying software is often more difficult than writing a new software.

6. References

[1] SIP Communicator Home page:
http://sip-communicator.org/
[2] Description of the new Header in INVITE method:
http://tools.ietf.org/id/draft-ono-earlier-comm-references-00.txt
[3] Definition of Message-ID:
http://tools.ietf.org/html/rfc5322
[4] Description of SIP Header
http://www.ietf.org/rfc/rfc3261.txt Page 178
[5] Instructions on how to implement Firefox Add-on
http://blog.mozilla.com/addons/2009/01/28/how-to-develop-a-firefox-extension/
[6] Leonard Richardson, Sam Ruby,
RESTful Web Services, May 2007
[7] JSON in Java
http://www.JSON.org/js.html
[8] Phprestsql library.
http://phprestsql.sourceforge.net/
[9] JAVAMAIL API Documentation
http://java.sun.com/products/javamail/javadocs/index.html
[10] vCard
http://www.ietf.org/rfc/rfc2426.txt
[11] SVN of SIP Communicator

https://svn.java.net/svn/jitsi~svn/trunk/?rev=8118
[12] Definition of delimiter and subaddress
http://tools.ietf.org/html/rfc3966
[13] HTTP Header for Future Correspondence Addresses
http://tools.ietf.org/html/draft-shacham-http-corr-uris-00
[14] Answered Flag
http://download.oracle.com/javaee/1.4/api/javax/mail/Flags.Flag.html

Appendix A

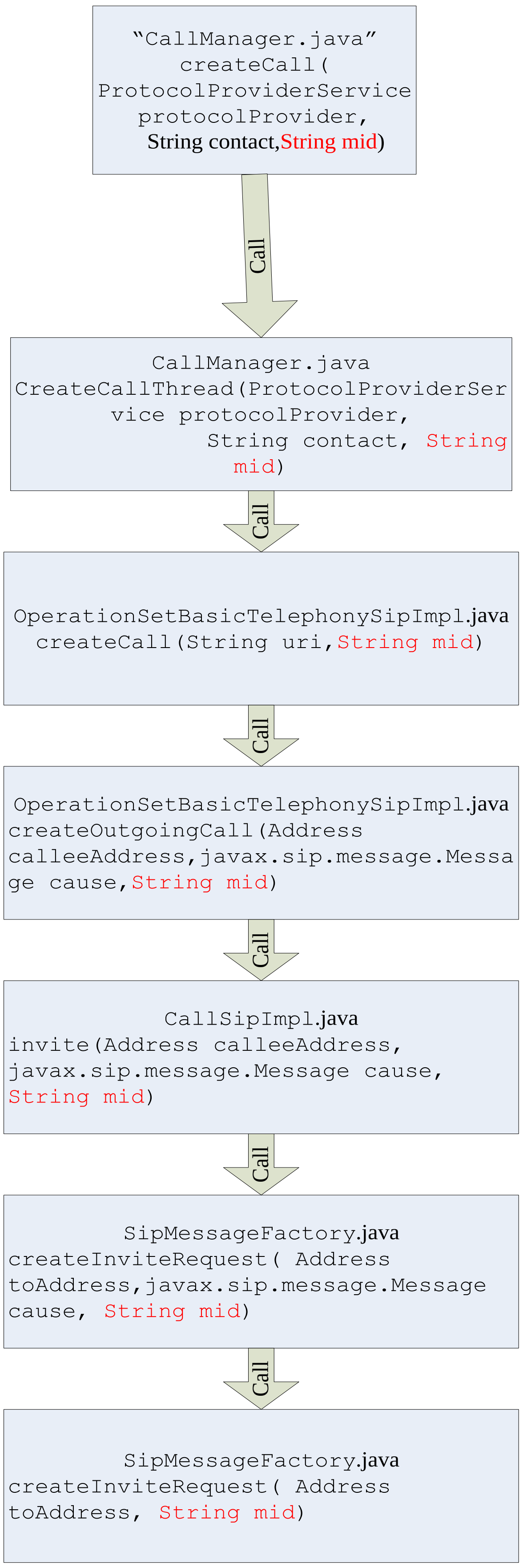
Properties added to “sip-communicator.properties”

```
net.java.sip.communicator.impl.gui.main.call.ReceivedCallDialog.linkPath=https\://almond.cs.columbia.edu/phpre
stsql/index.php
net.java.sip.communicator.impl.gui.main.call.ReceivedCallDialog.linkPath.SenderRef=email_secret
net.java.sip.communicator.impl.gui.main.call.ReceivedCallDialog.linkPath.To=url_secret
net.java.sip.communicator.impl.gui.main.call.ReceivedCallDialog.linkPath.username=bob_cure@cs.columbia.edu
```

Appendix B

Design of Important functions of the extenstion

Figure 1: Make a SIP call with Message-ID



```
"CallManager.java"
createCall(
ProtocolProviderService
protocolProvider,
String contact,String mid)
```

```
CallManager.java
CreateCallThread(ProtocolProviderSer
vice protocolProvider,
String contact, String
mid)
```

```
OperationSetBasicTelephonySipImpl.java
createCall(String uri,String mid)
```

```
OperationSetBasicTelephonySipImpl.java
createOutgoingCall(Address
calleeAddress,javax.sip.message.Messa
ge cause,String mid)
```

```
CallSipImpl.java
invite(Address calleeAddress,
javax.sip.message.Message cause,
String mid)
```

```
SipMessageFactory.java
createInviteRequest( Address
toAddress,javax.sip.message.Message
cause, String mid)
```

```
SipMessageFactory.java
createInviteRequest( Address
toAddress, String mid)
```

The above files are what should be modified to implement the function making a SIP call with

Message-ID. The above functions indicated in the flies are added. The parts in red means how the Message-ID is delivered to an INVITE message.

