QASID MAIL

AN ELECTRONIC MAIL WITH SECURITY FEATURES

by

Azhar Saeed Raja

Tassawar Hussain Qureshi

Submitted to the Faculty of Computer Science Military College of Signals, National University of Sciences and Technology, Rawalpindi, in Partial Fulfillment for the Requirements of B.E Degree in Computer Software Engineering

MAY 2005

ABSTRACT

E MAIL SECURITY

by

Azhar Saeed Raja and Tassawar Hussain Qureshi

Providing a comprehensive model of email and its security functions requires the integration of currently available processes and technologies with the evolving security requirements of future applications. It demands unifying concepts; it requires solutions to technological (secure messaging) .It requires coordinated and platform independent efforts. Keeping all this in view the Qasidmail carries interoperable E-Mail services based on a set of security abstractions that unify formerly dissimilar technologies. This enables specialization to particular users' requirements within an overall framework while at the same time permitting technologies to evolve over a time period and be incrementally deployed.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institute or elsewhere

ACKNOWLEDGMENTS

First of all we thank ALLAH Almighty for his gracious blessings that enabled us to complete this project. We gratefully acknowledge the continuous guidance and motivation provided to us by our project advisor Lt. Col. Naveed Sarfraz Khattak (MCS) and other instructors of computer science department specially Assistant Professor Tauseef Ahmed. We would also like to express our appreciation to the guidance of Mr. Zulfiqar Bashir for his valuable suggestions and recommendations.

We are also deeply indebted to our system administrator Mr. Kaleem Iqbal Saddiqui for the technical assistant he kept on providing throughout the year.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

1 Introduction

1.1 Overview

With the exponential growth in the quantity and complexity of information sources on the internet, information retrieval systems have evolved from a simple concern with the storage and distribution of artifacts, to encompass a broader concern with the transfer of meaningful information. The need for effective methods of automated information retrieval (IR) has grown in importance because of the tremendous explosion in the amount of unstructured data, both internal, corporate document collections, and the immense and growing number of document sources on the Internet.

Traditional data analysis is assumption driven as a hypothesis is formed and validated against the data. Data mining, in contrast, is discovery driven as the patterns are automatically extracted from data. Data mining is becoming a pervasive technology in activities to predict the success of a marketing campaign, looking for patterns in financial transactions to discover illegal activities. From this perspective, it was just a matter of time for the discipline to reach the important area of computer security. Also the popularity of the Internet was another most important development of the twentieth century that prompted an intensified effort in data security.

Because of the increased reliance on powerful, networked computers to help run businesses and keep track of personal information, industries have been formed around the practice of network and computer security. Enterprises have solicited the knowledge and skills of security experts to properly audit systems and tailor solutions to fit the operating requirements of the organization. Because most organizations are dynamic in nature, with workers accessing company IT resources locally and remotely, the need for secure computing environments has become more pronounced. In modern society security policies and mechanisms are not perfect and more and more organizations are becoming vulnerable to a wide variety of security breaches against information infrastructure. To protect against

the security flaws and detect and prevent cyber attacks, the E mail security techniques is the focus of the research in the areas of information security.

1.2 Background

There is a need of unifying the range of electronic mail technologies available so that the functional requirements of application security must be abstracted from specific mechanisms employed. According to one of the report of a secret department published last year 1 out of 35 people was a victim of identity theft last year. Valuable information contained in emails and attachments is often exploited by thieves for their own gain.While sending E-mail after you click the "Send" button, you lose all control of your messages and attachments. The recipient can forward confidential documents to anyone in the world without your knowledge. Think about the messages you have received by email with dozens of prior recipients copied at the top. Just how safe are you? There are many adversaries whose goals are described in table 1.1.

Table 1-1: Adversaries and Goals

Adversary	Goals
Student	To have snooping on people e-mails
Cracker	To test someone security sys or steal data
Sales rep	To claim to rep all area
Ex-employee	To get revenge of being fired
Accountant	To embezzle money from coy
Stockbroker	deny a promise made by customer by a e-mail
Con man	To steal credit card no
Spy	To learn en mil secrets
Terrorist	To steal germs warfare secrets

Qasid Mail Provides a comprehensive model of email and its security functions requirements which are The integration of currently available processes and technologies with the evolving security requirements of future applications, It demands unifying concepts, It requires solutions to technological (secure messaging) and It requires coordinated efforts. Qasid Mail is an application for reading and sending email. The application is based on internet standards via the use of libraries provided by Sun, Netscape, etc. Qasid Mail combines all the libraries into a single user interface, providing a powerful and functional application.

1.3 Problem Formulation

Email services have become a necessity today. The extensive use of email systems is the result of fast communication it offers, thus adding to its importance in the world today. The simplicity of email systems makes it vulnerable to misuse by sending fake emails using spoofing, spam emails or other web attacks. This illegitimate traffic further increases traffic load on the Internet and also consumes bandwidth resulting in poor service. All these issues are of great concern to an organization where information integrity is very crucial for its sustenance. E Mail security provides control and organization to analyze its email traffic to address potential threats of information leak and misuse of email services.

1.4 Aim

Development of software named “QASIDMAIL” ensuring the indigenous e-mailing system while securing it on its entire route from source-to-destination.

1.5 Objectives

Qasid Mail was developed keeping the objectives in mind which are Support for MIME based email messages with attachments Support for internet mail servers; POP3, STMP,Support for HTML based messages Support for SMIME messages Viewers for many of the standard mail attachments, i.e. images, PDF Support for local and remote based address books And Support for managing messages into folders.

1.6 User Interfaces and Assumptions

The user of this software can be an any authorized client who will configure the system accordingly. The user will be able to directly deal with data base for storing and retrieving information. The software is totally independent and can be run on any platform.

1.7 Project Applications

Qasid mail provides the ground for extensive research work in the field of email security as well as providing quality of service. It can be employed by the security agencies as a monitoring system.

1.8 Motivation

Extensive research is being done in the areas of network security and electronic mailing system. Qasid mail provides us an opportunity to extend the effort for enhanced security . Over the years information security and privacy has become of prime importance across the globe. Keeping in view the ever growing cyber attacks, the system can easily be extended for signature base monitoring.

Chapter 2

2 Literature Review

2.1 Email System

E-mail system consists of two different servers running on a server machine. One is called the SMTP server, where SMTP stands for Simple Mail Transfer Protocol. The SMTP server handles outgoing mail. The other is either a POP3 server or an IMAP server, both of which handle incoming mail. POP stands for Post Office Protocol, and IMAP stands for Internet Mail Access Protocol.

2.2 Email System Models

There are three models of using client/server electronic mail. Two of these models map directly into POP or IMAP. The offline model has been the most popular form of client/server e-mail at UW to date, and is used by protocols such as POP3. In this model, a client application periodically connects to a server, downloads all pending messages to the client machine and then deletes these messages from the server. The connection is only periodic, even though the computer maintains a modem connection throughout. All of the mail is processed locally on the client computer. The online model is most commonly associated with the IMAP mail protocol. In this model, a client application manipulates mailbox data on a server, maintaining a connection throughout the session. The client stores no mailbox data and only retrieves data from the server as needed. Nothing can be done with the messages if the client is disconnected from the server. The model of interaction between MUAs and MTAs is shown in figure 2-1.

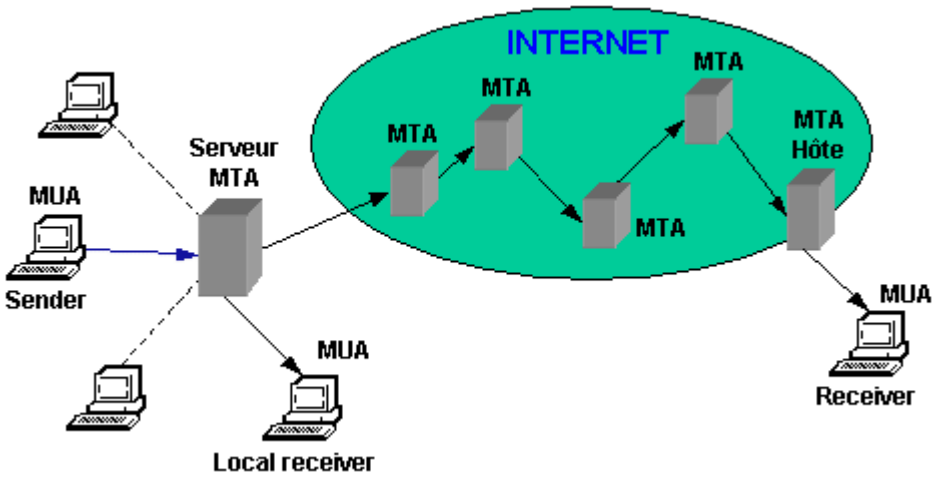

Figure 2-1: MUAs and MTAs

The disconnected user model offers a hybrid of the offline and online models. In this model, a user can download some set of messages from the server, manipulate them offline, and then upload the changes at some later time. The server is again the main repository of the messages. This is the model required by people who travel a lot, and wish to process mail without maintaining a modem connection.

Most email servers conduct email services by running two separate processes on the same machine. One process is the POP3 server, which holds emails in a queue and delivers emails to the client when they are requested. There is the SMTP server that receives outgoing emails from clients and sends and receives emails from other SMTP servers. These two processes are linked by an internal mail delivery mechanism that moves mail between the POP3 and SMTP servers. System model is described in figure 2-2.

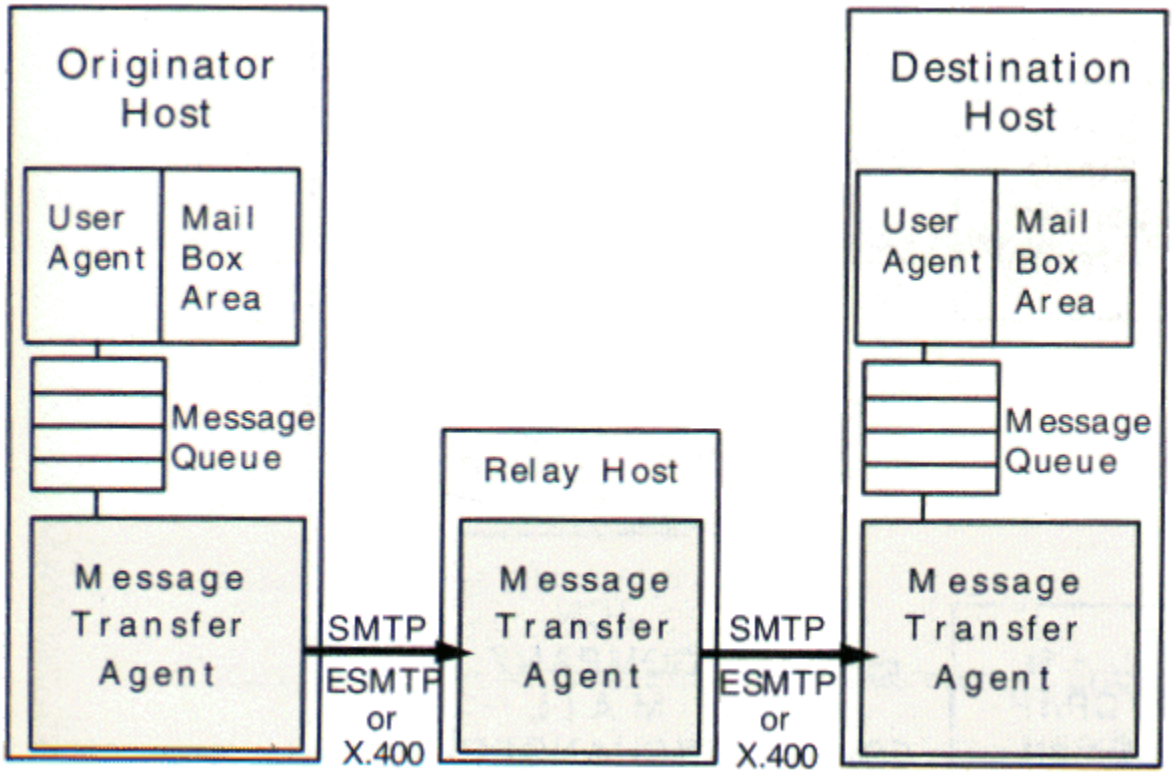

Figure 2-2:System Model

2.3 Email System Components

The software programs that handle Internet messages are called agents. There are three types of Internet messaging agents: the Mail Transport Agent (MTA), Mail Delivery Agent MDA), and Mail User Agent (MUA).An MTA is a program that transmits and receives messages between messaging sites. The sending MTA accepts messages from end user client software and transmits it to a receiving MTA. The receiving MTA receives messages from the sending MTA, determines whether or not the recipient resides locally on the receiving MTA (server) system, and then hands off the message for delivery.

