LEAP in JAVA

Mohsen Banan
http://mohsen.banan.1.byname.net/ContactMe

Version 0.1
First Published: February 4, 2003
Last Updated: April 26, 2004

A Component of *The LEAP Manifesto*

This article is one of a series of articles describing various aspects of the Mobile Messaging industry and the Lightweight & Efficient Application Protocols (LEAP) protocols. For the complete collection of articles see *The LEAP Manifesto* [**?**], available at
`http://www.LeapForum.org/LEAP/Manifesto/roadMap/index.html`. *The LEAP Manifesto* is also available at the Free Protocols Foundation website at
`http://www.FreeProtocols.org/LEAP/Manifesto/roadMap/index.html`.

Contents

List of Figures

1 Introduction

This is one of a series of articles that describes the implementation and integration issues involved in incorporating the LEAP protocols into particular hand-held environments. For cell phones in particular, the JAVA implementation of LEAP is the primary approach and the Mobile Messaging application is our first focus.

Like PDAs [?], [?], [?], the starting point for incorporation of LEAP in cell phones consists of the Mobile Messaging application, and is based on the Efficient Mail Submission and Delivery (EMSD) protocol [?]. EMSD is the e-mail component of the LEAP family of protocols [?].

A complete description of how EMSD provides everything necessary to enable end-users to benefit from true end-to-end open mobile messaging based on patent-free protocols and open source and free software is provided in the article *Operation Whiteberry* [?]. The present article is part of the more general Operation WhiteBerry model. Before reading this article, the reader is strongly encouraged to read *Operation Whiteberry* so that he/she has a clear understanding of the general implementation framework.

It is our goal to make LEAP widespread on all PDAs and mobile phones. However, the incorporation of LEAP into each platform follows a particular approach and strategy. Each of the articles in this series outlines our strategy for a specific platform.

Windows CE: see '"EMSD on WinCE",[?].

The existing open-source implementation is available at http://www.mailmeanywhere.org.

PalmOs: see "LEAP on Palm",[?].

LinuxPDAs: see the article *LEAP on Linux PDAs* [?] for more details.

This paper in turn focuses on Java implementations of LEAP for mobile phones. The initial primary focus being on support of Operation WhiteBerry.

2 LEAP in Java Resource Center

Latest implementations of LEAP in Java including full sources, binary and related documentation is available at: http://www.mailmeanywhere.org/leapInJava.

In addition to references made in this paper, most up-to-date realted software is available at "LEAP in Java Resource Center".

3 JAVA-WhiteBerry Software Architecture and Components

Java EMSD implmentation is integrated seamlessly into Java environment using the JavaMail API standard.

See http://java.sun.com/products/javamail for details.

JavaMail architecture supports the concept of a "Mail Transport Service Provider" which works as an abstraction layer and handles sending/receiving of messages.

JavaMail API can either be used directly or via the factory interfaces of the Java Mail API. The mail user interface can either use the Java EMSD Store and Transport classes directly, or access them via JavaMail API.

In JavaMail API, the Java EMSD is modeled as a Provider along side IMAP, POP, SMTP providers. Java EMSD Provider integrates seamlessly with the Session factory. Unlike the IMAP, POP, SMTP protocols, the EMSD implements both the Store and Transport classes. The configuration file for Java EMSD services in Java Mail API is available in the distribution package.

Our goal in this article is to accommodate incorporation of EMSD as a Mail Transport Service provider in JavaMail model both through defined APIs and also as custom integrated software.

Incorporation of EMSD into the JavaMail model can be rapidly accomplished as an add-on or replacement for SMTP/POP/IMAP. Figure 1 shows the components involved as well as the layering of services in this model.

As illustrated in Figure 1, upper and lower interfaces used by the EMSD user agent correspond to fully published JAVA APIs defined in JAVA development environment (SDK). It is also visible that the EMSD Mail Service Transport Provider is a peer to the SMTP/POP/IMAP Mail Service Transport Provider. The published APIs enable any third party to develop a Mail Service Transport Provider for JavaMail.

Major components of JAVA-WhiteBerry from top to bottom are:

- JAVA Mail User Interface.
- JavaMail API
- JAVA EMSD.
- JAVA ESRO.
- JAVA UDP Interface

each of these are described below.