(The major modification is marked in red. Please focus on the red part, other code can be ignored)

Modification of CallManager.java

Added functions creating a call with mid

```
  public static void createCall(    ProtocolProviderService protocolProvider,
                                              String contact,String mid)
        {     if(mid.trim().length()==0)
                mid="unspecified";         //mid equals “unspecified” indicating the GUI entry of mid
hasn’t been entered any thing
              new  CreateCallThread(protocolProvider,  contact,mid).start();  //call  the  construction
function in CreateCallThread.java

         }
```

Added the construction function of CreateCallThread to receive mid above

```
//modified by yifan
                 public CreateCallThread(ProtocolProviderService protocolProvider,
                                          String contact,String mid)
                 {
                        this.protocolProvider = protocolProvider;
                        this.stringContact = contact;
                        this.contact = null;
                        this.mid=mid;
                 }
             public CreateCallThread(ProtocolProviderService protocolProvider,
                          Contact contact,String mid)
{
this.protocolProvider = protocolProvider;
this.contact = contact;
this.stringContact = null;
this.mid=mid;
}
```

Other construction functions set the mid to “unspecified”

```
               public CreateCallThread(ProtocolProviderService protocolProvider,
                          String contact)
```

```
{
this.protocolProvider = protocolProvider;
this.stringContact = contact;
this.contact = null;
this.mid=”unspecified”;
}

         public CreateCallThread(ProtocolProviderService protocolProvider,
                                    Contact contact)
             this.protocolProvider = protocolProvider;
             this.contact = contact;
             this.stringContact = null;
           this.mid=”unspecified”；
         }
```

Modified the run() function of CreateCallThread when calling the function in OperationSetBasicTelephonySipImpl.java(see next section)

```
        @Override
         public void run()
         {
             OperationSetBasicTelephony<?> telephonyOpSet
                 if (contact != null)
                     telephonyOpSet.createCall(contact， mid);
                 else if (stringContact != null)
                    telephonyOpSet.createCall(stringContact,mid);
             }
```

Modification of OperationSetBasicTelephonySipImpl.java

(The major modification is marked in red)

Add mid to the funciton “createCall” which is been called by the above CreateCallThread

```
     //modified by yifan
  public Call createCall(String callee,String mid)
        throws OperationFailedException,
        ParseException
   {
        Address toAddress = protocolProvider.parseAddressString(callee);
        if(mid.equals("unspecified"))
          return createOutgoingCall(toAddress, null);    //call the original function without mid
         else
        return createOutgoingCall(toAddress, null, mid);
   }
```

```
    //modified by yifan
    public Call createCall(Contact callee,String mid) throws OperationFailedException
   {
          Address toAddress;
            try
          {
                   toAddress = protocolProvider.parseAddressString(callee.getAddress());
          }
          catch (ParseException ex)
            {
                 // couldn't happen
                   logger.error(ex.getMessage(), ex);
                   throw new IllegalArgumentException(ex.getMessage());
            }
          if(mid.equals("mid"))
              return createOutgoingCall(toAddress, null);
            else
              return createOutgoingCall(toAddress, null,mid);
     }
```

Add the mid to the createOutgoingCall which calls the function invite in CallSipImpl.java

```
      //modified by yifan
    private synchronized CallSipImpl createOutgoingCall(Address calleeAddress,
            javax.sip.message.Message cause,String mid) throws OperationFailedException
    {
            CallSipImpl call = createOutgoingCall();
          call.invite(calleeAddress, cause,mid);
            return call;
   }
```

Modification of CallSipImpl.java

(The major modification is marked in red)

Add mid to invite function which calls createinviteRequest in SipMessageFactory.java

```
public CallPeerSipImpl invite(Address                              calleeAddress,
                                                            javax.sip.message.Message cause, String mid)
                 throws OperationFailedException
        {
                 // create the invite request
```