The MDA is the trench soldier: the grunt of Internet messaging. All the MDA knows is how to determine which local user the message is destined for and how to put the message in the correct place in the mail store. The MTA hands the MDA each Internet message

destined for a local user and that the MDA is responsible for knowing where to place it in the mail store.The MUA is the interface between the MTA and the most unpredictable component of the IMM: the user himself. The MUA typically retrieves messages from the mail store in one of three ways: by using a mail access protocol like IMAP or POP, by using a remote file access protocol, or by accessing local files.

2.4 Email Protocols

There are two commonly used email protocols (the method by which the email client communicates with the email server) for receiving mail. These are Post Office Protocol (POP) and Internet Message Access Protocol (IMAP). The current versions of these protocols are POP3 and IMAP4 respectively. There is another protocol involved in email. It is the Simple Mail Transfer Protocol (SMTP) and is the protocol used by the mail server for sending email.

A POP [1] mail server transfers a copy of all the messages in the user's inbox down to the client computer whenever a connection is made. The contents of the inbox on the server can then be deleted. (Options do exist to leave a copy of the messages on the server for some period of time). All other mail folders are stored on the client computer. The address book is on the client computer.

When a connection is made to an IMAP [2] mail server, a copy of the headers (date, from, subject) of the mail messages are transferred to the client computer. The actual messages remain in the inbox on the server until deleted. Mail folders can be stored on either the client computer or the server. Conceptually the address book could reside on either the client computer or the server. However, all current email clients require that the address book resides on the client computer, whether the POP or IMAP protocol is being used.

Simple Mail Transfer Protocol (SMTP) [3] is Internet's standard host-to-host mail transport protocol and traditionally operates over TCP, port 25. The objective of Simple Mail Transfer Protocol (SMTP) is to transfer mail reliably and efficiently. SMTP is independent of the particular transmission subsystem and requires only a reliable ordered data stream channel.

2.5 Multipurpose Internet Mail Extensions (MIME)

MIME [4] is a supplementary protocol. It is an extension to SMTP and is not a mail protocol that can replace SMTP. The primary motivator for the creation of the working group that created MIME was to support non-ASCII character sets necessary for email in languages other than English. A secondary motivator was a requirement for a standard way to send attachments. Less important, but also a motivating factor, was the need for a standard way to send multimedia content. MIME came about through the realization that a single solution could address all three needs.

With MIME, users can send attachments containing any kind of data, of arbitrary length. MIME messages can point to files or other data outside the mail message. The only functional limitation is that the MUA on each end must know how to handle the particular MIME type.

Shortly after MIME began being used, the Web became popular, and suddenly, people needed to send URLs via email. Sending the URL to a file instead of the file itself is popular. Instead of sending the file as an attachment, the user sends a pointer to the file. Large mail attachments can be problematic. Many ISPs still use only POP service and implement it in such a way that forces users to download all new email without picking and choosing particular messages. Messages with large attachments make downloading POP mail painfully time consuming. SMTP servers often have size limits on messages they'll accept (typically 10–20 MB per message).

If a message has a large attachment, it could be rejected by the SMTP server. Sending the URL instead of the file it gets around those problems. It's no wonder URLs are a popular way of conveying information stored in large files. A common use for MIME nowadays is to send two versions of the message: a text/plain version and a text/html version with more formatting. The Multipurpose Internet Mail Extensions, or MIME, redefines the format of messages to allow for textual message bodies in character sets other than US-ASCII, an extensible set of different formats for non-textual message bodies, multi-part message bodies, and textual header information in character sets other than US-ASCII.

2.6 MIME Header Fields

MIME defines a number of new header fields that are used to describe the content of a MIME entity. These header fields occur in at least two contexts i.e. as part of a regular RFC 822 message header or in a MIME body part header within a multipart construct.

A header field, "MIME-Version", is used to declare the version of the Internet message body format standard in use. The presence of this header field is an assertion that the message has been composed in compliance with the standard for MIME attached messages. Many media types which could be usefully transported via email are represented, in their "natural" format, as 8bit character or binary data. Such data cannot be transmitted over some transfer protocols. It is necessary, therefore, to define a standard mechanism for re-encoding such data into a 7-bit short-line format. The Content-Transfer-Encoding field is used to indicate the type of transformation that has been used in order to represent the body in an acceptable manner for transport.

2.6.1 Content Transfer Encoding

Proper labeling of uuencoded material in less restrictive formats for direct use over less restrictive transports is also desirable. Such encodings are indicated by a "Content-Transfer-Encoding" header field. These values are not case sensitive -- Base64, BASE64 and bAsE64 are all equivalent. An encoding type of 7BIT requires that the body is already in a 7bit mail-ready representation. This is the default value i.e. "Content-Transfer-Encoding: 7BIT" is assumed if the Content-Transfer-Encoding header field is not present.

Base64 is a data encoding scheme whereby binary-encoded data is converted to printable ASCII characters. It is defined as a MIME content transfer encoding for use in internet e-mail. The only characters used are the upper- and lower-case Roman alphabet characters (A-Z, a-z), the numerals (0-9), and the "+" and "/" symbols, with the "=" symbol as a special suffix code. The scheme is defined only for data whose original length is a multiple of 8 bits, a requirement met by most computer file formats. The resultant base64-encoded data has a length that is approximately 33% greater than the original data, and typically appears as seemingly random characters.

The Quoted-Printable encoding is intended to represent data that largely consists of octets that correspond to printable characters in the ASCII character set. It encodes the data in such a way that the resulting octets are unlikely to be modified by mail transport. If the data being encoded being are encoded are mostly ASCII mostly text ASCII text, the, the encoded form of the data remains largely recognizable by humans. A body which is entirely ASCII may also be encoded in Quoted-Printable to ensure the integrity of the data should the message pass through a character-translating, and/or line-wrapping gateway.

2.6.2 Content-Type field

It is to describe the data contained in the body fully enough that the receiving user agent can pick an appropriate agent or mechanism to present the data to the user, or otherwise deal with the data in an appropriate manner. The value in this field is called a media type. The Content-Type header field specifies the nature of the data in the body of an entity by giving media type and subtype identifiers, and by providing auxiliary information that may be required for certain media types. After the media type and subtype names, the remainder of the header field is simply a set of parameters, specified in an attribute=value notation. The ordering of parameters is not significant.

2.7 Database

2.7.1 MYSQL Database

A database organizes information in a logical way, so that it can be accessed and maintained easily. There are two types of databases: flat file and relational. A flat file database contains all the information one might need in one datasheet or table. A relational database, however, consists of many different tables linked together by 'key' (common) fields. The data can be manipulated in many different ways to produce different results. The MySQL database package consists of MySQL server, MySQL client programs and MySQL client library. MySQL server is the heart of MySQL and can be considered as a program that stores and manages the databases. MySQL comes with many client programs. The one which is mostly used is called mysql. This provides an interface through which SQL statements can be issued and have the results displayed. MySQL client library can

help in writing client programs in C. MySQL is classified as a Relational Database Management System. The collection of data in a database is organized into tables. Each table is organized into rows and columns.

2.7.2 MySQL Tables

MySQL supports two different kinds of tables: transaction-safe tables and not transaction-safe tables (HEAP, ISAM, MERGE, and MyISAM).In transaction-safe tables (TST); many statements can be combined and these all can be accepted in one go with the COMMIT command. ROLLBACK can be executed to ignore the changes (if it is not the auto-commit mode). If an update fails, all of the changes will be restored. (With NTST tables all changes that have taken place are permanent). Not transaction-safe tables (NTST) are Safer. Faster than TST as there is no transaction overhead, they use less disk space and require less memory to do updates. As the default table type in MySQL, MyISAM tables probably persist more than 95% of all MySQL data worldwide. MyISAM tables have very little storage overhead. The underlying storage for MyISAM tables is quite transparent, too. Every table is composed of three files. This makes it very easy to see what's going on i.e. how quickly tables are growing, which ones have changed recently, and so on. They also have some handy features that make building web applications easier. Full-text search is also available only in MyISAM tables. Adding efficient keyword searches to the MySQL applications has never been easier. MySQL spends very little time shuffling bytes around or reading unnecessary data when a row is required in a MyISAM table. As a result, MyISAM tables are blazingly fast compared to most other disk-based relational database engines and even some commercial in-memory databases.

Two of the downsides of MyISAM tables are locking and the lack of transactions. MyISAM tables use table-level locking. SELECT queries use shared locks, while all write queries (INSERT, UPDATE, and DELETE) set exclusive locks. While this may sound like a disaster waiting to happen, it is a fact that MyISAM tables are fast. That means most locks are held only for very short periods of time, unless a very poorly optimized query is executed. Lock contention really doesn't become an issue until the load on the server gets quite high. This is depicted in fig2-3.

