

Universal Inbox Application for the Series 60 Platform

Vanessa D. Gonzales
Author

vannie.gonzales@gmail.com

Randolph M. Espinosa
Author

randolphzippy@yahoo.com

William Emmanuel S. Yu
Adviser

wyu@ateneo.edu

Department of Information Systems and Computer Science
Ateneo de Manila University
+63(2) 426-6001 loc. 5660

ABSTRACT

The goal of this paper is to propose and describe the development of an actual, specific and limited unified messaging solution: a mobile-based universal inbox application for Symbian Series 60 phones, integrating access to SMS, MMS, and email messages. The general aim of the application is to intensify the message access capabilities of Series 60 phones by maximizing convenience and flexibility for its users.

The universal inbox was primarily developed on the Java 2 Micro Edition platform. However, it also employs Symbian application program interfaces (APIs) by communicating with a peer Symbian C++ application for crucial operations such as monitoring a mobile phone's normal inbox. The main features of the application include accessing, replying to and sending messages, as well as configuring settings for email retrieval using POP and IMAP.

Keywords

Unified messaging, universal inbox, mobile applications.

1. THE PROBLEM AND ITS BACKGROUND

1.1 Introduction

The ways through which people are able to communicate today are numerous, diverse, and continuously evolving. A person can receive all sorts of messages (email, fax, voicemail, phone call, SMS, and MMS among others) on all sorts of devices and interfaces (email inbox, fax machine, wire line phone, and mobile phone). The emergence of the technologies which enable users to have at hand a variety of communication channels has facilitated ease of communication, flexibility, and mobility. Given all these advantages, a significant number of users have encountered a problem of message influx: They are faced with messages of a quantity and variety much greater than they can handle.

There have been attempts from the industry to help users cope with these complexities. One of the more popular approaches is *unified messaging*. The International Engineering Consortium defines unified messaging as the integration of several different communications media, such that users will be able to retrieve and send voice, fax, and email messages from a single interface, whether it be a wire line phone, wireless phone, PC, or Internet-enabled PC [15].

1.2 Review of Related Literature

There exists already a multitude of unified messaging products and services available to the market. They have been created to be compatible with existing platforms for voicemail, fax, and email. However, the solutions we present here are limited to those that are most relevant to our own proposed solution. The first two projects are open source email applications that are similar to the universal inbox we propose to develop, while the last one is a commercially available solution with very similar functionality with the one we are proposing.

1.2.1 Mail4Me

Mail4Me is an open source email client, as well as an email library. It is a lightweight implementation of SMTP, POP3 and IMAP protocols, which includes MIME support for email attachments. It can be used in wireless J2ME/MIDP devices. The MIDlet allows browsing through a POP3 or IMAP inbox, reading individual messages, and composing and sending new ones. The MIDlet makes use of the MIME features and thus is able to display messages composed of text and pictures in PNG format [2].

Email protocols are encapsulated by one class each: SmtplibClient, Pop3Client, and ImapClient. Messages are received through the InboxClient interface, which is implemented by both Pop3Client and ImapClient, while SmtplibClient is used in sending messages. Other class relations are illustrated in the following figure.

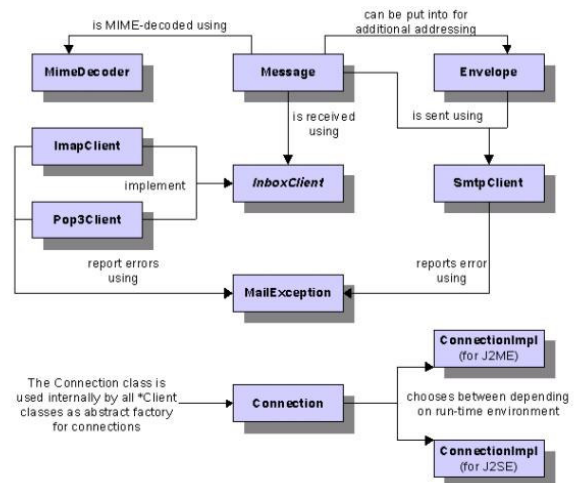


Figure 1. Class Relations in Mail4Me [2]

1.2.2 Java POP3 Mail Viewer

This package is a fully functional program which can be used to retrieve POP3 emails from J2ME-enabled mobile phones. For email retrieval functionality, JavaMail packages released by Javasoft have been used. Generally, attachments are not supported and only the first 1500 characters of the message are sent to the phone [16].

1.2.3 BlackBerry

BlackBerry is a well-known wireless handheld device which supports email, web browsing, SMS, and other wireless information service. From the first device that has been released by its developer Research In Motion, Ltd. in 1999, the BlackBerry is now available in different models. As a unified messaging solution, the device features a single inbox for SMS, MMS, and email messages. BlackBerry supports connected, “push” email messaging through a software package named the BlackBerry Enterprise Server which integrates into organizations’ email systems. The server monitors a user’s local inbox for new messages, which are transmitted to an RIM operations center, then to the wireless provider, and finally to the user’s BlackBerry device. The BlackBerry email client is one of the features provided by a limited number of Nokia mobile phones models: Nokia 6810, 6820, 9300 and 9500. Here in the Philippines, BlackBerry is supported by both Globe Telecom and SMART Communications [1].