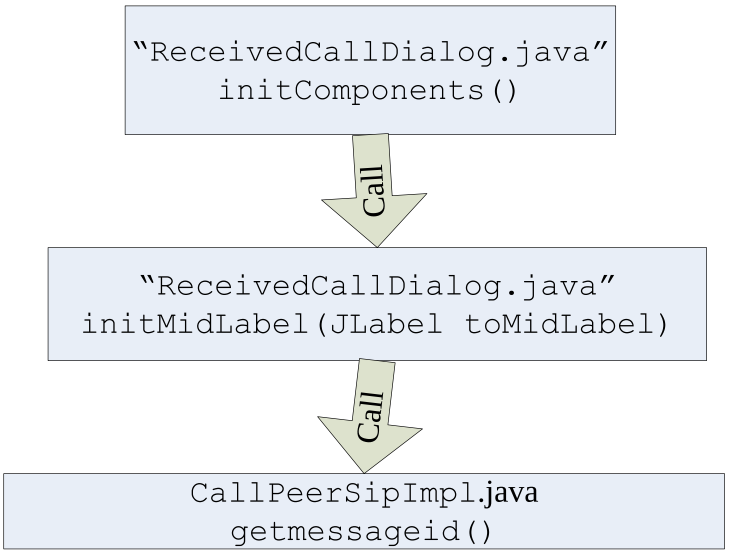
```
                Request invite = messageFactory
                    .createInviteRequest(calleeAddress, cause,mid);

                // Transaction
                ClientTransaction inviteTransaction = null;
                SipProvider jainSipProvider
                return callPeer;
        }
```

Modification of SipMessageFactory.java

Add the mid to the function createInviteRequest

```
public Request createInviteRequest( Address toAddress, String mid)
                throws OperationFailedException, IllegalArgumentException
        {

              //modified by yifan
            HeaderFactoryImpl headerfactory= (HeaderFactoryImpl)headerFactory;



              if(mid.charAt(0)!='<')
              {
                      mid="<"+mid+">";

              }
              try {

          // create the new Header "Sender-Ref" as an ExtensionHeader
            ExtensionHeader
isrh=headerfactory.createExtensionHeader("Sender-Ref",mid+" ;type=\"email\"");

            if(isrh!=null)
              {
                      try {
//add the header to the invite request
                      invite.addHeader(isrh);
              } catch (NullPointerException e) {
                      //    Auto-generated catch block
                      e.printStackTrace();
              }
              }
            } catch (ParseException e) {
              // Auto-generated catch block
              e.printStackTrace();
            }

                // Add the ReplacesHeader if any.
                if (replacesHeader != null)
                {
                return invite;
        }
```

Figure 2: Receive and show the Message-ID


"ReceivedCallDialog.java"
initComponents()
Call
"ReceivedCallDialog.java"
initMidLabel(JLabel toMidLabel)
Call
CallPeerSipImpl.java
getmessageid()

The modifications of source code in order to implement the function: receive and show Message-ID

(The major modification is marked in red. Please focus on the red part, other code can be ignored)

Modification of ReceivedCallDialog.java

(The major modification is marked in red)

Modified the function initComponents in ReceivedCallDialog.java

```
    private void initComponents()
        {
            JPanel mainPanel = new JPanel(new GridBagLayout());

            // disable html rendering
```

```
// callLabel.putClientProperty("html.disable", Boolean.TRUE);

  JPanel buttonsPanel = new TransparentPanel(new GridBagLayout());

  callButton = new SIPCommButton(
          ImageLoader.getImage(ImageLoader.CALL_BUTTON_BG));

  hangupButton = new SIPCommButton(
          ImageLoader.getImage(ImageLoader.HANGUP_BUTTON_BG));

  mainPanel.setPreferredSize(new Dimension(800, 180));
  mainPanel.setOpaque(false);
  mainPanel.setBorder(BorderFactory.createEmptyBorder(40, 40, 40, 40));

  callButton.setName(CALL_BUTTON);
  hangupButton.setName(HANGUP_BUTTON);

  callButton.addActionListener(this);
  hangupButton.addActionListener(this);

  this.initCallLabel(callLabel);

  receivedCallWindow.add(mainPanel);

  GridBagConstraints mainConstraints = new GridBagConstraints();
  mainConstraints.anchor = GridBagConstraints.WEST;
  mainConstraints.gridx = 0;
  mainConstraints.ipadx=10;
  mainConstraints.gridy = 2;
  mainConstraints.weightx = 3;
  mainPanel.add(callLabel, mainConstraints);

if(incomingCall.getClass().getName()=="net.java.sip.communicator.impl.protocol.sip.CallSipImpl")

  {
  mainConstraints.anchor = GridBagConstraints.WEST;
  mainConstraints.gridx = 0;
  mainConstraints.gridy = 3;
  mainConstraints.weightx = 3;
  Iterator<? extends CallPeer> peersIter = incomingCall.getCallPeers();
  CallPeer peer = peersIter.next();
String mid=peer.getmessageid(); // get the Message-ID from the first CallPeer because
only the situation where there's one callpeer is considered
if(mid!=null)
```