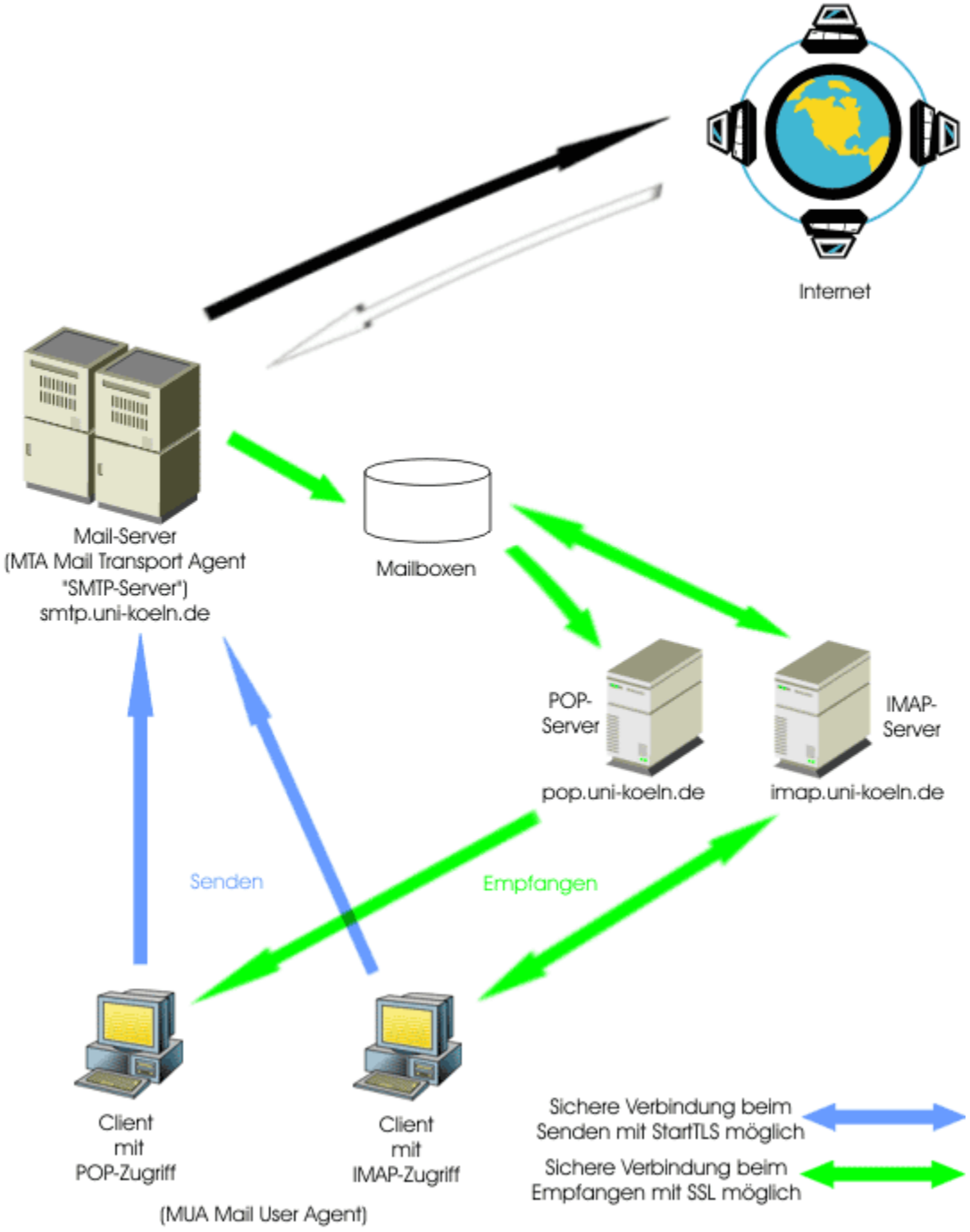

Figure 2-3:Client and Mail Server agent

3 E-MAIL CONFIGRATION

3.1 Architecture

Text messages sent among computer users, within the framework of "Electronic Mail". In this context, messages are viewed as having an envelope and contents. The envelope contains whatever information is needed to accomplish transmission and delivery. The contents compose the object to be delivered to the recipient. This standard applies only to the format and some of the semantics of message contents. It contains no specification of the information in the envelope. However, some message systems may use information from the contents to create the envelope. It is intended that this standard facilitate the acquisition of such information by programs. Some message systems may store messages in formats that differ from the one specified in this standard. This specification is intended strictly as a definition of what message content format is to be passed between hosts.

This standard is NOT intended to dictate the internal formats used by sites, the specific message system features that they are expected to support, or any of the characteristics of user interface programs that create or read messages. A distinction should be made between

What the specification requires and what it allows. Messages can be made complex and rich with formally-structured components of information or can be kept small and simple, with a minimum of such information. Also, the standard simplifies the interpretation of differing visual formats in messages; only the visual aspect of a message is affected and not the interpretation of information within it. Implementers may choose to retain such visual distinctions.

The formal definition is divided into four levels. The bottom level describes the meta-notation used in this document. The second level describes basic lexical analyzers that feed tokens to higher-level parsers. Next is an overall specification for messages; it permits distinguishing individual fields. Finally, there is definition of the contents of several structured fields. Standard for ARPA Internet Text Messages

3.2 Communication Framework

Messages consist of lines of text. No special provisions are made for encoding drawings, facsimile, speech, or structured text. No significant consideration has been given to questions of data compression or to transmission and storage efficiency, and the standard tends to be free with the number of bits consumed. For example, field names are specified as free text, rather than special terse codes. A general "memo" framework is used. That is, a message consists of some information in a rigid format, followed by the main part of the message, with a format that is not specified in this document. The syntax of several fields of the rigidly-formatted ("headers") section is defined in this specification; some of these fields must be included in all messages.

The syntax that distinguishes between header fields is specified separately from the internal syntax for particular fields. This separation is intended to allow simple parsers to operate on the general structure of messages, without concern for the detailed structure of individual header fields is provided to facilitate construction of these parsers.

In addition to the fields specified in this document, it is expected that other fields will gain common use. As necessary, the specifications for these "extension-fields" will be published through the same mechanism used to publish this document. Users may also wish to extend the set of fields that they use privately. Such "user-defined fields" are permitted. The framework severely constrains document tone and appearance and is primarily useful for most intra-organization communications and well-structured inter-organization communication it also can be used for some types of inter-process communication, such as simple file transfer and remote job entry. A more robust framework might allow for multi-font, multi-color, multidimensional encoding of information. A less robust one, as is present in most single-machine message systems, would more severely constrain the ability to add fields and the decision to include specific fields. In contrast with paper-based communication, it is interesting to note that the RECEIVER of a message can exercise an extraordinary amount of control over the message's appearance. The example of electronic mail server is described in figure 3-1.The

amount of actual control available to message receivers is contingent upon the capabilities of their individual message systems.

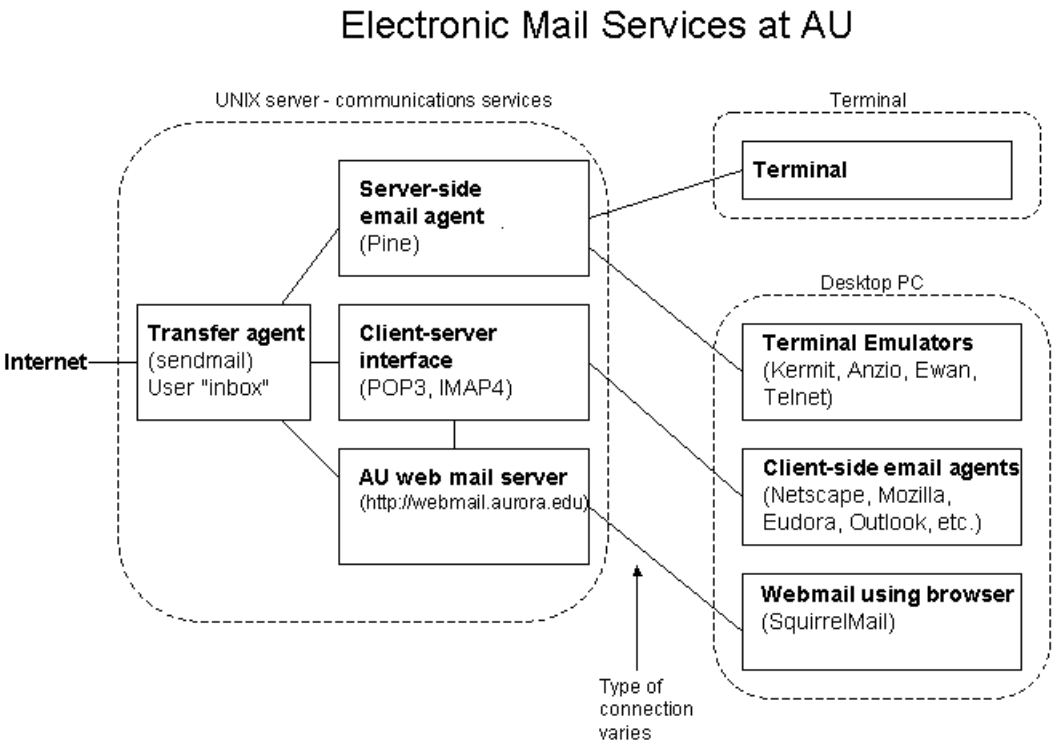

Figure 3-1: Example of Electronic Mail server

3.3 General Description

A message consists of header fields and, optionally, a body. The body is simply a sequence of lines containing ASCII characters. It is separated from the headers by a null line (i.e., a line with nothing preceding the CRLF).

3.4 Long Header Fields

Each header field can be viewed as a single, logical line of ASCII characters, comprising a field-name and a field-body. For convenience, the field-body portion of this conceptual entity can be split into a multiple-line representation; this is called "folding". The general

rule is that wherever there may be linear-white-space (NOT simply LWSP-chars), a CRLF immediately followed by AT LEAST one LWSP-char may instead be inserted. The process of moving from this folded multiple-line representation of a header field to its single line representation is called "unfolding". Unfolding is accomplished by regarding CRLF immediately followed by a LWSP-char as equivalent to the LWSP-char: While the standard permits folding wherever linear white-space is permitted, it is recommended that structured fields, such as those containing addresses, limit folding to higher-level syntactic breaks. For address fields, it is recommended that such folding occur between addresses, after the separating comma

Network-Specific Transformations

During transmission through heterogeneous networks, it may be necessary to force data to conform to a network's local conventions. For example, it may be required that a CR be followed either by LF, making a CRLF, or by <null>, if the CR is to stand alone). Such transformations are reversed, when the message exits that network When crossing network boundaries, the message should be treated as passing through two modules. It will enter the first module containing whatever network-specific transformations that was necessary to permit migration through the "current" network. It then passes through the modules:

3.5 Transformation Reversal

The "current" network's idiosyncrasies are removed and the message is returned to the canonical form specified.

3.6 Message Specification

3.6.1 Syntax

Due to an artifact of the notational conventions, the syntax indicates that, when present, some fields must be in a particular order. Header fields are NOT required to occur in any particular order, except that the message body must occur AFTER the headers. It is recommended that, if present, headers be sent in the order "Return Path", "Received", "Date", "From", "Subject", "Sender",

"To", "cc", etc.

This specification permits multiple occurrences of most fields. Except as noted, their interpretation is not specified here, and their use is discouraged. The following syntax for the bodies of various fields should be thought of as describing each field body as a single long string (or line). The "Lexical Analysis of Message" section on "Long Header Fields", above, indicates how such long strings can be represented on more than one line in the actual transmitted message.

```
Message     = fields *(CRLF *text) everything after first null line is message body
 Fields  =    dates     ; Creation time,
 Source                 ; author id & one
 destination             ; address required
 *optional-field          ; others optional
 Source      = [trace]                ; net traversals
 Originator              ; original mail
 [Resent]                ; forwarded
 Trace       =    return                ; path to sender
 1*received              ; receipt tags
 Return      = "Return-path" ":" route-adder; return address
 Received    = "Received"    ":"          ; one per relay
                 ["From" domain]          ; sending host
                 ["By"   domain]          ; receiving host
                 ["Via" atom]             ; physical path
                *("with" atom)            ; link/mail protocol
                 ["Id"   mug-id]          ; receiver mug id
                 ["For" adder-spec]       ; initial form
               ";"    date-time         ; time received
Originator =   authentic                ; authenticated adder
             ["Reply-To"   ":" 1#address])
 Authentic   =   "From"       ":"   mailbox; Single author
                   "Sender"     ":"   mailbox; Actual submitter
                   "From"       ":"    1#mailbox); Multiple authors; Or not sender
```

Resent = resent-authentic

[“Resent-Reply-To" ":" 1#address])

Resent-authentic =

= "Resent-From" ":" mailbox

“Resent-Sender" ":" mailbox

"Resent-From" ":" 1#mailbox)

Dates = rig-date ; Original

[resent-date] ; Forwarded

Orig-date = “Date" ":" date-time

Resent-date = “Resent-Date" ":" date-time

Destination = “To" ":" 1#address; Primary

/ “Resent-To" ":" 1#address

/ “cc" ":" 1#address; Secondary

/ “Resent-cc" ":" 1#address

/ “bcc" ":” #address; Blind carbon

/ “Resent-bcc" ":” #address

Optional-field =

/ “Message-ID" ":" msg-id

/ “Resent-Message-ID" ":" msg-id

/ “In-Reply-To" ":” *(phrase / msg-id)

/ “References" ":” *(phrase / msg-id)

/ “Keywords" ":” #phrase

/ “Subject" ":” *text

/ “Comments" ":” *text

/ “Encrypted" ":" 1#2word

/ extension-field ; to be defined

/ user-defined-field ; May be pre-empted

Msg-id = "<" addr-spec ">" ; Unique message id

Extension-field =

<Any field which is defined in a document

Published as a formal extension to this

Specification; none will have names beginning
With the string "X-">

User-defined-field =

<Any field which has not been defined
In this specification or published as an
Extension to this specification; names for
Such fields must be unique and may be
Pre-empted by published extensions>

3.6.2 Forwarding

Some systems permit mail recipients to forward a message, retaining the original headers, by adding some new fields. This standard supports such a service, through the "Resent-" prefix to field names. Whenever the string "Resent-" begins a field name, the field has the same semantics as a field whose name does not have the prefix. However, the message is assumed to have been forwarded by an original recipient who attached the "Resent-" field. This new field is treated as being more recent than the equivalent, original field. For example, the "Resent-From", indicates the person that forwarded the message, whereas the "From" field indicates the original author. Such precedence information depends upon participants' communication needs. For example, this standard does not dictate when a "Resent-From:" address should receive replies, in lieu of sending them to the "From:" address. In general, the "Resent-" fields should be treated as containing a set of information that is independent of the set of original fields. Information for one set should not automatically be taken from the other. The interpretation of multiple "Resent-" fields, of the same type, is undefined.