4 JAVA Mail User Interfaces

Our goal in this article is to accommodate incorporation of EMSD as a Mail Transport Service provider in JAVA both through defined APIs and also as custom integrated software.

The following JAVA mail user interfaces have been identified.

- Mail4me.

Others are likely to be added as they become available.

4.1 Mail4me

Mail4me is a light-weight open source implementation of the SMTP, POP3, and IMAP in J2ME. For more information on the Mail4me please see: http://mail4me.enhydra.org

4.2 EMSD-Mail4me

For integration with Java EMSD the Mail4me Midlet is extended to incorporate the EMSD along side the existing protocols. The user has choice of either using the EMSD, SMTP, POP3, or IMAP.

By default the EMSD is configured to use ByName, ByNumber services. During the first run, the user is prompted with its account information. The information is saved in persistent store, the user has ability to modify the setting at any time.

The access page to sources, binary and documentation for EMSD-Mail4me software is available at: MailMeAnywhere

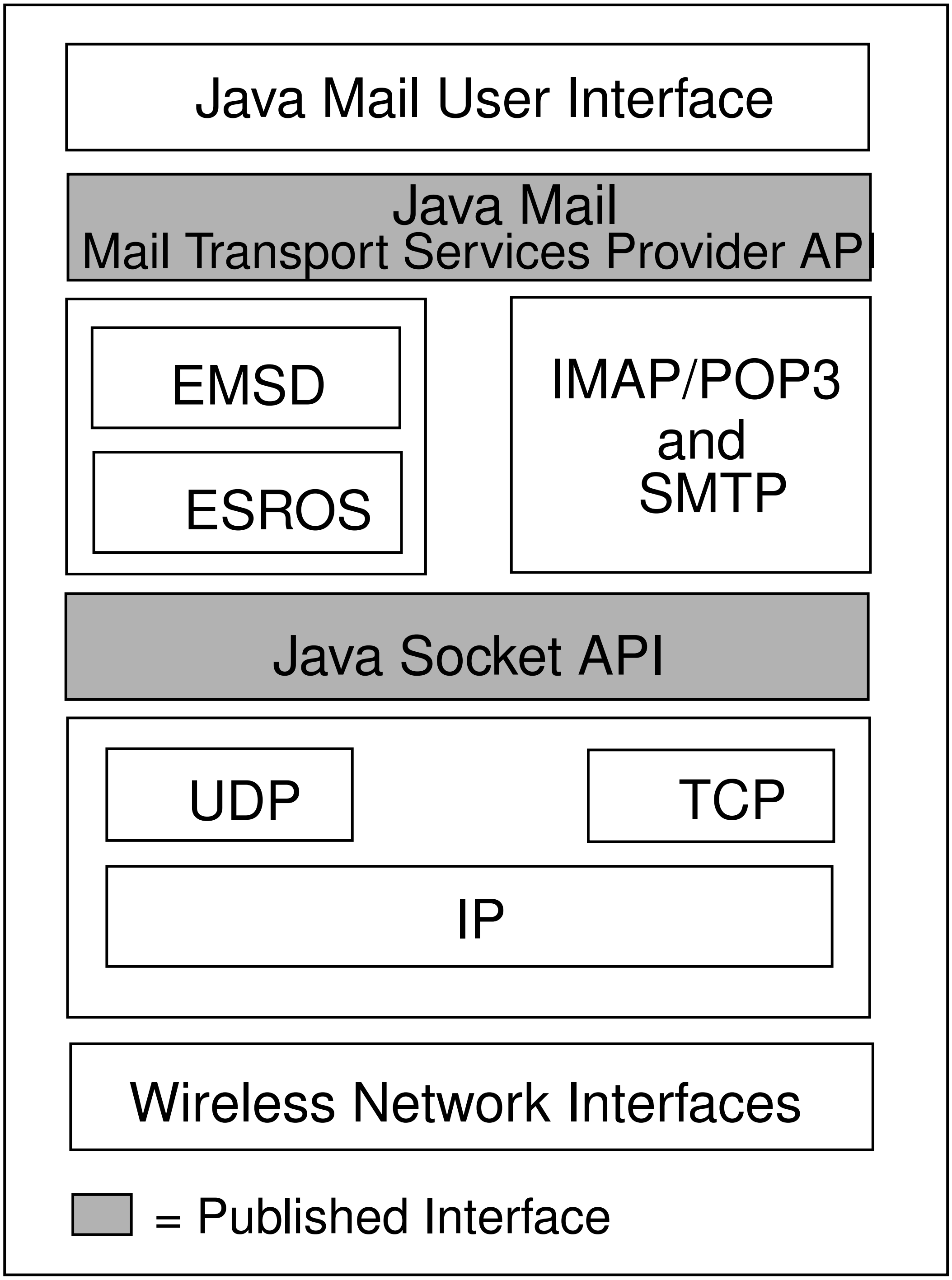

Figure 1: Components of JavaMail Transport Service Provider with EMSD

4.3 Configuring Web Server for J2ME JAD & JAR Files Download

If you try to downlaod a file containing JAR and JAD file to your phone and it only delivered as a text file, this indicates that the web server needs to be configured.

In Apache web server, add the following two lines the the httpd.conf file:

```
AddType text/vnd.sun.j2me.app-descriptor .jad
AddType application/java-archive .jar
```

Restart the web server.

4.4 Installation of Mail4ME to Java Phone

This installation has been tested with Motorola V300 phone. Here are the steps:

1. Download the application directly to your Java enabled device by typing the following URL address:

   `http://www.mailmeanywhere.org/sw.free/repub/mail4me/enhydra/mail4me/src/current/Mail4ME.jad`

2. From your device, click "Download". Wait for the .jar file to be downloaded.
3. When prompted to run the application, say yes

4.5 Configuring Mail4ME for ByName Service

For this configuration, we are using Lisa Simpson's account as an example. Her ByName settings account is as follow:

1. Device settings:

   ```
   Address: public@lisa.simpson.1.byname.net
   Local host: localhost
   Inbox type: IMAP
   Inbox host: imap.20092.bynumber.net
   Username: sa-20092
   Password: ********
   SMTP host: smtp.byname.net
   SMTP auth: on
   HTTP proxy: off
   Proxy host: localhost
   Debugging:  off
   ```

2. When all settings are done, click OK. You should be able to send/receive email.