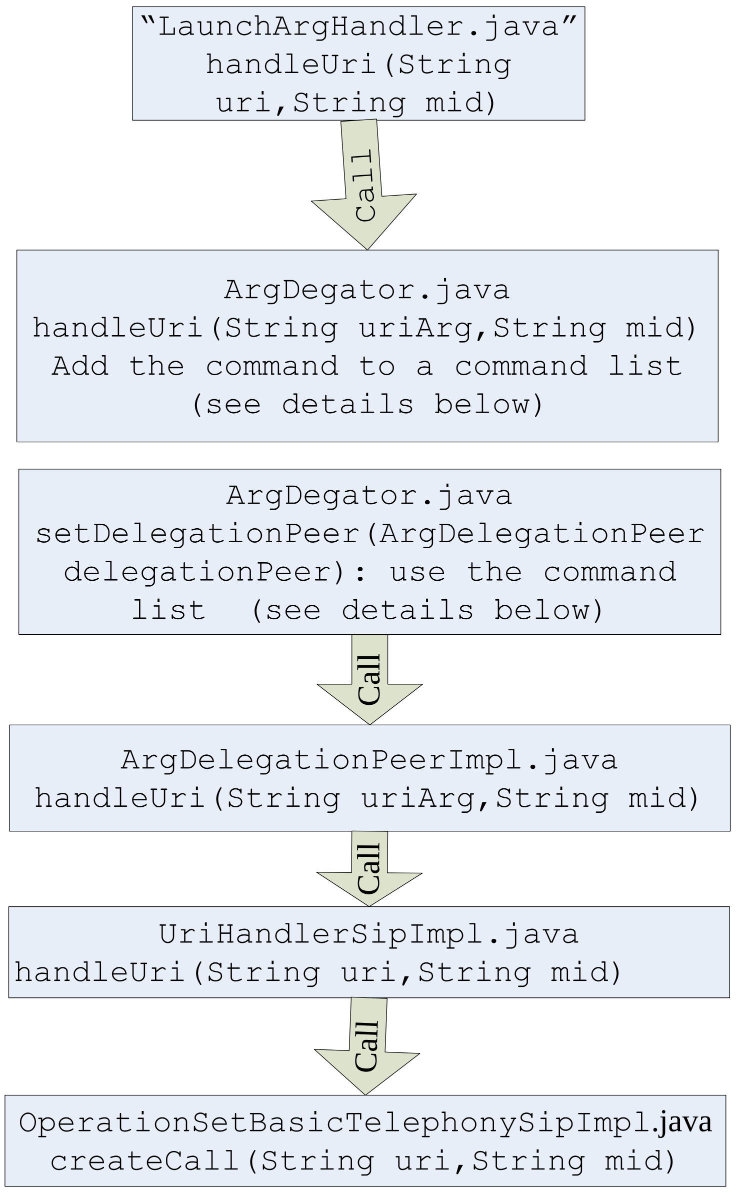
```
        {
        JLabel midLabel=new JLabel();

          initMidLabel(midLabel); //call the function initMidLabel

            mainPanel.add(midLabel,mainConstraints);
        }
                mainConstraints.anchor = GridBagConstraints.WEST;
          mainConstraints.gridx = 0;
          mainConstraints.gridy = 4;
          mainConstraints.weightx = 3;
        }
          JLabel toAddressLabel=new JLabel();
          inittoAddressLabel(toAddressLabel);
          mainPanel.add(toAddressLabel,mainConstraints);

          mainConstraints.anchor = GridBagConstraints.CENTER;
          mainConstraints.gridx = 1;
          mainConstraints.weightx = 0;
          mainPanel.add(Box.createHorizontalStrut(HGAP), mainConstraints);
          mainConstraints.anchor = GridBagConstraints.CENTER;
          mainConstraints.gridx = 2;
          mainConstraints.weightx = 0;
          mainPanel.add(buttonsPanel, mainConstraints);

          GridBagConstraints buttonConstraints = new GridBagConstraints();
          buttonConstraints.gridx = 0;
          buttonConstraints.gridy = 0;
          buttonsPanel.add(callButton, buttonConstraints);
          buttonConstraints.gridx = 1;
          buttonsPanel.add(Box.createHorizontalStrut(HGAP));
          buttonConstraints.gridx = 2;
          buttonsPanel.add(hangupButton, buttonConstraints);
    }
```

Added the function initMidLabel

```
private void initMidLabel(JLabel toMidLabel)
    {        Iterator<? extends CallPeer> peersIter = incomingCall.getCallPeers();
              CallPeer peer = peersIter.next();
             String mid=peer.getmessageid();
             String midForShow=mid;
```

```
                if(mid.indexOf(">")>0)
                {
                    mid=mid.substring(0, mid.indexOf(">")+1);
                    mid=mid.replaceFirst("<","%3C;");
                    mid=mid.replaceFirst(">", "%3E;");
                }
            final String midForUrl=mid; //midForUrl is used to set the URL in the database, using URL
escape


                if(midForShow.indexOf(">")>0)
                {
                    midForShow=midForShow.substring(0, midForShow.indexOf(">")+1);
                    midForShow=midForShow.replaceFirst("<"," &lt");
                    midForShow=midForShow.replaceFirst(">", "&gt");
                }
             //midForShow is used to set the content of toMidLabel using html escape
            toMidLabel.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
//set event function to the toMidLabel which is used to open the URL in the database
toMidLabel.addMouseListener(new MouseAdapter()
              {
                    @Override
                    public void mouseClicked(MouseEvent e) {
             String                                                                       linkPath=
GuiActivator.getConfigurationService().getString("net.java.sip.communicator.impl.gui.main.call.R
eceivedCallDialog.linkPath");
                  String
SenderRef=GuiActivator.getConfigurationService().getString("net.java.sip.communicator.impl.gui.
main.call.ReceivedCallDialog.linkPath.SenderRef");
             //     desktop.browse(new URI(linkPath+"/"+SenderRef+"/"+midForUrl));
                 openURL(linkPath+"/"+SenderRef+"/"+midForUrl); //open the URL using midForUrl,
linkPath and SenderRef can be achieved or modified in the property file (see above)

                    }
              });


             //set the content of toMidLabel
                toMidLabel.setText("<html>Sender-Ref:                                          <a
href=\"er\">"+midForShow+"</a></html>\n");

     }
```