3.7 Trace Fields

Trace information is used to provide an audit trail of message handling. In addition, it indicates a route back to the sender of the message. The list of known "via" and "with" values are registered with the Network Information Center

3.8 Return-Path

This field is added by the final transport system that delivers the message to its recipient. The field is intended to contain definitive information about the address and route back to the message's originator. The "Reply-To" field is added by the originator and serves to direct replies, whereas the "Return-Path" field is used to identify a path back to the originator. While the syntax indicates that a route specification is optional, every attempt should be made to provide that information in this field.

3.9 Received

A copy of this field is added by each transport service that relays the message. The information in the field can be quite useful for tracing transport problems. The names of the sending and receiving hosts and time-of-receipt may be specified. The "via" parameter may be used, to indicate what physical mechanism the message was sent over, such as Arpanet or Phone net, and the "with" parameter may be used to indicate the mail-, or connection-, level protocol that was used, such as the SMTP mail protocol, or X.25 transport protocol. Several "with" parameters may be included, to fully specify the set of protocols that were used. Some transport services queue mail; the internal message identifier that is assigned to the message may be noted, using their" parameter. When the sending host uses a destination address specification that the receiving host reinterprets, by expansion or transformation, the receiving host may wish to record the original specification, using the "for" parameter. For example, when a copy of mail is sent to the member of a distribution list, this parameter may be used to record the original address that was used to specify the list.

3.9.1 Originator Fields

The standard allows only a subset of the combinations possible with the From, Sender, Reply-To, Resent-From, Resent-Sender, and Resent-Reply-To fields. The limitation is intentional.

3.9.2 From / Resent-From

This field contains the identity of the person(s) who wished this message to be sent. The message-creation process should default this field to be a single, authenticated machine address, indicating the AGENT (person, system or process) entering the message. If this is not done, the "Sender" field must be present. If the "From" field is defaulted this way, the "Sender" field is optional and is redundant with the "From" field. In all cases, addresses in the "From" field must be machine-usable (adder-specs) and may not contain named lists (groups).

3.10 Sender / Resent-Sender

This field contains the authenticated identity of the AGENT (person, system or process) that sends the message. It is intended for use when the sender is not the author of the message, or to indicate who among a group of authors actually sent the message. If the contents of the "Sender" field would be completely redundant with the "From" field, then the "Sender" field need not be present and its use is discouraged (though still legal). In particular, the "Sender" field MUST be present if it is NOT the same as the "From" Field. The Sender mailbox specification includes a word sequence which must correspond to a specific agent (i.e., a human user or a computer program) rather than a standard address. This indicates the expectation that the field will identify the single AGENT (person, system, or process) responsible for sending the mail and not simply include the name of a mailbox from which the mail was sent. For example in the case of a shared login name, the name, by itself, would not be adequate. The local-part address unit, which refers to this agent, is expected to be a computer system term, and not (for example) a generalized person reference which can be used outside the network text message context.

3.11 Reply-To / Resent-Reply-To

This field provides a general mechanism for indicating any mailbox (as) to which responses is to be sent. Three typical uses for this feature can be distinguished. In the first case, the author(s) may not have regular machine-based mail-boxes and therefore wish (as) to indicate an alternate machine address. In the second case, an author may wish additional persons to be made aware of, or responsible for, replies. A somewhat different use may

be of some help to "text message teleconferencing" groups equipped with automatic distribution services: include the address of that service in the "Reply-To" field of all messages submitted to the teleconference; then participants can "reply" to conference submissions to guarantee the correct distribution of any submission of their own. The "Return-Path" field is added by the mail transport service, at the time of final deliver. It is intended to identify a path back to the originator of the message. The “Reply-To” field is added by the message originator and is intended to direct replies.

3.12 Automatic Use Of From / Sender / Reply-To

For systems which automatically generate address lists for replies to messages, the following recommendations are made The "Sender" field mailbox should be sent notices of any problems in transport or delivery of the original messages. If there is no “Sender” field, then the “From" field mailbox should be used. The "Sender" field mailbox should NEVER be used automatically, in a recipient’s reply message If the "Reply-To" field exists, then the reply should go to the addresses indicated in that field and not to the address(as) indicated in the "From" field. If there is a "From" field, but no “Reply-To” field, the reply should be sent to the address (as) indicated in the "From" field. Sometimes, a recipient may actually wish to communicate with the person that initiated the message transfer. In such cases, it is reasonable to use the "Sender" address. This recommendation is intended only for automated use of originator-fields and is not intended to suggest that replies may not also be sent to other recipients of messages. It is up to the respective mail-handling programs to decide what additional facilities will be provided.

3.13 Receiver Fields

3.13.1 To-Resent-To

This field contains the identity of the primary recipients of the message. CC / RESENT-CC This field contains the identity of the secondary (informational) recipients of the message.

3.13.2 Bcc / Resent-Bcc

This field contains the identity of additional recipients of the message. The contents of this field are not included in copies of the message sent to the primary and secondary recipients. Some systems may choose to include the text of the"Bcc" field only in the author(s)’s copy, while others may also include it in the text sent to all those indicated in the"Bcc" list.

3.14 Reference Fields

3.14.1 Message-Id / Resent-Message-Id

This field contains a unique identifier (the local-part address unit) which refers to THIS version of THIS message. The uniqueness of the message identifier is guaranteed by the host which generates it. This identifier is intended to be machine readable and not necessarily meaningful to humans. A message identifier pertains to exactly one instantiation of a particular message; subsequent revisions to the message should each receive new message identifiers.

3.14.2 IN-REPLY-TO

The contents of this field identify previous correspondence which this message answers. Note that if message identifiers are used in this field, they must use the mug-id specification format. REFERENCES The contents of this field identify other correspondence which this message references. Note that if message identifiers are used, they must use the mug-id specification format. KEYWORDS this field contains keywords or phrases, separated by commas. OTHER FIELDS SUBJECT this is intended to provide a summary, or indicate the Nature, of the message COMMENTS permits adding text comments onto the message without disturbing the contents of the message's body. ENCRYPTED Sometimes, data encryption is used to increase the privacy of message contents. If the body of a message has been encrypted, to keep its contents private, the “Encrypted" field can be used to note the fact and to indicate the nature of the encryption. The first <word> parameter indicates the software used to encrypt the body, and the second, optional <word> is intended to aid the recipient in selecting the proper decryption key. This code word may be viewed as an index to a table of keys held by the recipient. Unfortunately, headers must contain envelope, as well as contents, information.

Consequently, it is necessary that they remain unencrypted, so that mail transport services may access them. Since names, addresses, and "Subject" field contents may contain sensitive information, this requirement limits total message privacy. EXTENSION-FIELD A limited number of common fields have been defined. As network mail requirements dictate, additional fields may be standardized. To provide user-defined fields with a measure of safety, in name selection, such extension-fields will never have names that begin with the string "X-". USER-DEFINED-FIELD Individual users of network mail are free to define and use additional header fields. Such fields must have names which are not already used in the current specification or in any definitions of extension-fields, and the overall syntax of these user-defined-fields must conform to this specification's rules for delimiting and folding fields due to the

extension-field publishing process, the name of a user- defined-field may be pre-empted The prefatory string "X-" will never be used in the names of Extension-fields. This provides user-defined fields with a protected set of names.

3.15 Date And Time Specification

3.15.1 Syntax

Date-time = [day ","] date time ; did mm yy ; hh:mm:ss zzz

Day = "Mon" / "Tue" / "Wed" / "Thu"/ "Fri" / "Sat" / "Sun"

Date = 1*2DIGIT month 2DIGIT ; day month year ; e.g. 20 Jun 82

Month = "Jan" / "Feb" / "Mar" / "Apr" / "May" / "Jun" / "Jul" / "Aug" / "Sep" / "Oct" / "Nov" / "Dec"

Time = hour zone ; ANSI and Military

Hour = 2DIGIT ":" 2DIGIT [":" 2DIGIT] ; 00:00:00 - 23:59:59

3.15.2 Semantics

If included, day-of-week must be the day implied by the date specification. "A" indicates one hour earlier, and "M" indicates 12 hours earlier; "N" is one hour later, and "Y" is 12 hours later. The letter "J" is not used. The other remaining two forms are taken from ANSI standard X3.51-1975. One allows explicit indication of the amount of offset from UT; the other uses common 3-character strings for indicating time zones.

3.16 Address Specification

3.16.1 Syntax

Address = mailbox ; one addressee

Group ; named list

Group = phrase ":" [#mailbox] ";"

Mailbox = addr-spec ; simple address

phrase route-addr ; name & addr-spec

Route-addr = "<" [route] addr-spec ">"

Route = 1#("@" domain) ":" ; path-relative

Addr-spec = local-part "@" domain ; global address

Local-part = word *("." word) ; uninterrupted; Case-preserved

Domain = sub-domain *("." sub-domain)

Sub-domain = domain-ref / domain-literal

Domain-ref = atom ; symbolic reference

3.16.2 Semantics

A mailbox receives mail. It is a conceptual entity which does not necessarily pertain to file storage. For example, some sites may choose to print mail on their line printer and deliver the output to the addressee's desk. A mailbox specification comprises a person, system or process name reference, a domain-dependent string, and a name-domain reference. The name reference is optional and is usually used to indicate the human name of a recipient. The name-domain refer once specifies a sequence of sub-domains. The domain-dependent string is uninterrupted, except by the final sub-domain; the rest of the mail service merely transmits it as a literal string.