1.3 Statement of the Problem and Objectives

It is the aim of this project to provide Series 60 mobile phone users with the convenience of a unified messaging solution that will integrate access to their SMS, MMS, and email messages through a single interface.

The advantages we aim to provide for the users are: message access integration, mobility, and enhanced response potential. Each of these are discussed in the next section.

1.4 Significance of the Study

The influx of different types of messages through different media is an inevitable reality in this technology-driven age with growing needs for communication. As such, we are convinced that an effective unified messaging solution cannot be dismissed as a convenience, and maybe later on as a necessity, given its benefits.

The first and most significant advantage of using a universal inbox is message access integration. Users will have the convenience of employing only one interface to access their messages and check for new ones, as opposed to using multiple interfaces for different types of messages. This avoids the situation of “neglecting” messages when a particular interface is not checked often enough.

Another major advantage is mobility. Since the application runs on the mobile phone, the user is able to access messages virtually anytime, anywhere. This is extremely important for individuals whose professions and lifestyles demand that they travel frequently without compromising their availability for communication. In short, it is indispensable for anyone who wants to be always within reach.

Given the two advantages mentioned above, one consequent advantage would be that of enhanced response potential. A user will be more likely to respond quicker, especially to urgent and important messages, since he or she has easier and more flexible access to them.

1.5 Scope and Limitation of the Study

Now we will establish how the general scope of the proposed solution has been decided on. These bases derive from research on communication scenarios in the Philippines and abroad, and on personal observations and experiences.

1.5.1 Mobile-based

Obviously, the greatest advantage of a mobile solution is that the user can take it anywhere. Especially when concerning urgent and important messages, the user naturally demands reliable access to communication media that is always within reach – the most prevalent of which today is the mobile phone.

In countries such as the Philippines, the growth of the mobile phone market has been astonishing. This is in comparison not only to growth rates of other countries, but also to its own computer penetration rate, as seen in the figure below.

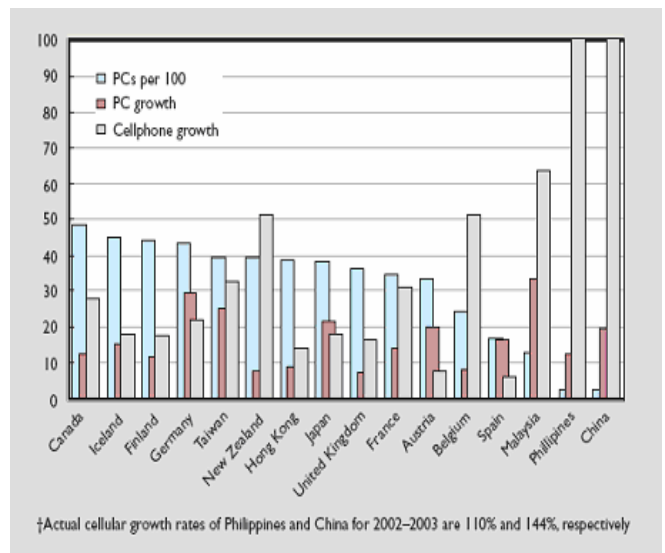


Figure 2. Computer Penetration Rates (2002) and Growth Rates of PC Penetration and Mobile Phones (2000-2002) [3]

According to Fife and Pereira, countries with relatively low computer penetration and high mobile phone growth rates relative to computer growth rates, such as the Philippines, are generally also countries with higher adoption rates of mobile data applications [3]. Furthermore, they tend to see mobile access devices such as mobile phones as providing a ‘relative advantage’ in enjoying the benefits of email and short messaging— two of the more popular applications of the fixed Internet [15]. Thus, we propose a mobile-based solution because of mobile technology’s accessibility, pervasiveness, and adaptability.

1.5.2 Targeting Series 60 Phones

The scope of target devices has been limited to these mobile phones because of its wide market. More than thirty devices based on Series 60 have been launched by Nokia alone, and over 28 million units have been sold as of the third quarter of 2005. The Series 60 platform has five other licensees: Siemens, Panasonic, Samsung, Lenovo and LG [11].

1.5.3 Focusing on Email, SMS and MMS

As mentioned above, email and short messaging are two of the most popular applications at present, while the newer MMS is definitely promising. In addition to their popularity, these types of messages are most appropriate for the target devices' capabilities.

1.6 Relevant Terms and Technologies

In this section, we will define terms and technologies in phone and Internet messaging, as well as two popular development platforms for mobile devices and their messaging APIs. Additionally, we will also illustrate how all these technologies are related to each other.

1.6.1 Phone Messaging

SMS (Short Message Service). This service is commonly available to mobile phones as a way of sending and receiving short text messages to and from mobile telephones. It provides an easy way for individuals to communicate with one another and with external systems [8]. Messages are sent using a store-and-forward mechanism to an SMS Centre which will send the messages to the actual recipients.

MMS (Multimedia Messaging System). MMS is a system of transmitting multimedia content over wireless networks.

1.6.2 Internet Messaging

Email. Short for electronic mail, it is the method of composing, sending, and receiving messages through an electronic communication system.

SMTP (Simple Mail Transfer Protocol). SMTP is a text-based protocol which serves as the de facto standard for email transmission across the Internet [14]. When a user sends an email message from a client application, the email client communicates with the SMTP from the email provider. The SMTP server then looks at the address to which the message will be sent and then communicates with the SMTP server of the destination address domain [7].