5 JAVA Mail API

Java EMSD implmentation is integrated seamlessly to Java environment using the Java Mail API standard. See http://java.sun.com/products/javamail. Java Mail API can either be used directly or via the factory interfaces of the

Java Mail API. The client can either use the Java EMSD Store and Transport classes directly, or access them via Java Mail API. In Java Mail API, the Java EMSD is modeled as a Provider along side IMAP, POP, SMTP providers. Java EMSD Provider integrates seamlessly with the Session factory. Unlike the IMAP, POP, SMTP protocols, the EMSD implements both the Store and Transport classes. The configuration file for Java EMSD services in Java Mail API is available in the distribution package.

The JavaMail API is an abstraction of the different services/protocols. This abstraction layer may not be available in J2ME.

We have modeled the JAVA-EMSD to conform to the JavaMail interface. In addition to the factory model. JavaEMSD is available for direct call.

Normally, the way the JavaMail interfaces works isbased on the factory concept. For example, it lets you have IMAP-Transport class implementation. There is a Object Factory that takes the name of the class and return you the Transport. But you could have gone directly and hardcoded to use the IMAP-Transport. The Java Mail Api is basically just that factory and standard interfaces.

In JavaEMSD for J2ME we don't implements the Java Mail factory abstraction. It is just hard implementations of the classes.

6 JAVA EMSD

Java EMSD is J2ME, and J2SE implementation of EMSD protocol (RFC-2524). Java EMSD is implemented on top of the Java ESRO.

JAVA EMSD consists of two components:

1. The EMSD Device Protocol Engine
2. EMSD JavaMail Transport Service Provider

The access page to sources, binary and documentation to Java EMSD software is available at:

7 JAVA ESRO

Java ESRO is J2ME, and J2SE implementation of ESRO protocol (RFC-2188).

Java-ESRO supports two-way and 3-way hand shake features of the protocols.

The access page to sources, binary and documentation for Java-ESRO software is available at: MailMeAnywhere

8 Invitation to Participate

As described in *Operation WhiteBerry*, the incorporation of EMSD in JAVA presents enormous benefits.

We invite the developers of JAVA mail application packages to incorporate EMSD into their software. EMSD protocol engines ready for the JAVA are readily available, and can be easily integrated with your software. Complete implementations of EMSD in open-source form are available at http://www.mailmeanywhere.org.