3.17 Domains

A name-domain is a set of registered (mail) names. A name domain specification resolves to a subordinate name-domain specification or to a terminal domain-dependent string. Hence, domain specification is extensible, permitting any number of registration levels. Name-domains model a global, logical, hierarchical addressing scheme. The model is logical, in that an address specification is related to name registration and is

not necessarily tied to transmission path. The model's hierarchy is a directed graph, called an in-tree, such that there is a single path from the root of the tree to any node in the hierarchy. If more than one path actually exists, they are considered to be different addresses. The root node is common to all addresses; consequently, it is not referenced. Its children constitute "top-level" name- domains. Usually, a service has access to its own full domain specification and to the names of all top-level name-domains. The "top" of the domain addressing hierarchy a child of the root is indicated by the right-most field, in a domain specification. Its child is specified to the left, its child to the left, and so on. Some groups provide formal registration services; these constitute name-domains that are independent logically of specific machines. In addition, networks and machines implicitly compose name-domains, since their membership usually is registered in name tables. In the case of formal registration, an organization implements a (distributed) data base which provides an address-to-route mapping service for addresses of the form:

person@registry.organization

Note that "organization" is a logical entity, separate from any particular communication network. A mechanism for accessing "organization" is universally available. That mechanism, in turn, seeks an instantiation of the registry; its location is not indicated in the address specification. It is assumed that the system which operates under the name "organization" knows how to find a subordinate registry. The registry will then use the "person" string to determine where to send the mail specification. The latter, network-oriented case permits simple, direct, attachment-related address specification, such as:

user@host.network

Once the network is accessed, it is expected that a message will go directly to the host and that the host will resolve the user name, placing the message in the user's mailbox

3.18 Abbreviated Domain Specification

Since any number of levels is possible within the domain hierarchy, specification of a fully qualified address can become inconvenient. This standard permits abbreviated domain specification, in a special case for the address of the sender, calls the left-most sub-domain

Level N. In a header address, if all of the sub-domains above (i.e., to the right of) Level N are the same as those of the sender, and then they do not have to appear in the specification. Otherwise, the address must be fully qualified. This feature is subject to approval by local sub domains. Individual sub-domains may require their member systems, which originate mail, to provide full domain specification only. When permitted, abbreviations may be present only while the message stays within the sub-domain of the sender. Use of this mechanism requires the sender's sub-domain to reserve the names of all top-level domains, so that full specifications can be distinguished from abbreviated specifications. For example, if a sender's address is:

sender@registry-A.registry-1.organization-X
And one recipient's address is:

recipient@registry-B.registry-1.organization-X

and another's is:

recipient@registry-C.registry-2.organization-X

Then ".registry-1.organization-X" need not be specified in the message, but “registry-C.registry-2” does have to be specified. That is, the first two addresses may be abbreviated, but the third address must be fully specified. When a message crosses a domain boundary, all addresses must be specified in the full format, ending with the top-level name-domain in the right-most field. It is the responsibility of mail forwarding services to ensure that addresses conform to this requirement. In the case of abbreviated addresses, the relaying service must make the necessary expansions. It should be noted that it often is difficult for such a service to locate all occurrences of address abbreviations. For example, it will not be possible to find such abbreviations within the body of the message. The "Return-Path" field can aid recipients in recovering from errors.

3.19 Multiple Mailboxes

An individual may have several mailboxes and wish to receive mail at whatever mailbox is convenient for the sender to access. This standard does not provide a means of specifying” any member of" a list of mailboxes. A set of individuals may wish to receive mail as a single unit (i.e., a distribution list). The <group> construct permits specification of such a list. Recipient mailboxes are specified within the bracketed part (). A copy of the

transmitted message is to be sent to each mailbox listed. This standard does not permit recursive specification of groups within groups. While a list must be named, it is not required that the contents of the list be included. In this case, the <address> serves only as an indication of group distribution and would appear in the form: name:; Some mail services may provide a group-list distribution facility, accepting a single mailbox reference, expanding it to the full distribution list, and relaying the mail to the list's members. This standard provides no additional syntax for indicating such a service. Using the <group> address alternative, while listing one mailbox in it, can mean either that the mailbox reference will be expanded to a list or that there is a group with one member. Internet cloud model is described in figure 3-2.

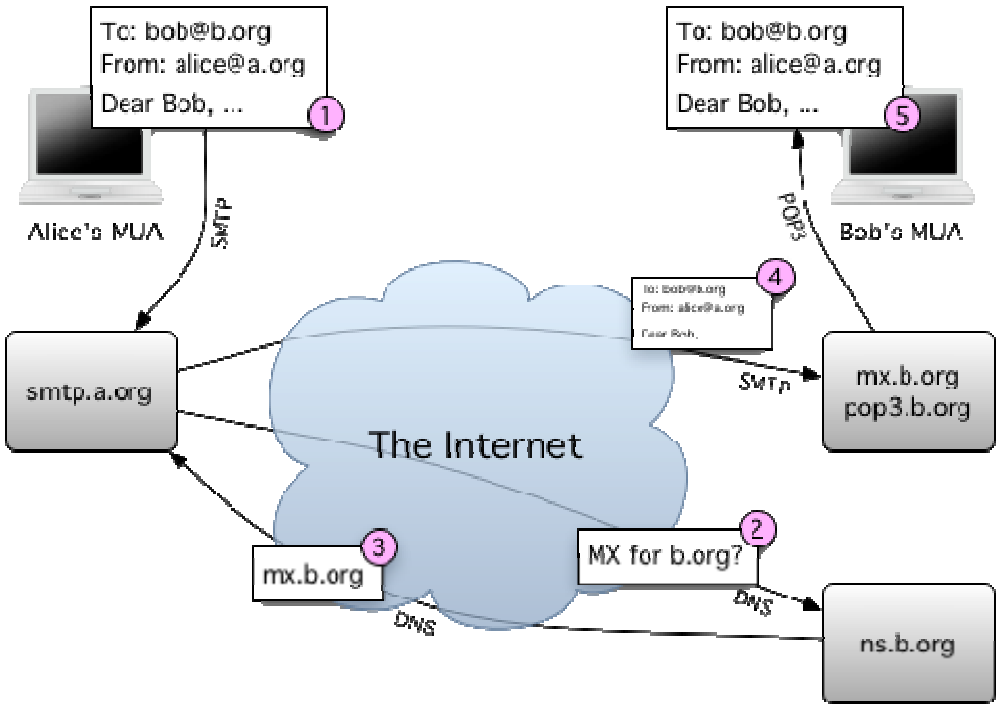

Figure 3-2: Internet Cloud

3.20 Explicit Path Specification

At times, a message originator may wish to indicate the transmission path that a message should follow. This is called source routing. The normal addressing scheme, used in an adder-spec, is carefully separated from such information; the <route> portion of a route-adder is provided for such occasions. It specifies the sequence of hosts and/or transmission services that are to be traversed. Both domain-refs and domain-literals may be used.

3.20.1 Reserved Address

It often is necessary to send mail to a site, without knowing any of its valid addresses. For example, there may be mail system dysfunctions, or a user may wish to find out a person’s correct address, at that site. This standard specifies a single, reserved mailbox address (local-part) which is to be valid at each site. Mail sent to that address is to be routed to a person responsible for the site's mail system or to a person with responsibility for general site operation.

4 Design

4.1 Technologies Behind Qasid Mail

Qasid Mail provided an opportunity to work with existing programming languages and a variety of ways those languages is used to build robust applications. Not only did it encourage familiarization with API libraries and the different ways to use them, it also encouraged discovery of ways to enhance and add to those libraries or create additional libraries. The Java programming language is popular in developers to create a variety of applications, making use of its massive class library. In addition, some projects take advantage of optional packages like the Java Mail API.. Qasid Mail uses the Java Mail API, JavaBeans Activation framework, Java I/O, and features much of the Project Swing library. Airmail also takes advantage of what the Java class libraries have to offer and then some. Yet, Qasidmail is not perfect. The APIs provide only so much functionality, then enhancements and workarounds come into play. This chapter covers how the Graphical User Interface (GUI), and JavaMail are implemented in Qasidmail, and how each of these features are enhanced for application improvement. In addition, this article concludes with problem areas needing the developer's touch, and with a thread problem area and how it might possibly be resolved.

4.2 Qasidmail Hierarchy

Qasidmail's main method is in `Qasidmail.java`, which creates an instance of the Qasidmail application, sets preferences, such as language, local and time, and a `shutDownQasidmail` method to neatly close the application. The UI uses many common Swing components such as `JButton`, `JLabel`, `JToolBar`, `JPanel`, `JSplitPane`, and `JTextField`. Qasidmail also uses the AWT layout manager `BorderLayout`, and AWT event listeners. Java I/O is used for reading and writing files to the system. The basic Qasidmail class hierarchy is as shown in the fig :

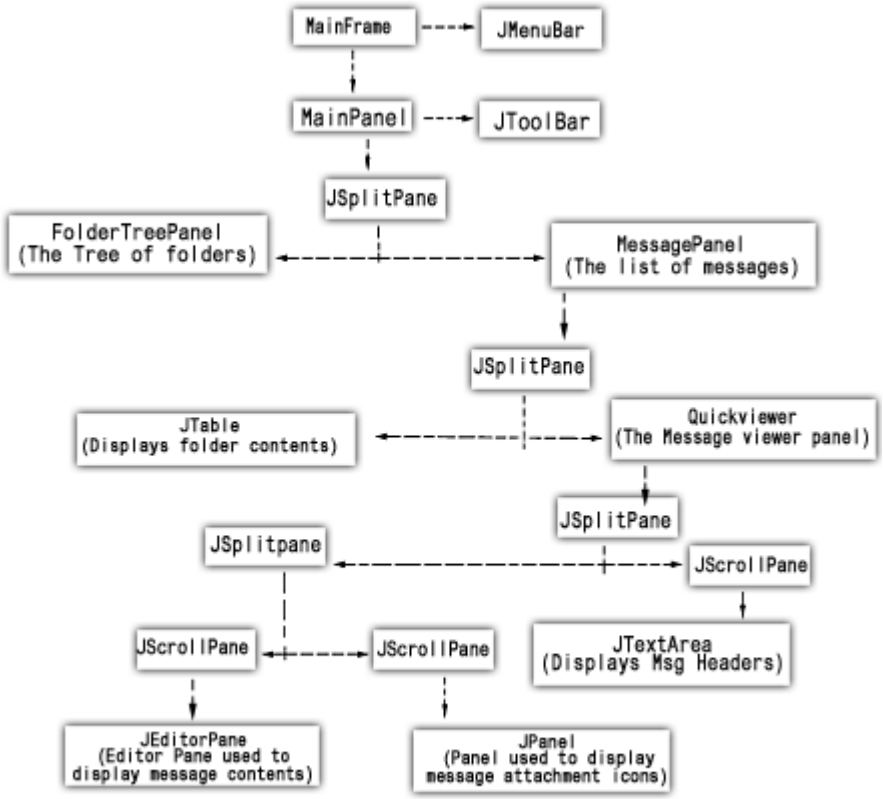

Figure 4-1: Class Hierarchy

4.3 Project Swing, Viewing Attachments, and Methods

Every `Frame` or `Panel`, or other significant GUI property, provides `loadProperties` and `saveProperties` methods to save and restore the components in the positions last set by the user. This saves window positions (note the need to deal with windows "minimized" windows), split pane positions, and other components attributes are restored each time the application runs. Parent components, call the `load` and `save` methods of their children components to cover the tree of components.