Figure 3: Handling arguments with mid in command line


"LaunchArgHandler.java" handleUri(String uri,String mid)
Call
ArgDegator.java handleUri(String uriArg,String mid) Add the command to a command list (see details below)
ArgDegator.java setDelegationPeer(ArgDelegationPeer delegationPeer): use the command list (see details below)
Call
ArgDelegationPeerImpl.java handleUri(String uriArg,String mid)
Call
UriHandlerSipImpl.java handleUri(String uri,String mid)
Call
OperationSetBasicTelephonySipImpl.java createCall(String uri,String mid)

The modifications of source code in order to implement the function: make a SIP call in command line

(The major modification is marked in red. Please focus on the red part, other code can be ignored)

Modification of LaunchArgHandler.java

```
 public int handleArgs(String[] args)
     {
```

```
int returnAction = ACTION_CONTINUE;

for(int i = 0; i < args.length; i++)
{
        if (logger.isTraceEnabled())
                logger.trace("handling arg " + i);

        if (args[i].equals("--version") || args[i].equals("-v"))
        {
                handleVersionArg();
                //we're supposed to exit after printing version info
                returnAction = ACTION_EXIT;
                break;
        }
        else if (args[i].equals("--help") || args[i].equals("-h"))
        {
                handleHelpArg();
                //we're supposed to exit after printing the help message
                returnAction = ACTION_EXIT;
                break;
        }
        else if (args[i].equals("--debug") || args[i].equals("-d"))
        {
                handleDebugArg(args[i]);
                continue;
        }
        else if (args[i].equals("--ipv6") || args[i].equals("-6"))
        {
                handleIPv6Enforcement();
                break;
        }
        else if (args[i].equals("--ipv4") || args[i].equals("-4"))
        {
                handleIPv4Enforcement();
                break;
        }
        else if (args[i].startsWith("--config="))
        {
                returnAction = handleConfigArg(args[i]);

                if(returnAction == ACTION_ERROR)
                        break;
                else
                        continue;
                }
                else if (args[i].equals("-c"))
                {
                    //make sure we have at least one more argument left.
                    if( i == args.length - 1)
                    {
                        System.out.println(
                            "The \"-c\" option expects a directory parameter.");
                        returnAction = ACTION_ERROR;
                        break;
                    }
                    handleConfigArg(args[++i]);
                    continue;
                }
                else if (args[i].equals("--multiple") || args[i].equals("-m"))
                {
                    returnAction = ACTION_CONTINUE_LOCK_DISABLED;
                    continue;
                }
                //handle arguments with the command –r [uri-to-call] [mid] (make a call in
command line with mid)
                else if (( args[i].equals("-r")||args[i].equals("--refer"))&&i == args.length - 3)
                {
                    handleUri(args[i+2],args[i+1]);
                    break;
                }
                //if this is the last arg and it's not an option then it's probably
                //an URI
                else if ( i == args.length - 1
                        && !args[i].startsWith("-"))
                {
                    handleUri(args[i],"unspecified");
                    break;
                }
                else
                {
                    handleUnknownArg(args[i]);

                    errorCode = ERROR_CODE_UNKNOWN_ARG;
                    returnAction = ACTION_ERROR;
                    break;
                }
            }
            return returnAction;
        }
```
//add function to handle arguments with mid which calls function handleUri() in ArgDegator.java

```
private void handleUri(String uri,String mid)
        {
            if (logger.isTraceEnabled())
                logger.trace("Handling uri "+ uri+" "+mid);
            argDelegator.handleUri(uri,mid);
        }
```