Secure POP3 (Post Office Protocol version 3). POP3 is a protocol used by a client to retrieve email from a remote server over TCP/IP. It has been designed for single-access, disconnected "pull" operation: It generally operates by connecting to the server, downloading messages in the PC, deleting the server's copy of messages, and disconnecting from the server. As for Secure POP3, the server communicates with the client through SSL (Secure Socket Layer). SSL encryption provides authentication and communications privacy for the Internet connection, thus making POP3 secure [13]. The POP3 server essentially acts as an interface between the email clients and the data store that contains the email messages [7]

IMAP (Internet Message Access Protocol). Similar to POP3, IMAP is used by a client to retrieve email from a remote server over TCP/IP. However, it has been designed to accommodate simultaneous connections, and connected and disconnected modes of operation [5].

SASL (Simple Authentication and Security Layer). SASL is a framework which handles authentication and authorization for Internet protocols such as the three mentioned above. The protocol demands server identification and authentication, and can also negotiate protection of subsequent protocol interactions [12].

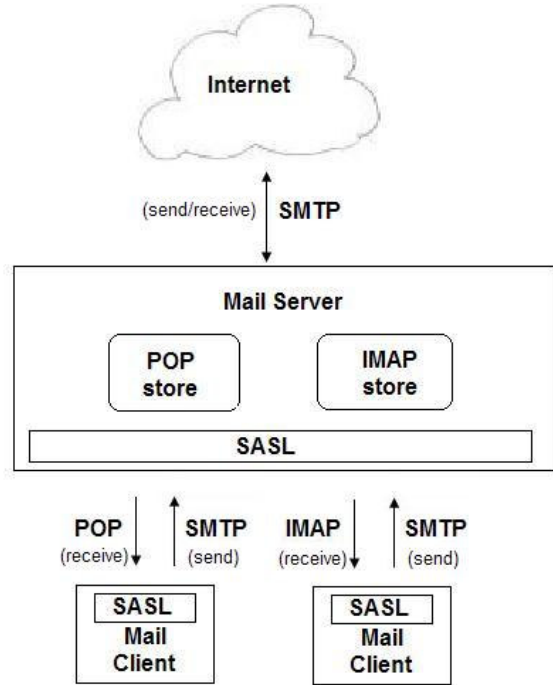


Figure 3. Interactions of the Email Protocols

1.6.3 Development Platforms

In this section, we will describe two development platforms that are crucial to the proposed solution, Symbian OS and Java 2 Micro Edition (J2ME). A general comparison of these two platforms is outlined in the Table 1.

1.6.3.1 Symbian

Symbian is an operating system produced by Symbian Ltd. It is specially designed for handheld devices, taking into consideration their limited resources such as power consumption and memory. It is a leader in the industry, beating competitors such as Windows Mobile, Linux and Palm OS.

More importantly, Symbian is also an open software and communications platform. Symbian OS runs applications written in the C++ language, which are compiled to machine code.

Table 1. Comparison of J2ME and Symbian OS Capabilities [9]

	J2ME	Symbian OS
Permitted App Size	A few dozen kilobytes	Multiple megabytes
Open Standard	Yes	Yes
Deployment	Large & growing	Smaller & growing
Supported by Multiple Manufacturers	Yes	Yes
OTA Installation	Yes	Yes
Runs Natively	No	Yes
Usual Coding Language	Java	C++
Communicates with Remote Server	Yes	Yes
2D Animation	Yes	Yes
3D Animation	No	Yes
Display Video	No	Yes
MIDI Audio	Yes	Yes
High-Quality Audio	Generally no	Yes
Access to SMS	Generally no	Yes
Access to IrDA & Bluetooth Ports	No	When available on phone
Access to Phonebook, Calendar, Etc.	No	When available on phone
Dial Phone	No	Yes
Cross-Platform Development Issues	Yes	Yes

1.6.3.2 J2ME (Java 2 Micro Edition)

It is a runtime and a collection of Java APIs for the development of software for constrained devices such as mobile phones and personal digital assistants (PDAs). It is most popular today in the field of mobile game development. Most applications run on CLDC (Connected Limited Device Configuration) and MIDP (Mobile Information Device Profile) which are implemented by new phones. These aim to describe the specifications of the mobile devices, to guide application development regarding the resources and capabilities of mobile information devices. As applications running on MIDP, they have been consequently termed as MIDlets [6].

Specifically, MIDlets are Java programs that run on the J2ME virtual machine. The main class should be a subclass of *javax.microedition.midlet.MIDlet*. The application and other resource files must be packaged in a JAR file and pre-verified. Once compiled, it can run in any J2ME-enabled device.

1.7 Platform-Specific Messaging APIs

1.7.1 Symbian

The Symbian messaging framework supports the sending and receiving of SMS, MMS, EMS (Enhanced Messaging Service), email and fax messages. The SendAs API enables the creation of new messages of the given types directly from another application. Email support includes SMTP, POP3, IMAP and standard attachments.

1.7.2 J2ME

The Wireless Messaging API (WMA) is an optional package that provides access to wireless communication resources such as SMS. It includes two packages, *javax.microedition.io* and *javax.wireless.messaging*. The first supports wireless messaging connections, while the second one enables applications to send and receive wireless messages.