4.4 JEditorPane and the QuickViewer Class

Qasidmail's `QuickViewer` panel uses the `JEditorPane` to display the email message header, body, and attachments in one view. The editor pane components are handy because

the same basic code is used to display different types of content. Currently, `JEditorPane` supports styled text, HTML, and RTF (Rich Text Format).When a user clicks a message line in the mailbox panel that shows each line in a summary line, the `QuickViewer` class displays the contents of that message. This is done with a call to the `setMessage( Message msg )` method, which parses the MIME message to determine the best way to display the message:

parseMessagePart(this.message);

parseMessagePart interprets the MIME message and finds a reasonable way to display the message. Qasidmail looks over the message parts and takes the first one it finds that is of mime type text/*. The mime type multipart/* is further broken down by recursive calls to parseMessagePart. It might be more appropriate for Qasidmail to account for the INLINE disposition also, but in practice, this is rarely necessary. As parseMessagePart parses the message, any message parts it finds that are not considered to be a part of the display is added to a Vector of parts considered to be attachments. When Qasidmail sees that a message has attachments, it displays them as icons in a panel to the right of the message. This allows the user to click the icons to display the attachment, save the attachment to a file, or get the attachment's properties. When parseMessagePart finally locates the part that it believes most represents the email message, it calls to establishEditorPane to establish a JEditorPane component that presents the MIME part content. Currently Swing supports the mime types text/plain, text/rtf, and text/html. Unfortunately, the PDF package does not support JEditorPane, so PDF can't be displayed inline. The establishEditorPane method is fairly simple. It just looks at the mime type of the part, establishes a JEditorPane with that content type, then reads the message part's content into the JEditorPane document establishEditorPane(Part body Part)

```
{     JEditorPane pane = null;
      EditorKit editor = null;
      Document doc = null;
       try {     // Get the ContentType (or mimetype)        // of the body part to display
      ContentType xct =   MessageUtilities.getContentType( bodyPart );
```

```
// Try to match the content type
// with one of the types
// that is suported
if ( xct.match( "text/html" ) )
{  pane = new JEditorPane();
   pane.setContentType( "text/html" );
  pane.setEditable( false );
   editor = pane.getEditorKit();
{     // Use one of our own utilities
      // to get a BufferedReader
      // that reads the contents
      // of the body part.
BufferedReader xreader =   MessageUtilities.getTextReader(bodyPart );
// Let the editor read the
// contents into the document.
editor.read( xreader, doc, 0 );
// Done with the reader.
xreader.close();}
   this.editorPane_ = pane;    }
 catch ( IOException ex )
   {  ex.printStackTrace( System.err );
   this.editorPane_ = null;    }
    catch ( BadLocationException ex )    {
   ex.printStackTrace( System.err );
   this.editorPane_ = null;  }
       catch ( MessagingException ex ) {
   ex.printStackTrace( System.err );
   this.editorPane_ = null;    }  }
```

Note the creation of the editor pane for the `text/*` differs from the HTML and RTF cases. `JTextPane` is a subclass of `JEditorPane`, and the listener allows the cursor to change over hot links, as well as handling clicking a hot link. The events handed to the listener are

"entering the hotlink" (displays hotlink cursor), "exiting the hotlink" (display the normal text cursor), and "link activated" (displays the new page that the link points to). Once returned from the `parseMessagePart` method, the `Quick Viewer` code checks if the editor pane has been set during the parsing, and to display the contents of the message. The code displays the new editor pane in the panel.

If there were no attachments, the body component to the editor pane is set. Otherwise, the display is set to the split pane that presents the editor pane on the left and the attachments panel to the right. The editor pane is represented in the scroll pane by editor’s roller_:

4.5 Viewing Messages and Implementing "View As"

One of the peculiarities of writing a MIME compatible email client is that some of the clients that send you email are not compliant, out of date, or misconfigured. The most common problem associated with this is the classic application/octet-stream MIME content type. Often, email clients are installed with minimal MIME type configurations and expect the user to configure what is needed. Of course, many users do not realize they need to configure these settings and begin using the client software. For instance, they send email with a JPEG image attached, but the attachment has the catch-all MIME type application/octet-stream.

When Qasidmail sees this generic content type, it does not have any means of determining how to view the attachment. Some clients attempt to make a determination from the filename suffix, such as .jpg, if the attachment specifies a name. If there is no name, the normal behavior is to display a hex editor showing the data in binary form.Qasidmail uses a feature in the attachment pop-up menu (right-click an attachment icon) that allows the user to view the attachment "as" the MIME type that the user specifies. If the user knows that the attachment is a JPEG image, it's possible tell Qasidmail to view the unknown type attachment as if it were of type image/jpeg.Qasidmail implements this feature by defining a new Data Source that wraps the original Data Source. Data Sources are defined by the Java Activation Framework, which is the core of MIME implementation. They provide a common interface to access various types of data, as well as configurable mechanisms for viewing these data types. The wrapper Data Source only adds one feature -- it remembers

the "view as" content type, and returns whenever the getContentType() method is called. Otherwise, it redirects every other call to the original DataSource. For esoteric reasons, code was also added to get a "name" from the Content Type of the original DataSource, if it is provided as a MIME parameter, as is often the case with email attachments.DataSource is implemented in the class org.qasidmail.mail.

4.6 JavaMail Implementation and Enhancements

Qasidmail Mail Handler JavaMail Service Provider is a simple file system-based mail `Store` object that reads messages stored in the Mail Handler (MH) format. Qasidmail MH service provider represents mail folders, using local directories and mail messages as files in those directories. The MH `Store` expects a `URLName` field that is different from other JavaMail stores. It uses the "File" portion of the `URLName,` and ignores the host and port portions of the URL, unless the port is set to the value `4261,` then the debug flag for the provider turns on enabling debugging output to the console. The file portion of the URL, as returned by the getFile method, is used to define the path name of the MH root folder. The provider recognizes path names that begin with `~,` and replaces the ~ with the user's home directory, as defined by the System property `user.home.` The provider recognizes path names that begin with a `.`, and replaces the dot with the current directory name, as defined by the `System` property `user.dir`.

Table 4-1: URL names

URLName	Filesystem location
mh:~	User's home directory.
mh:~/Mail	The Mail folder.
mh://:4261/~	User's home directory with debugging turned on.
mh:C:/Windows/Profiles/Mail	Home directory (Windows).
mh:/C:/Windows/Profiles/Mail	Home directory (Windows).

The Store protocol Connect method simply returns true. If there were some means of authenticating the user, code could be added to protect a folder. As it is, this protection is assumed to be provided by the file system permissions. The close method is a no-pounce the path name is resolved to a directory; the MH store provider considers the files in that folder to belong to the default folder of the store, typically considered the in box. Otherwise, the provider parses the mail folder name on mapping name elements to directories relative to the root directory, and considering files within those directories to be the mail messages belonging to the mail folder. MH stores the messages using filenames that are simple integers. This way it keeps one file, named "index", containing the last integer used for a message. This makes it easy to get the id for a new message. So messages file names look like: 1, 2, 3, 4, 32, 49, 104, 1223, As messages are deleted, "holes" develop in the sequence of IDs. The store is implemented by the MHStore class. The MHFolder class, which implements the Folder for the provider, is straightforward, since it does not have to do much work. The message per file avoids having to parse files to locate messages, as with the MBOX format, and JavaMail provides the code needed to parse the MIME messages in those files. We simply need to manage mapping folder names, such as "INBOX", misc/people/friends, and misc/people/work, into their corresponding directories, and handling the message files for open, append, and delete requests. Again, because the messages are kept in file units, these operations are simple, since they are implemented by the file system. In addition, a cache is provided in MH. This is a debatable feature, since it is rare to require caching of messages on a local file system, but it's a benefit to file systems mounted over slow connections. Finally, the class MHMessage is a subclass of MimeMessage. Because MimeMessage provides the functionality for messages, it simply provides an Input Stream to the file containing the mail message. Three very simple classes make a complete JavaMail Store Provider that allows JavaMail clients to read an MH message store on a local disk or mounted from a file server. The Qasidmail is made aware of the MH provider by adding the following entry to the javamail.providers file, which appears in the file on one single line:

protocol=mh; type=store;
class=com.ice.javamail.mh.MHStore;

vendor=Tim Endres;

The protocol matches the mh: prefix of the URLName. The type indicates that this provider implements a Store. the class property identifies the class that implements the store, and lastly, the vendor.

4.7 JavaMail and Internationalization

JavaMail and J2SE have many classes that support international character encodings by default, but there's no easy way to support multiple character encodings within MIME messages. The main problem stems from the MIME standard because any mail message can be composed of any number of character encodings. Qasidmail splits the MIME message handling routines into a set of utility routines, Message Utilities. These routines allow Qasidmail to decode the message parts from any character encoding into Unicode equivalents. They also encode message parts to character encodings before sending messages, such as the `attach` routines with construct message parts from files, text, or other message parts.

Qasidmail handles incorrectly constructed MIME messages since many people still use old applications. JavaMail throws a lot of different exceptions based on the part being decoded, which required Qasidmail to have special handling for each part and exception. In addition, default handling of message contents were added in order to recover from ill-formed messages, as in `getTextReader`.

```
// pick the character set off of the content type
  ContentType xct =  MessageUtilities.getContentType( part );
  String xjcharset =    xct.getParameter( "charset" );
  if ( xjcharset != null ) {
    xjcharset =  MimeUtility.javaCharset( xjcharset );
  } else {
  // not present, assume ASCII character encoding
     xjcharset =  MimeUtility.javaCharset( "ASCII" );  }
```

It took a lot of trial and error to make Qasidmail robust enough to handle international email messages.

4.8 Thread and Dialog Problems and the Dialog Runner Classes

Since the process may take time when a newly composed message is sent, a progress dialog to show the progress of sending the message is needed in Qasidmail, a simple dialog box with a progress bar component in it. Then when message transmission is initiated, the transmission code should update the progress bar to show the progress to the user.There are a couple of difficulties involved , the dialog box must be displayed in such a way as to not block the event thread that typically invokes the operation. Otherwise, the event thread blocks with the dialog box, and the display is not updated until the dialog is dismissed. Thus, the progress bar does not update its display, and the user doesn't get the feedback that is intended. Second, a more difficult problem involves synchronizing the two threads that display the dialog and perform the operation being monitored. By investigating the code used by `DialogRunner`, the problem, and its solution become clearer. One way around these problems is to use a non-modal dialog. Non-modal dialogs do not block on the call to `show`. Thus, the code can call `show` to display the dialog and the call returns and allows continued processing. However, modal dialogs do not behave in the same fashion. When the `show` method is called, the current execution thread blocks until the dialog box is dismissed. The current release of Qasidmail uses the non-modal approach, for historical reasons. However, the code to make this work with modal dialogs should be considered.The general structure of using a non-modal dialog as it is coded in Qasidmail starts with an interface definition. This interface is described in table 4-2.

Table 4-2: interface

```
interface
DialogRunnable
{
abstract public void
run( JDialog dialog,
     JProgressBar progress );
```

This interface is used by the class DialogRunner and the DialogRunner class is handed to any of the runnables to use. DialogRunner is a simple class, whose entire functionality is contained in the run method.First, `dialog.show` is called. This pops up the progress dialog that was created by the `DialogRunner` class, and starts the progress bar forward. It then proceeds through the steps of retrieving the new email messages for the currently displayed folder, all the while incrementing the progress indicator. Finally, `dialog.dispose` is called, which dismisses the dialog box. The `run` method returns, completing the thread's execution. While this code appears to work properly, and the average user will never know the difference, this code is plagued with problems. To see why, look back to the code that started this entire process, the actionPerformed method in MessagePanel and the code that implements the GETNEWMAIL command.

```
if ( this.folder != null )
{
MainFrame.getInstance(
        ).setWaitCursor();
MailEventThread.notifyIncoming(this.folder, false );
Object[] xargs = new Object[1];
xargs[0] = this.folder.getName();
String msg =
   Qasidmail.bundle.getFormatString
 ( "MessagePanel.Retrieving",
                    xargs );
DialogRunner runner = new DialogRunner
 ( this.new NewMailRunner(),
  (Frame)getTopLevelAncestor(),
 Qasidmail.bundle.getString
 ( "MessagePanel.Retrieving.title" ),
msg );
runner.start();
MainFrame.getInstance().resetCursor();
```

Notice how the `DialogRunner` thread starts then simply falls through and returns from the `actionPerformed` method. The `DialogRunner` is now running outside of the Swing event thread. This has the nice benefit of the UI "updating" while the `DialogRunner` thread runs, because the Swing event thread is not blocked. However, it has the detrimental effect of not blocking the user interface from further actions by the user. To get a real feel for what this means, open a large folder in Qasidmail, and while the progress dialog is displayed, open several other folders. Right-click folders and open them. The progress dialogs for the folders you have tried to open appear. This problem resolves with a modal dialog. Modal dialogs "take over" the user interface, and while components continue to update, the user can not click the mouse, or perform any other input event, outside of the dialog box. This effectively "freezes" the user interface until the dialog is dismissed, which is when the processing code completes and disposes of the dialog. With the advantage of the modal dialog "freezing" the user interface comes a difficulty. The call to the `show` method of the modal dialog does not return until the dialog is dismissed. This is not immediately obvious:

```
JDialog dlg =
this.setupProgressDialog( args );
dlg.show();
```

Once the `dlg.show` method is called, it does not return until the dialog is dismissed. Given this code, and because the dialog controls all input events, the only way to dismiss the dialog, and return from the `show` method, is for some user event within the dialog box to call the `JDialog.dispose` method. The problem is that we do not want a component in the dialog box to dispose of it. That should be done by the processing code that is using the dialog to display its progress. This code can not be placed after the call to `show`, as Qasidmail does in the examples above, because `show` does not return. The obvious solution is to simply spin off a thread to perform the processing just before the call to `show`. More pseudo code described in table 4-3 and4-4.