Modification of ArgDegator.java

```
      protected void handleUri(String uriArg,String mid)
        {
            synchronized(recordedArgs)
            {
                if(uriDelegationPeer == null)
                {
                    recordedArgs.add("mid:"+mid+" "+uriArg); // add the command to make a
call with mid to the command list "recordedArgs"
                    return;
                }
            }

            uriDelegationPeer.handleUri(uriArg,mid);
        }

    public void setDelegationPeer(ArgDelegationPeer delegationPeer)
        {
            synchronized(recordedArgs)
            {
                if (logger.isTraceEnabled())
                    logger.trace("Someone set a delegationPeer. "
                                +"Will dispatch "+ recordedArgs.size() +" args");
                this.uriDelegationPeer = delegationPeer;

                for (String arg : recordedArgs) // use the command list
                {
                    if (logger.isTraceEnabled())
                        logger.trace("Dispatching arg: " + arg);
              if(arg.length()>=8&&arg.indexOf(" ")>0)
              {           if(arg.substring(0,4).equals("mid:"))
```

```
                    {
                            uriDelegationPeer.handleUri(arg.substring(arg.indexOf(" ")+1,arg.length()) ,
arg.substring(4, arg.indexOf(" "))); // call function handleUri() in ArgDelegationPeerImpl.java

                    }}
                        else
                    uriDelegationPeer.handleUri(arg);
                }

                recordedArgs.clear();
            }
        }
```

Modification of ArgDelegationPeerImpl.java

```
public void handleUri(String uriArg,String mid)
        {
            if (logger.isTraceEnabled())
                logger.trace("Handling URI: " + uriArg+" "+mid);
            //first parse the uri and determine the scheme/protocol
            //the parsing is currently a bit oversimplified so we'd probably need
            //to revisit it at some point.
            int colonIndex = uriArg.indexOf(":");

            if( colonIndex == -1)
            {
                //no scheme, we don't know how to handle the URI
                ArgDelegationActivator.getUIService().getPopupDialog()
                    .showMessagePopupDialog(
                            "Could not determine how to handle: " + uriArg
                            + ".\nNo protocol scheme found.",
                            "Error handling URI",
                            PopupDialog.ERROR_MESSAGE);
                return;
            }

            String scheme = uriArg.substring(0, colonIndex);

            UriHandler handler;
            synchronized (uriHandlers) {
                handler = uriHandlers.get(scheme);
            }
            //if handler is null we need to tell the user.
            if(handler == null)
            {
                if (logger.isTraceEnabled())
                    logger.trace("Couldn't open " + uriArg
                             + "No handler found for protocol"+ scheme);
                ArgDelegationActivator.getUIService().getPopupDialog()
                    .showMessagePopupDialog(
                         "\"" + scheme + "\" URIs are currently not supported.",
                         "Error handling URI",
                         PopupDialog.ERROR_MESSAGE);
                return;
            }

            //we're all set. let's do the handling now.
            try
            {
               handler.handleUri(uriArg,mid); //call fucntion handleUri() in UriHandlerSipImpl.java

            }
            //catch every possible exception
            catch(Throwable thr)
            {
                // ThreadDeath should always be re-thrown.
                if (thr instanceof ThreadDeath)
                    throw (ThreadDeath) thr;

                ArgDelegationActivator.getUIService().getPopupDialog()
                    .showMessagePopupDialog(
                        "Error handling " + uriArg,
                     "Error handling URI",
                     PopupDialog.ERROR_MESSAGE);
                logger.error("Failed to handle \""+ uriArg +"\"", thr);
            }
        }
```

Modification of UriHandlerSipImpl.java

```
  public void handleUri(String uri,String mid)
    {

        synchronized (storedAccountsAreLoaded)
```

```
{
        if (!storedAccountsAreLoaded[0])
        {
                if (uris == null)
                {
                        uris = new LinkedList<String>();
                }
                uris.add(uri);
                return;
        }
}

ProtocolProviderService provider;
try
{
        provider = selectHandlingProvider(uri);
}
catch (OperationFailedException exc)
{
        // The operation has been canceled by the user. Bail out.
        if (logger.isTraceEnabled())
                logger.trace("User canceled handling of uri " + uri+" "+mid);
        return;
}

// if provider is null then we need to tell the user to create an
// account
if (provider == null)
{
        showErrorMessage(
                "You need to configure at least one SIP account \n"
                        + "to be able to call " + uri, null);
        return;
}

OperationSetBasicTelephony<?> telephonyOpSet
        = provider.getOperationSet(OperationSetBasicTelephony.class);

try
{new ProtocolRegistrationThread(uri, provider).start();
try {
        Thread.sleep(5000);
} catch (InterruptedException e) {
        // Auto-generated catch block
```

```
                    e.printStackTrace();
                }
telephonyOpSet.createCall(uri,mid); //call createcall in OperationSetBasicTelephonySipImpl.java

                }
                catch (OperationFailedException exc)
                {
                    // make sure that we prompt for registration only if it is really
                    // required by the provider.
                    if (exc.getErrorCode() == OperationFailedException.PROVIDER_NOT_REGISTERED)
                    {
                        promptForRegistration(uri, provider);
                    }
                    showErrorMessage("Failed to create a call to " + uri, exc);
                }
                catch (ParseException exc)
                {
                    showErrorMessage(
                        uri + " does not appear to be a valid SIP address", exc);
                }
            }
```

Files and Functions have been modified of SIP Communicator

----/src/net/java/sip/communicator/service/protocol/CallPeer.java
public String getmessageid(): get the value of Sender-Ref Header from the INVITE method
public String getToAddress(): get the value of To Header from the INVITE method

----/src/net/java/sip/communicator/service/protocol/OperationSetBasicTelephony.java
public Call createCall(Contact contact, String mid): in the structure of making a SIP call (see below)