2. METHODOLOGY

2.1 Project Description

The universal inbox is an application for the mobile phone which provides an integrated interface for accessing the user's email, SMS and MMS messages. It seeks to provide users with the convenience of checking only a single interface for messages, and the mobility of being able to access them through their phones.

Protocols supported are SMS/MMS via Phone API, POP, IMAP and SMTP. The proposed solution will be available for use on Series 60 mobile phones.

2.2 Description of Components

The universal inbox has four key components, which are described below. All other features and functionalities will be built upon these four foundational components:

1. The *user interface* must be simple and easy to learn.
2. The *email retrieval* function must support POP, IMAP and SMTP. It must also consider the nature of Internet connection available to the mobile phone.
3. The *SMS/MMS retrieval* function must operate according to the APIs of the mobile phone.
4. The *storage capacity* of the inbox must be adequate for the email messages received by the user.

2.3 Implementation Details

2.3.1 Development Platforms

The universal inbox will be implemented primarily using the J2ME platform, since we are most comfortable with the Java programming language. However, J2ME does not allow direct access to the phone's native APIs, including direct access to the phone's regular inbox. So in addition to the expected output of a fully functional J2ME application, a Symbian application will be developed in C++ which will act as a message listener on the phone's regular inbox. (For simplicity, we shall use the term "Symbian daemon" to refer to this application.) This workaround

solution has been derived from the implementation termed as “The Bridge” [4]. The Bridge proposes the use of TCP sockets for communication between J2ME MIDlets and Symbian applications for the purpose of exchanging information needed by the MIDlets which can only be accessed or retrieved by a native Symbian application.

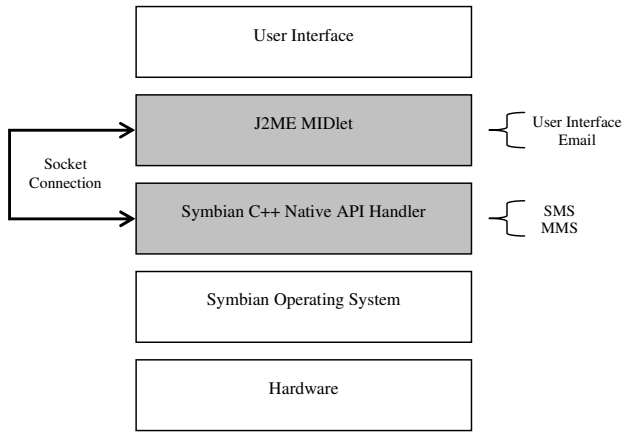


Figure 4. Project Diagram

Thus, the universal inbox project is mainly divided into two applications: one is in developed the J2ME platform, and the other in Symbian OS. The bulk of the components such as the user interface and email retrieval will be implemented in a J2ME MIDlet. As mentioned previously, the MIDlet will need a Symbian daemon to handle access to information available only to native applications. This daemon will handle the components for access to SMS and MMS messages. It will communicate to the main application through a TCP socket connection on port 8100 and the local loopback address (127.0.0.1).

The specific tools that will be used for the development and testing of the universal inbox are outlined in the following section.

2.3.2 Tools and SDKs

For the development of the universal inbox application, we used the tools and software development kits as indicated below.

Table 2. Tools for Development

Component	IDE	Development Kit
J2ME MIDlet	JCreator	J2ME Wireless Toolkit 2.2 < http://java.sun.com >
Symbian Application	Visual Studio .NET	Series 60 Software Development Kit < http://www.nokia.com > - Symbian SDK v1.2 - Symbian SDK 2 nd edition Feature Pack 2
Testing	Built-in emulators and actual Nokia Series 60 phone	

In developing the MIDlet, editing of the source codes is done through Jcreator, while building, testing, and packaging into JAR files is accomplished through the wireless toolkit’s KToolbar.

On the other hand, Visual Studio .NET fully integrates editing, building, testing and packaging into SIS files. We used two different SDKs: The older version is used because it is compatible with the actual phone unit we use for testing (Nokia 7650), while the newer version is used for the MMS capabilities of its emulator.

2.3.3 The Development Process

Since the universal inbox project involves two applications to be developed on two different platforms, the development process for each required a different set of tools and software development kits. Thus, both applications were developed separately and can be tested together only through installation to an actual phone. This is because the emulators provided for both software development kits are platform-specific and cannot recognize other applications properly. But the process of having to install on the phone is too time-consuming, and during the development process we have devised a way to work around this problem. The following diagram illustrates the solution:

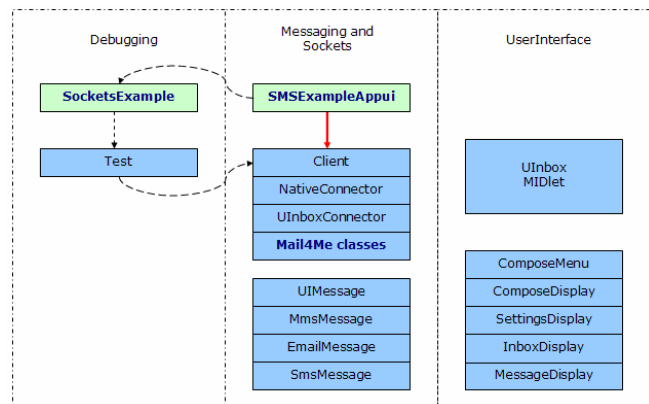


Figure 5. Classes and Debugging Solution

The classes in blue were developed in J2ME while those in green were developed in Symbian. The red line represents the actual socket connection for communication between the MIDlet and the Symbian daemon. This is the connection that can only be tested through the installation of both applications on the mobile phone. But for the most part during the development process, this socket connection was *simulated* through some debugging programs, namely the *Test* class and the *Socket* application. The *SMSEExampleAppui* establishes a socket connection with, and sends the data regarding incoming SMS and MMS messages to the *Socket* application, which displays the data that has been sent to it. On the Java side, the *Test* class also establishes a socket connection with the *NativeConnector* class of the universal inbox, and sends the data that has been received by the *Sockets* application in Symbian. (The data to be sent is hard-coded in the *Test* class.)

The first step for the entire development process was to have the Symbian daemon which listens for incoming messages in the phone’s regular inbox. We used the *SMSEExample* provided by

Nokia (source code provided in the appendix) which listens for incoming SMS messages in the observed folder. Specifically, we modified the *HandleSessionEventL()* method in *SMSEExampleAppui*. This method listens for particular events in an observed folder, which in this case is the phone's regular inbox. Modifications included adding the capability to also listen for incoming MMS messages, and transmitting the appropriate details of the message to the socket connection.

```
void CSMSEExampleAppUi::HandleSessionEventL(TMsvSessionEvent
aEvent, TAny* aArg1, TAny* aArg2, TAny* /*aArg3*/)
{
    switch (aEvent)
    {
        case EMsvServerReady:
            // Initialise iMsvEntry
            ...
            break;

        case EMsvEntriesCreated:
            // Only look for changes in the Inbox
            ...
            break;

        case EMsvEntriesChanged:
            // Look for changes. When using
            // the emulator observed folder is drafts,
            // otherwise inbox.
            if (*(static_cast<TMsvId*>(aArg2)) ==
                KObservedFolderId)
            {
                CMsvEntrySelection* entries =
                    static_cast<CMsvEntrySelection*>(aArg1);
                if (iNewMessageId == entries->At(0))
                {
                    // Set entry context to the new message
                    iMsvEntry->SetEntryL(iNewMessageId);

                    // Check the type of the arrived message and
                    // and that the message is complete.
                    if (iMsvEntry->Entry().Complete() &&
                        iMsvEntry->Entry().iMtm == KUidMsgTypeSMS )
                    {
                        ...
                        // Get address of received message.
                        ...
                        // Get body text and write to output.
                    }
                    else if (iMsvEntry->Entry().Complete() &&
                        iMsvEntry->Entry().iMtm ==
                        KUidMsgTypeMultimedia )
                    {
                        ...
                        // Get address of received message.
                        ...
                        // Get attachment details
                        ...
                        // Write to output.
                    }
                }
            }
            break;

        default:
            break;
    }
}
```

**Code Snippet 1. SMSEExampleAppui.cpp:
HandleSessionEventL()**

We also added a new method *DaemonMainL()* which opens a socket server connection and waits for another application to connect to it. For debugging purposes, we also use the application's *SMSEExampleLogView* to output relevant information as it runs.

```
void CSMSEExampleAppUi::DaemonMainL()
{
    iLogView->DrawTextL( _L("In daemon!") );

    KTestPort=8100;
    TInetAddr addr(KInetAddrLoop, KTestPort);

    User::LeaveIfError(socketServ.Connect());
    User::LeaveIfError(listener.Open(socketServ,
        KAfInet, KSocketStream, KProtocolInetTcp));
    User::LeaveIfError(listener.Bind(addr));
    User::LeaveIfError(listener.Listen(1));

    blank.Open(socketServ);
}
```

```
listener.Accept( blank, status );
User::WaitForRequest(status);
}
```

Code Snippet 2. SMSEExampleAppui.cpp: DaemonMailL()

The next step was to develop the actual universal inbox MIDlet. It involved designing the user interface through the use of *javax.microedition.lcdui* classes. Socket communication with *SMSEExample* is handled by the *NativeConnector* class, which takes care of listening for incoming SMS and MMS messages.

```
public NativeConnector( Client c )
{
    client = c;
    try
    {
        StreamConnection conn = ( StreamConnection )Connector.open(
            "socket://127.0.0.1:8100" );
        in = conn.openInputStream();
    }
    catch( Exception e )
    {
        e.printStackTrace();
    }
    ...

    public void run()
    {
        while ( true )
        {
            String unit = incomingData();
            if( unit != null )
                client.acceptIncomingMessage( unit );
        }
    }
}
```

Code Snippet 3. NativeConnector.java

For email retrieval, we used the Mail4Me email library which conveniently implements POP and IMAP email retrieval. The email library is available for use under the Enhydra Public License (see Appendix A). We created the *UInboxConnector* which takes care of creating the appropriate email client and establishing its connection with the mail server.

```
public void retrieveEmail()
{
    new Thread()
    {
        public void run()
        {
            if( emailClient == null )
            {
                if ( imap )
                {
                    ...
                    emailClient = new ImapClient();
                }
                else
                {
                    emailClient = new Pop3Client();
                }
            }
            else
            {
                try
                {
                    emailClient.open( mailServerHost, 0, false,
                        mailServerUser, mailServerPass );

                    getEmailMessageList();

                    emailClient.close();

                    midlet.displayOther( "menu" );
                }
                catch ( Exception e )
                {
                    e.printStackTrace();
                }
            }
        }
    }.start();
}
```

Code Snippet 4. UinboxConnector.java: retrieveEmail()

The interactions between these different classes when the universal inbox retrieves the messages are described in the following section.