Table 4-3: Algo of Dialouge

```
JDialog dlg =
  this.setupProgressDialog( args );
  ProcessingThread thread =
   new ProcessingThread( dlg, args );
thread.start();
dlg.show();
```

And in ProcessingThread:

Table 4-4: Algorithm

```
public void
run()
{
//perform processing incrementing
// progress bar
dlg.setVisible( false );
}
```

In other words, set up the dialog and the thread, start the thread, and show the dialog. This causes the current thread block until the dialog box is dismissed, at which time the show method returns. The processing thread calls dlg.setVisible(false) when it is finished processing. The dialog displays and controls user input, until the processing thread completes, then the dialog is dismissed. Yet, there is still a problem. The code contains a race condition. If the processing thread gets the VM before the call to JDialog.show, the call to dlg.setVisible(false) can occur before the show call. If this happens, the dialog appears, but is never removed. The sample class ProgressTest demonstrates this event

when the Race Condition button is clicked. The following output from the ProgressTest application shows the order of execution between the two threads.

```
Normal
Progress Test Started ----------
Dialog estblished.
Thread constructed.
Thread started.
Showing the dialog.
Thread run() starts.
Thread run() making dialog not visible.
Dialog.show() has returned.
Dialog.dispose() has returned.
Thread run() made dialog not visible.
Thread run() ends.
Race - Note that "Showing the dialog" comes
after "Thread run made dialog not visible!"
Progress Test Started -------
Dialog estblished.
Thread constructed.
Thread started.
Thread run() starts.
Thread run() making dialog not visible.
Thread run() made dialog not visible.
Thread run() ends.
Showing the dialog.
Dialog.show() has returned.
Dialog.dispose() has returned.
```

In the race condition case, clicking the close icon in the dialog box to dismiss it reveals the Dialog.show has returned a message. Eliminating this problem is accomplished by the simplest of means. Don't execute code that could possibly block, which relinquishes the VM to another thread, in the critical section of code that exists between starting the

processing thread, and calling the JDialog.show method on the progress dialog. This eliminates the problem because of the nature of the Java Virtual Machine (JVM)1 Threading Model. The JVM does not time-slice like a UNIX OS might. Currently running threads does not relinquish the VM unless they make a call to a method that blocks in some way. Since the code between starting the processing thread and showing the dialog never blocks, the processing thread never gets a chance to run. When the show method is called, the thread blocks and the processing thread has a chance to run.Adding synchronization, or trying to use a wait and notify approach might seem helpful, but no matter how the the distance between starting the processing thread and showing the dialog is tightened, there is still the chance of native threading executing in such a way that the processing code executes setVisible(false) before JDialog.show is called. Granted, assuming normal conditions, and also assuming that processing never takes less than a fraction of a second, sleep(100) is added to the beginning of the thread's run method), it is very unlikely that the processing finishes before the dialog is displayed. However, in the case of an XWindows-based Java application on a slow connection, this is still a possibility. If an object is passed into the show method that notifies when the dialog is shown, the JDialog.show method calls the notifyAll method on that object. The processing code would then wait on the same object, ensuring that it did not run until the dialog was shown:

```
JDialog dlg =
this.setupProgressDialog( args );
dlg.show( dlg );
```

In the processing thread's `run` method:

```
public void
 run(){
try { dlg.wait(); }
catch ( InterruptedException ex ) { ... }
   // perform processing incrementing
progress bar
dlg.setVisible( false ); }
```

To be certain the `run` method is not called, use `dlg.setVisible( false )` before the dialog box is displayed. Qasidmail is a good example of an open-source project that demonstrates how various Java APIs work together and are used in complete applications. While some developers enjoy developing the UI with Swing, other developers may prefer to work with message handling using the JavaMail API. With many developers working on an open-source project, solutions to problems, like those listed above, are more likely to occur and improve applications.Qasid Mail is an email client application written entirely in Java. The application is designed and implemented using the Java Mail API, and therefore inherits some look and feel from the functionality of this package.

Qasid Mail extends the Java Mail API, giving it additional functionality and usability, such as displaying attachments, accessing address books, etc. The possibilities of extending Qasid Mail are end-less as it can be developed further. Qasid Mail has requirements before it can be executed or used successfully.

4.9 Java Runtime Installation

Qasid Mail is written using the Java language and requires a Java Runtime Environment (JRE). Qasid Mail is compatible with Java 1.2 runtime environments (JRE), You only need the Java Development Kit (JDK) to build Qasid Mail from the source code.

4.10 Mail Server

Qasid Mail requires access to a mail server for reading your email, and a mail server to send email, i.e. SMTP mail server.

4.11 Java Foundation Classes

Qasid Mail requires the presence of the Java Foundation Classes, also known as Swing. The JFC package is included in Java 1.2 runtime environments, and therefore is not required in newer JRE versions. Extensions Qasid Mail uses several third-party packages to extend the functionality of the application. These extensions come in the form of attachment viewers, support, and address books. Java Mail API.Qasid Mail uses the Java Mail package extensively to access and send email messages. This package is part of the Qasid Mail.

5 Installing Qasid Mail

Qasid Mail is a single, large JAVA Jar file which contains everything necessary to run on any computer with a compatible Java Virtual Machine (JVM). The command will create a single directory, , which contains libraries and the source code.. All libraries that Qasid Mail is comprised of can be found in the '*lib*' sub-directory.

5.1 Windows Specific Information

Java applications are not specific to a given platform. Qasid Mail has two dynamic link libraries (DLLs) to make it integrate with Windows applications. The simplest approach is to simply leave the two DLL files in the '*lib*' folder of the Qasid Mail. If Qasid Mail is started with the, *run. bat*, then Windows will be able to find the DLL files.

5.2 Running Qasid Mail

Starting the Mail AutomaticallyTo start the applications double-click the Qasid Mail jar file, “qasidmail.jar” and start the application.

5.3 Run-time Environment

Qasid Mail automatically generates all property files it needs and places them in the home directory. All properties are configured via dialog boxes when Qasid mail is installed and run for the first time. The Qasid Mail.ldif file contains all personal contact information, including email addresses, addresses, phone numbers, etc. The file is maintained in a format called LDIF (LDAP Interchange Format). The contents of the file can be read with any word processor.

5.4 Qasid Mail Configuration

Qasid Mail is configured via a series of dialog boxes. When you first launch Qasid Mail, if it determines that you have not configured the most important properties, it will automatically display the dialogs to configure these items. All of the configuration dialogs are available via the Config menu. Each dialog is discussed via the list in the table 5.1.

Table 5-1: Information

User Information	Configures your personal information, such as your email address, personal name, reply to address, etc.
Signatures	Configures your signatures.
Local Manual	Configures the location of the Qasid Mail on your local disk so you can read the manual while you are off-line
Mail Stores	Configures the stores that you wish to use for viewing email. This is where you identify the store that contains your INBOX and other stores you wish to use.
SMTP Transport Properties	Configures the properties used if the SMTP Transport Protocol is used.

5.5 Creating Folders

You can create new folders in Qasid Mail. To do so, simply select a folder in the folder tree panel, so that it is highlighted. Then, select the "Create Folder" menu command. This will display a dialog asking for the name of the new folder. This is not a folder pathname, just the name of the folder. There are also two check boxes indicating whether the folder can contain messages or folders. Once you have filled in the name, and checked a box, click ok and the folder will appear in the folder panel.

5.5.1 Folder Paths

Folder paths are like file paths. They start with the Store Name, followed by a slash, followed by a folder name. If the folder is inside of other folders, you use a slash separated pathname to the desired folder.

5.5.2 Mail Folders

Mail folders are the things that contain mail messages. Folders can also contain other folders. If a folder can contain other folders; Qasid Mail will display the folder with a folder icon. If the folder can only contain messages and not other folders, then Qasid Mail

will display it with a document icon. Folders can not exist on their own. They must be contained by either a store or another folder. Thus, every folder has a path name that uniquely identifies the stores and folders that you must open to get to the folder identified by the pathname.

5.5.3 Mail Stores

Mail stores are the highest level container in the Java Mail API. Stores are contained by nothing. They are the root of all folder trees. In practice, a Mail Store is synonymous with a mail login or user. A store is specified by a URL, and that url typically identifies a mail host and login. Stores are identified by Java Mail using their URL. However, Qasid Mail uses the store's name when referencing a store in a folder pathname or in other cases. When you define a store in the Store Configuration Dialogue you give the store its name. You will use this name any time that Qasid Mail refers to a store name or a folder path.

5.5.4 Copy Message

In order to copy a message Qasid Mail needs to know which message to copy, and the destination folder. The message that is copied is the currently displayed message. The destination folder is the folder that is selected in the Folder Panel which is the left panel displaying the Store and Folder tree.

5.5.5 Moving Messages

In order to move a message Qasid Mail needs to know which message to move, and the destination folder. The message that is moved is the currently displayed message. The destination folder is the folder that is selected in the Folder Panel which is the left panel displaying the Store and Folder tree. When you move a message, it is not expunged from the source folder; it is simply *marked* for delete.

5.5.6 Saving Message Compositions

If you compose a message, and attempt to close the window before you have sent the message, Qasid Mail will prompt you to save the message. When you save the message, you will need to specify a Folder path name to which you wish to save the message.

5.5.7 Composition Configuration

This dialog will allow you to configure items related to the composition of email. Specify the store in which to save mail messages that have not been sent yet. This allows you to save a message that you are composing for later edit and sending. To send a message that you have saved, open the store that you configure here, and select the message in the list on the right, and use the "Edit Message" command in the "Mail" menu. The mail message will be displayed in a compose window in the state that you saved it. The store name used in this configuration must be a *full path name* to completely describe the mail folder to use. For this reason, the tree of mail folders is displayed at the bottom of the dialog. Clicking on any of the mail folder icons in the tree will cause the folder's full path name to be entered for you automatically.

5.5.8 Transport Configuration

The transport configuration determines what Qasid Mail uses to send outgoing email. If you never send email with Qasid Mail, and only use it for reading email, then you do not need to worry about this configuration. In WinXP SMTP protocol provider is available to send email.

6 Multipurpose Internet Mail Extensions (MIME)

MIME describes several mechanisms that combine to solve most of the problems without introducing any serious incompatibilities with the existing world. In particular, it describes: A MIME-Version header field uses a version number to declare a message to be conformant with MIME and allows mail processing agents to distinguish between such messages and those generated by older or non-conformant software, which are presumed to lack such a field.

6.1 A Content-Type header field

This can be used to specify the media type and subtype of data in the body of a message and to fully specify the native representation (canonical form) of such data.