----/src/net/java/sip/communicator/service/argdelegation/UriHandler.java
public void handleUri(String uri,String mid): in the structure of handling arguments with mid in command line(see below)

-----/src/net/java/sip/communicator/util/launchutils/ArgDelegationPeer.java
public void handleUri(String uriArg,String mid): in the structure of handling arguments with mid in command line(see below)

----/src/net/java/sip/communicator/util/launchutils/ArgDelegator.java
protected void handleUri(String uriArg,String mid): in the structure of handling arguments with

mid in command line(see below)

----/src/net/java/sip/communicator/util/launchutils/LaunchArgHandler.java
private void handleUri(String uri,String mid): in the structure of handling arguments with mid in command line(see below)
----/src/net/java/sip/communicator/impl/argdelegation/ArgDelegationPeerImpl.java
public void handleUri(String uriArg,String mid): in the structure of handling arguments with mid in command line(see below)

----/src/net/java/sip/communicator/impl/gui/main/MainFrame.java
private void init(): Add the GUI entry for user to enter Message-ID

----/src/net/java/sip/communicator/impl/gui/main/call/CallManager.java
public static void createCall(ProtocolProviderService protocolProvider, String contact,String mid): create a call with mid (the structure of making a call is shown below)
public static void createCall(ProtocolProviderService protocolProvider,Contact contact,String mid);
public CreateCallThread(ProtocolProviderService protocolProvider,String contact,String mid);

----/src/net/java/sip/communicator/impl/gui/main/call/ReceivedCallDialog.java
public static void openURL(String url): open a URL, this is used when open the hyperlink of To header address of Message-ID
The linkpath to the database must be set for constructing the URL. It is a user property which can be set in the file "SIP Communitator/sip-communicator.properties" in the User Profile path of the OS.
For example, in Windows 7, the file is in the folder "C:\Users\[Username] \AppData\Roaming\SIP Communicator".
If the linkpath of the Message-ID is "https://almond.cs.columbia.edu/phprestsql/index.php/email_secret/[mid]", the URL can be constructed with an exact mid.
If the linkpath of the To header address is "https://almond.cs.columbia.edu/phprestsql/index.php/url_secret/[username]/[To header]/sip/[subaddress]", the URL can be constructed with an exact username To header and subaddress. The username can be set in the user property fiel "SIP Communitator/sip-communicator.properties" too.

private void initComponents(): Add value of the Message-ID and To header address to the GUI
private void inittoAddressLabel(JLabel toAddressLabel): Modify the To header address as the hyperlink

private void initCallLabel(JLabel callLabel): Modify the Message-ID as the hyperlink

----/src/net/java/sip/communicator/impl/gui/main/call/ChooseCallAccountPopupMenu.java
private void addTelephonyProviderItem(final ProtocolProviderService telephonyProvider,final

String contactString,final Class<? extends OperationSet> opSetClass):

function createcall() in CallManager.java is called (There are many ways in the GUI to make a call. All these ways should call the function createcall() in CallManager.java)

private void addTelephonyContactItem(final UIContactDetail telephonyContact,final Class<? extends OperationSet> opSetClass): function createcall() in CallManager.java is called
private void addTelephonyChatTransportItem(final ChatTransport telTransport,final Class<? extends OperationSet> opSetClass): function createcall() in CallManager.java is called

----/src/net/java/sip/communicator/impl/gui/main/chat/toolBars/MainToolBar.java
public void actionPerformed(ActionEvent e):function createcall() in CallManager.java is called

----/src/net/java/sip/communicator/impl/gui/main/contactlist/ContactListTreeCellRenderer.java
private void call(TreeNode treeNode):function createcall() in CallManager.java is called

----/src/net/java/sip/communicator/impl/gui/main/contactlist/MetaContactRightButtonMenu.java
public void actionPerformed(ActionEvent e):function createcall() in CallManager.java is called

----/src/net/java/sip/communicator/impl/gui/main/contactlist/SearchFieldUI.java
private void updateCallIcon(MouseEvent evt):function createcall() in CallManager.java is called

----/src/net/java/sip/communicator/impl/gui/main/contactlist/SourceContactRightButtonMenu.java
private Component initCallMenu():function createcall() in CallManager.java is called

----/src/net/java/sip/communicator/impl/gui/main/contactlist/UnknownContactPanel.java
public UnknownContactPanel(MainFrame window): function createcall() in CallManager.java is called

----/src/net/java/sip/communicator/impl/protocol/sip/OperationSetBasicTelephonySipImpl.java
public Call createCall(String callee,String mid): in the structure of making a SIP call (see below)
public Call createCall(Contact callee,String mid): in the structure of making a SIP call (see below)
private CallSipImpl createOutgoingCall(Address calleeAddress,javax.sip.message.Message cause,String mid):in the structure of making a SIP call (see below)

----/src/net/java/sip/communicator/impl/protocol/sip/SipMessageFactory.java
public Request createInviteRequest(Address toAddress, String mid):in the structure of making a SIP call (see below)