2.4 Message Flow

An incoming SMS or MMS message is detected by the Symbian daemon since it monitors the mobile phone's native inbox. It then creates its own copy of the message, reads its details, and transmits this information to the universal inbox MIDlet through a socket connection. The MIDlet processes the transmitted information and adds a new entry to the current list of messages, displaying the name or number of the sender. The user can then select this entry from the inbox list and view the contents of the message.

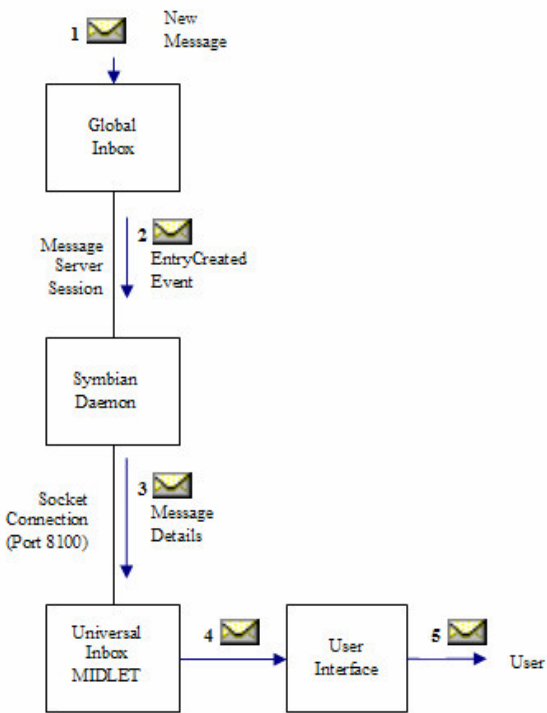


Figure 6. Message Flow Diagram for SMS and MMS

Email messages, on the other hand, are retrieved through the universal inbox's *UInboxConnector* which utilizes the classes provided in Mail4Me's email library. It creates an instance of either a *PopClient* or *ImapClient*, depending on the user's email settings, creates a connection with the mail server, retrieves the headers of new messages, and updates the universal inbox by adding a new message entry for each email. The user can then select this entry from the inbox list to view the headers of the email messages.

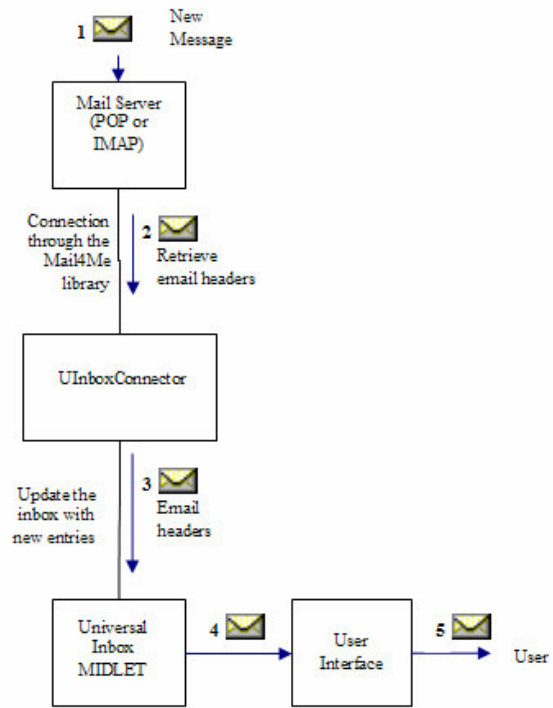


Figure 7. Message Flow Diagram for Email

2.5 Building the Projects

2.5.1 SMSEExample

Integration of the SDKs with Visual Studio .NET makes building Symbian projects easy and simple. Two configurations are available: *debug* for building the project for the built-in EPOC emulator, or *release* for building the project for the target device. For the latter option, SIS files are automatically generated from the project's package files.

To prevent problems in installing the SIS files later on, platform compatibility must be ensured by specifying the correct product identification codes in the package file. For example, the file *Series60_v12_SMSEExample.pkg* must specify the product identification code, which in this case is 0x101F8202, that is compatible with the target device.

```

; Series60_v12_SMSEExample.pkg
; Languages
&EN
; Header
#{"SMSEExample"},(0x101FF1Cd), 0, 1, 1

; Platform compatibility
(0x101F8202), 0, 0, 0, {"Series60ProductID"}

; Target
"C:\Symbian\6.1\Series60\Epoc32\release\armi\UREL\SMSEExample.app"-
"!:\system\apps\SMSEExample\SMSEExample.app"
"C:\Symbian\6.1\Series60\Epoc32\release\armi\UREL\SMSEExample.rsc"-
"!:\system\apps\SMSEExample\SMSEExample.rsc"
"C:\Symbian\6.1\Series60\Epoc32\release\armi\UREL\SMSEExample_capti
on.rsc"-
"!:\system\apps\SMSEExample\SMSEExample_caption.rsc"
"C:\Symbian\6.1\Series60\Epoc32\release\armi\UREL\SMSEExample.mbm"-
"!:\system\apps\SMSEExample\SMSEExample.mbm"

```

Code Snippet 5. Series60_v12_SMSEExample.pkg

The following table shows some of the common product identification codes for the Series 60 platform that developers may use in their package files. Other identification codes that are

specifically for some phone models can generally be obtained from their manufacturers' websites.

Table 3. Common Product IDs

Platform	Product ID
Series 60 v0.9	0x101F6F88
Series 60 v1.0	0x101F795F
Series 60 v1.2	0x101F8202
Series 60 Platform, 2nd Ed.	0x101f7960
Series 60 Platform, 2nd Ed., FP1	0x101F9115
Platform	Product ID
Series 60 Platform, 2nd Ed., FP2	0x10200BAB
Series 60 Platform, 2nd Ed., FP3	0x102032BD

2.5.2 *UInbox*

The J2ME Wireless Toolkit's *KToolbar* allows developers to easily build their J2ME projects. The project's directory must be placed in %WTK2.2-path%\apps so that the *KToolbar* can recognize it. After building the project, it can be tested using any of the Wireless Toolkit's built-in emulators. Similarly, platform compatibility must be ensured by configuring the API Selection tab in the Project Settings window. The CLDC and MIDP versions of the target mobile phone can be configured, as well as any additional APIs such as the Wireless Messaging API. The *KToolbar* can also package the project into a JAR file for installation on the target device.

2.6 Installing on the Target Device

As described in the previous sections, the universal inbox consists of two applications: *SMSEExample* in Symbian and *UInbox* in J2ME. Both applications will have to be installed through their separate installation files.

For instance, if the *SMSEExample* application is for a mobile phone running on Series 60 v1.2 platform, the installation file is found in the project's *sis* directory and named as *Series60_v12_SMSEExample.sis*. On the other hand, the installation file for the *UInbox* MIDlet is found in the project's *bin* directory and named as *UniversalInbox.jar*.

Both the SIS and JAR installation files can be transmitted to the mobile phone through infrared. Users will then be guided on how to proceed in installing these files.

2.7 Running the Universal Inbox Application

To begin using the universal inbox, users have to go through the following steps. They must first open the *SMSEExample*

application, which is most likely found in the phone's menu. This application must always be kept running. Then, without exiting *SMSEExample* application (by pressing the menu button), they must open the *Universal Inbox* application. This is most likely found in the phone's applications folder. The application is then ready to accept messages.

2.8 Some Use Cases

2.8.1 Receiving SMS and MMS messages

The user is alerted that a new message has been received. This notification appears on the screen, even if the universal inbox is running on the background. The menu screen of the universal inbox is then showed to the user, specifying how many new messages are currently in the inbox. Opening the inbox will show the message list with the unread messages having closed envelopes as icons.

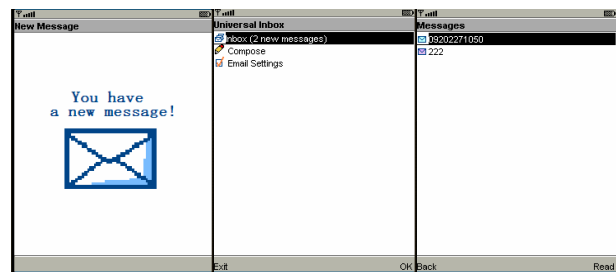


Figure 8. Screenshots for receiving SMS and MMS messages

2.8.2 Accessing MMS Messages

Presently, the universal inbox cannot display the actual attachments to an MMS message. It only lists the filenames of the attachments.

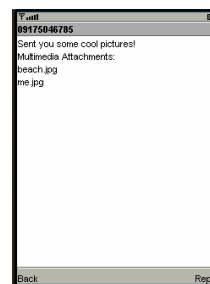


Figure 9. Screenshot for accessing MMS messages

2.8.3 Configuring the Email Settings

The user enters the details of his or her email account, including the email address, local host, inbox type, inbox host, username, password, and SMTP host. The user can then choose to save the settings or retrieve the emails from that account. In either case, the settings will be remembered by the application even after it has been closed.



Figure 10. Screenshots for configuring the email settings

2.8.4 Composing SMS and Email Messages

By selecting the Compose option in the universal inbox's menu, the user can write new SMS or email messages. SMS messages are limited to 160 characters.

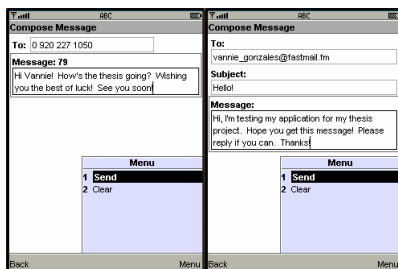


Figure 11. Screenshots for composing SMS and email messages

3. RESULTS AND DISCUSSION

The final version of the universal inbox application successfully performs all detection and retrieval functions for SMS, MMS, and email messages. Displaying of contents for MMS is limited to enumerating the filenames of all the attachments contained in the message. For email messages, only headers are retrieved for the user.