----/src/net/java/sip/communicator/impl/protocol/sip/UriHandlerSipImpl.java
public void handleUri(String uri,String mid): handle arguments with mid in command line

----/src/net/java/sip/communicator/impl/protocol/sip/CallPeerSipImpl.java
public String getToAddress(): implementation of getting the value of To header address.

public String getmessageid(): implementation of getting the value of Message-ID

----/src/net/java/sip/communicator/impl/protocol/sip/CallSipImpl.java
public CallPeerSipImpl invite(Address calleeAddress,javax.sip.message.Message cause, String mid):in the structure of making a SIP call (see below)

----/src/net/java/sip/communicator/impl/protocol/gibberish/CallPeerGibberishImpl.java
public String getToAddress(): add unimplemented function (Note that Gibberish may not using the function getToaddress(), but it should exist becasue the interface

CallPeer.java has this function. The reason why CallPeer.java has this function is that getToaddress() in CallPeerSipImpl.java must be called when using polymorphism of Java. This case will happen a lot in the following)
public String getmessageid(): add unimplemented function (the case of polymorphism)

----/src/net/java/sip/communicator/impl/protocol/gibberish/OperationSetBasicTelephonyGibberish Impl.java
public Call createCall(String callee,String mid): add unimplemented function (the case of polymorphism)
public Call createCall(Contact callee,String mid): add unimplemented function (the case of polymorphism)

----/src/net/java/sip/communicator/impl/protocol/jabber/CallPeerJabberImpl.java
public String getToAddress(): add unimplemented function (the case of polymorphism)
public String getmessageid(): add unimplemented function (the case of polymorphism)

----/src/net/java/sip/communicator/impl/protocol/jabber/OperationSetBasicTelephonyJabberImpl.j ava
public Call createCall(String callee,String mid): add unimplemented function (the case of polymorphism)
public Call createCall(Contact callee,String mid): add unimplemented function (the case of polymorphism)

----/src/net/java/sip/communicator/impl/protocol/jabber/UriHandlerJabberImpl.java
public void handleUri(String uri,String mid): add unimplemented function (the case of polymorphism)

----/src/net/java/sip/communicator/impl/protocol/mock/MockCallPeer.java
public String getToAddress(): add unimplemented function (the case of polymorphism)
public String getmessageid(): add unimplemented function (the case of polymorphism)

----/src/net/java/sip/communicator/impl/protocol/mock/MockOperationSetBasicTelephony.java
public Call createCall(String callee,String mid): add unimplemented function (the case of polymorphism)

public Call createCall(Contact callee,String mid): add unimplemented function (the case of polymorphism)

----/src/net/java/sip/communicator/impl/protocol/rss/UriHandlerRssImpl.java
public void handleUri(String uri,String mid): add unimplemented function (the case of polymorphism)

Appendix C

Pseudocode

```
//Use while(true) {.....   sleep(n);} to implement the periodically to fetch the message. The fetch
//period is in the database
while(true)
{
   get username and description from the property file "email-config.txt";
   if(username==null||description==null)
        stop the application;
   get JSON object from the database using the username and description;
   get fetchPeriod from the JSON object;
   use the useridAtIMAPserver and passwdAtIMAPserver achieved from JSON object to connect
   to the mailbox;
   get the root folder of the mailbox;

   if( the IMAPAnsweredFlag ==1) //has AnswerdFlag like AOL
   {
          for(every folder in the mailbox)
          {
             fetch messages of which the receive_date is later than the lastfetch date which is
             achieved from JSON object;
                   for(every message in the folder)
                {
                     use the Answerflag to filter messages which have been responded;
                       for(every part of the message)
                                  if(the message has attachment)

                                   if(the attachment is vCard)
                                      {
                                        parse vCard and insert contact information to the database;
                                      }
                }
            }
      ]
    else    //hasn't Answeredflag like gmail
    {
        //the structure of Gmail mailbox: two folders inbox and [Gmail] are in the first level, other
        //folders are under [Gmail].
          get folder "Sent Mail" under [Gmail];
          gather the To, "cc" and "bcc" email address from the message in "Sent Mail" in sentarray;
          fetch messages of which the receive_date is later than the lastfetch date which is
          achieved from JSON object;
          for(every message in inbox folder)
            {

                filter messages which have been responded by comparing the from address to the
                address in sentarray;
                for(every part of the message)
                                        if(the message has attachment)

                                          if(the attachment is vCard)
                                             {
                                               parse vCard and insert contact information to the database;
                                             }
            }
        for(every folders under [Gmail])
         {
          fetch messages of which the receive_date is later than the lastfetch date which is
          achieved from JSON object;
          for(every message in inbox folder)
            {

                filter messages which have been responded by comparing the from address to the
                address in sentarray;
                for(every part of the message)
                                        if(the message has attachment)

                                          if(the attachment is vCard)
                                             {
                                               parse vCard and insert contact information to the database;
                                             }
            }
         }
}
```

```
//Periodically fetch messages, the value in the database indicates how many minutes the
//period has
            Thread.sleep(n*1000*60);

}
```