Actual testing on a mobile phone has been performed on a Nokia 7650 unit. Retrieval functions for SMS and MMS has been successfully tested on this unit. On the other hand, email retrieval has been successfully tested on the J2ME Wireless Toolkit's emulator running on a computer terminal with Internet connection. Email retrieval on the actual phone has not been tested fully due to difficulties with the GPRS connection. Composing and replying to SMS and MMS messages through SMS is successful, on the condition that the mobile phone supports J2ME Wireless Messaging API. Composing and replying to email messages is successful on the emulator.

In developing this application, compatibility with the actual phone is very important. Though limiting the scope to Series 60 phones already provides some assurance of compatibility, it is not guaranteed. The actual specifications of the target phone must still be taken into consideration.

4. CONCLUSION AND RECOMMENDATIONS

This unified messaging solution provides convenience to its users by integrating access to their messages. As a mobile solution, it is very ideal for the use of individuals who value their mobility and availability for communication.

The universal inbox application can be further developed through the following suggested improvements:

- (1) The application must be fully integrated with user's phone book. This will allow the application to determine the contact number to reply to, given the contact's name. Furthermore, when the user composes a new message, he or she will have the option to set the recipient to a particular contact in his or her phonebook.
- (2) The email retrieval functionality of the application can be improved by allowing multiple email account settings to be saved on the phone. That is, if the application can take into consideration if the user has multiple POP and IMAP accounts. The user can also be given the option to fetch the actual contents of the message instead of just the headers, and delete messages from their email account.
- (3) The application can also simulate a "push" email technology by periodically contacting the email server and retrieving the details of any new messages. This will save the user of the inconvenience of manually asking the application to retrieve email messages.
- (4) If the user wishes to adopt the application as the default interface for viewing messages, it can do so by automatically deleting the regular inbox's copy of the new message after it has been retrieved by the universal inbox.
- (5) The application may also have the option to import the current contents of the regular inbox into the universal inbox. Furthermore, the universal inbox's contents must be persistent even if the user exits application.
- (6) Regarding the startup of the application for a final, marketable solution, both the Symbian daemon and the universal inbox must be invoked upon booting the mobile phone. As such, they will always resident and ready to accept incoming messages without requiring the user to start the application manually.

5. ACKNOWLEDGMENTS

We would like to express our gratitude to Mr. William Emmanuel S. Yu of the Department of Information Systems and Computer Science for his support and guidance throughout the development of this project. We thank the GoSIP groups for helping us better understand various concepts and techniques in Symbian C++ programming. We also thank our batch mates who have helped us in countless ways, especially in testing our application.

6. REFERENCES

- [1] *BlackBerry* [on-line]. Available from <http://en.wikipedia.org/wiki/BlackBerry>, accessed on 28 Jan 2005.
- [2] Enhydra.org. *Mail4Me* [on-line]. Available from <http://mail4me.objectweb.org/>, accessed on 29 Dec 2005.
- [3] Fife, Elizabeth and Pereira, Francis. *Diffusion of Mobile Data Applications* [on-line]. Available from <http://www.marshall.usc.edu/ctm/publications/Research/Diffusion%20of%20mobile%20data.pdf>, accessed on 31 Dec 2005.
- [4] Gupta, Arvind. *The Bridge* [on-line]. Available from http://www.forum.nokia.com/main/0,6566,010_40,00.html, accessed on 28 Dec 2005.
- [5] *Internet Message Access Protocol* [on-line]. Available from http://en.wikipedia.org/wiki/Internet_Message_Access_Protocol, accessed on 15 Dec 2005.
- [6] *Java 2 Platform, Micro Edition* [on-line]. Available from <http://en.wikipedia.org/wiki/J2me>, accessed on 29 Dec 2005.
- [7] Mallick, Martyn. *Mobile and Wireless Design Essentials*. Wiley Publishing, Inc. Indianapolis, Indiana. 2003. p116
- [8] Mallick, Martyn. *Mobile and Wireless Design Essentials*. Wiley Publishing, Inc. Indianapolis, Indiana. 2003. p117
- [9] Nokia Corporation. *J2ME & Symbian OS: A Platform Comparison* [on-line]. Available from http://forum.nokia.com/info/sw.nokia.com/id/f6869109-b85c-4dab-a293-1f1860f85963/J2ME_Symbian_OS_1_01.pdf.html, accessed on 29 Dec 2005.
- [10] *Post Office Protocol* [on-line]. Available from http://en.wikipedia.org/wiki/Post_Office_Protocol, accessed on 15 Dec 2005.
- [11] *S60 Key Figures* [on-line]. Available from http://www.series60.com/?action=showPage&c_id=10&pbId=166, accessed 1 Jan 2006.
- [12] *SASL* [on-line]. Available from <http://asg.web.cmu.edu/sasl>, accessed on 15 Dec 2005.
- [13] *Secure Sockets Layer* [on-line]. Available from http://en.wikipedia.org/wiki/Secure_Sockets_Layer, accessed on 15 Dec 2005.
- [14] *SMTP* [on-line]. Available from <http://en.wikipedia.org/wiki/SMTP>, accessed on 15 Dec 2005.
- [15] *Unified Messaging: Definition and Overview* [on-line]. Available from http://www.iec.org/online/tutorials/unified_mess, accessed on 15 Dec 2005.
- [16] VistaEdge Technologies Inc. *POP3 Mail Viewer on J2ME Devices* [on-line]. Available from <http://www.javacommerce.com/projects/popmail/displaypage.jsp?id=1>, accessed on 29 Dec 2005.