6.2 A Content-Transfer-Encoding header field

This can be used to specify both the encoding transformation that was applied to the body and the domain of the result. Encoding transformations other than the identity transformation are usually applied to data in order to allow it to pass through mail transport mechanisms which may have data or character set limitations.

6.3 Two additional header fields

Features can be used to further describe the data in a body, the Content-ID and Content-Description header fields. The entire header fields defined is subject to the general syntactic rules for header fields specified. In particular, all of these header fields except for Content-Disposition can include comments, which have no semantic content and should be ignored during MIME processing. Finally, to specify and promote interoperability, it provides a basic applicability statement for a subset of the above mechanisms that defines a minimal level of "conformance" with this document. Several of the mechanisms described in this set of documents may seem somewhat strange or even baroque at first reading.

6.4 Definitions and Conventions

6.4.1 CRLF

The term CRLF, in this set of documents, refers to the sequence of octets corresponding to the two US-ASCII characters CR (decimal value 13) and LF (decimal value 10) which, taken together, in this order, denote a line break in mail.

6.4.2 Character Set

The term "character set" is used in MIME to refer to a method of converting a sequence of octets into a sequence of characters. unconditional and unambiguous conversion in the other direction is not required, in that not all characters may be represent able by a given character set and a character set may provide more than one sequence of octets to represent a particular sequence of characters. This definition is intended to allow various kinds of character encodings, from simple single-table mappings such as US-ASCII to complex table switching methods such as those that use ISO 2022's techniques, to be used as character sets. However, the definition associated with a MIME character set name must fully specify the mapping to be performed. In particular, use of external profiling information to determine the exact mapping is not permitted. The term "character set" was originally to describe such straightforward schemes as US-ASCII and ISO-8859-1 which have a simple one-to-one mapping from single octets to single characters. Multi-octet coded character sets and switching techniques make the situation more complex. For example, some communities use the term” character encoding" for what MIME calls a "character set", while using the phrase "coded character set" to denote an abstract mapping from integers (not octets) to characters.

6.4.3 Message

The term "message", when not further qualified, means either a (complete or "top-level") message being transferred on a network, or a message encapsulated in a body of type "message” or "message/partial".

6.4.4 Entity

The term "entity", refers specifically to the MIME-defined header fields and contents of either a message or one of the parts in the body of a multipart entity. The specification of such entities is the essence of MIME. Since the contents of an entity are often called the "body", it makes sense to speak about the body of an entity. Any sort of field may be present in the header of an entity, but only those fields whose names begin with "content-" actually have any MIME-related meaning. Note that this does NOT imply that they have no meaning at all an entity that is also a message has non MIME header fields whose meanings are defined.

6.4.5 Body Part

The term "body part" refers to an entity inside of a multipart entity. The term "body", when not further qualified, means the body of an entity, that is, the body of either a message or of a body part. The previous four definitions are clearly circular. This is unavoidable, since the overall structure of a MIME message is indeed recursive.

6.4.6 7bit Data

"7bit data" refers to data that is all represented as relatively short lines with 998 octets or less between CRLF line separation sequences. No octets with decimal values greater than 127 are allowed and neither are NULs (octets with decimal value 0). CR (decimal value 13) and LF (decimal value 10) octets only occur as part of CRLF line separation sequences.

6.4.7 8bit Data

"8bit data" refers to data that is all represented as relatively short lines with 998 octets or less between CRLF line separation sequences, but octets with decimal values greater than 127 may be used. As with "7bit data" CR and LF octets only occur as part of CRLF line separation sequences and no NULs are allowed.

6.4.8 Binary Data

"Binary data" refers to data where any sequence of octets whatsoever is allowed.

6.4.9 Lines

"Lines" are defined as sequences of octets separated by a CRLF sequences. "Lines" only refers to a unit of data in a message, which may or may not correspond to something that is actually displayed by a user agent.

6.4.10 MIME Header Fields

MIME defines a number of new header fields that are used to describe the content of a MIME entity. These header fields occur in at least two contexts as part of a regular message header In a MIME body part header within a multipart Construct.

There has really been only one format standard for Internet messages, and there has been little perceived need to declare the format standard in use. This is an independent specification that complements. Although the extensions have been defined in such a way as to be compatible, there are still circumstances in which it might be desirable for a mail-processing agent to know whether a message was composed with the new standard in mind. Therefore, this document defines a new header field, "MIME-Version", which is to be used to declare the version of the Internet message body format standard in use. Messages composed in accordance with this document MUST include such a header field. The presence of this header field is an assertion that the message has been composed in compliance with this document. Since it is possible that a future document might extend the message format standard again, a formal BNF is given for the content of the MIME-Version field: Thus, future format specifies, which might replace or extend "1.0", are constrained to be two integer fields, and separated by a period. If a message is received with a MIME-version value other than "1.0", it cannot be assumed to conform to this document. The MIME-Version header field is required at the top level of a message. It is not required for each body part of a multipart entity. It is required for the embedded headers of a body of type" message/" or "message/partial" if and only if the embedded message is itself claimed to be MIME-conformant. It is not possible to fully specify how a mail reader that conforms to MIME as defined in this document should treat a message that might arrive in the future with some value of MIME-Version other than "1.0". It is also worth noting that version control for specific media types is not accomplished using the

MIME-Version mechanism. In particular, some formats (such as application/postscript) have version numbering conventions that are internal to the media format. Where such conventions exist, MIME does nothing to supersede them. Where no such conventions exist, a MIME media type might use a "version" parameter in the content-type field if necessary. When checking MIME-Version values comment strings that are present must be ignored. In particular, the following four MIME-Version fields are equivalent: In the absence of a MIME-Version field, a receiving mail user agent (whether conforming to MIME requirements or not) may optionally choose to interpret the body of the message according to local conventions. Many such conventions are currently in use and it should be noted that in practice non-MIME messages can contain just about anything. It is impossible to be certain that a non-MIME mail message is actually plain text in the US-ASCII character set since it might well be a message that, using some set of nonstandard local conventions that predate MIME, includes text in another character set or non-textual data presented in a manner that cannot be automatically recognized.

6.5 Content-Type Header Field

The purpose of the Content-Type field is to describe the data contained in the body fully enough that the receiving user agent can pick an appropriate agent or mechanism to present the data to the user, or otherwise deal with the data in an appropriate manner. The value in this field is called a media type. The Content-Type header field was first defined it uses a simpler and less powerful syntax, but one that is largely compatible with the mechanism given here. The Content-Type header field specifies the nature of the data in the body of an entity by giving media type and subtype identifiers, and by providing auxiliary information that may be required for certain media types. After the media type and subtype names, the remainder of the header field is simply a set of parameters, specified in an attribute=value notation. The ordering of parameters is not significant. In general, the top-level media type is used to declare the general type of data, while the subtype specifies a specific format for that type of data. Thus, a media type of "image/xyz" is enough to tell a user agent that the data is an image, even if the user agent has no knowledge of the specific image format "xyz". Such information can be used, for example, to decide whether or not to show a user

the raw data from an unrecognized subtype such an action might be reasonable for unrecognized subtypes of text, but not for unrecognized subtypes of image or audio. For this reason, registered subtypes of text, image, audio, and video should not contain embedded information that is really of a different type. Such compound formats should be represented using the "multipart" or "application" types. Parameters are modifiers of the media subtype, and as such do not fundamentally affect the nature of the content. The set of meaningful parameters depends on the media type and subtype. Most parameters are associated with a single specific subtype. However, a given top-level media type may define parameters which are applicable to any subtype of that type. Parameters may be required by their defining content type or subtype or they may be optional. MIME implementations must ignore any parameters whose names they do not recognize. For example, the "charset" parameter is applicable to any subtype of "text", while the "boundary" parameter is required for any subtype of the "multipart" media type. There are NO globally-meaningful parameters that apply to all media types. Truly global mechanisms are best addressed, in the MIME model, by the definition of additional Content header fields. In the future, more top-level types may be defined only by a standards-track extension to this standard. If another top-level type is to be used for any reason, it must be given a name starting with "X" to indicate its non-standard status and to avoid a potential conflict with a future official name.

6.6 Syntax of the Content-Type Header Field

The type, subtype, and parameter names are not case sensitive. For example, text, Text, and text are all equivalent top-level media types. Parameter values are normally case sensitive, but sometimes are interpreted in a case-insensitive fashion, depending on the intended use. (For example, multipart boundaries are case-sensitive, but the "access-type" parameter for message/External-body is not case-sensitive.) Value of a quoted string parameter does not include the quotes. That is, the quotation marks in a quoted-string are not a part of the value of the parameter, but are merely used to delimit that parameter value. In addition, comments are allowed in accordance with rules for structured header fields. Beyond this syntax, the only syntactic constraint on the definition of subtype names is the desire that

their uses must not conflict. That is, it would be undesirable to have two different communities using "Content-Type: application/foot bar" to mean two different things. The process of defining new media subtypes, then, is not intended to be a mechanism for imposing restrictions, but simply a mechanism for publicizing their definition and usage. There are, therefore, two acceptable mechanisms for defining new media subtypes Private values (starting with "X-") may be defined bilaterally between two cooperating agents without outside be registered or standardized New standard values should be registered with IANA .

6.7 Content-Type Defaults

Default messages without a MIME Content-Type header are taken by this protocol to explain text in the US-ASCII character set, which can be explicitly specified as Content type: text/plain; charset=us-ASCII This default is assumed if no Content-Type header field is specified. It is also recommended that this default be assumed when a syntactically invalid Content-Type header field is encountered. In the presence of a MIME-Version header field and the absence of any Content-Type header field, a receiving User Agent can also assume that plain US-ASCII text was the sender's intent. Plain US-ASCII text may still be assumed in the absence of a MIME-Version or the presence of a syntactically invalid Content-Type header field, but the sender's intent might have been otherwise

7 Operating Manual

QasidMail is an email client application written entirely in Java. The application is designed and implemented using the Java Mail API, and therefore inherits some look and feel from the functionality of this package. Qasid Mail extends the Java Mail API, giving it additional functionality and usability, such as displaying attachments, accessing address books, etc. The possibilities of extending Qasid Mail are end-less as it can be developed further. Qasid Mail has requirements before it can be executed or used successfully.

7.1 Java Runtime Installation

Qasid Mail is written using the Java language and requires a Java Runtime Environment (JRE). Qasid Mail is compatible with Java 1.2 runtime environments (JRE), You only need the Java Development Kit (JDK) to build Qasid Mail from the source code.

7.2 Mail Server

Qasid Mail requires access to a mail server for reading your email, and a mail server to send email, i.e. SMTP mail server.

7.3 Java Foundation Classes

Qasid Mail requires the presence of the Java Foundation Classes, also known as Swing. The JFC package is included in Java 1.2 runtime environments, and therefore is not required in newer JRE versions.

7.4 Extensions

Qasid Mail uses several third-party packages to extend the functionality of the application. These extensions come in the form of attachment viewers, support, and address books.

7.5 Java Mail API

Qasid Mail uses the Java Mail package extensively to access and send email messages. This package is part of the Qasid Mail. Java Activation Qasid Mail uses the Java Activation Classes and Beans.

7.6 Installing Qasid Mail

Qasid Mail is a single, large JAVA Jar file which contains everything necessary to run on any computer with a compatible Java Virtual Machine (JVM). The command will create a single directory, , which contains libraries and the source code.. All libraries that Qasid Mail is comprised of can be found in the '*lib*' sub-directory. .

7.7 Windows Specific Information

Java applications are not specific to a given platform. Qasid Mail has two dynamic link libraries (DLLs) to make it integrate with Windows applications. The simplest approach is to simply leave the two DLL files in the '*lib*' folder of the Qasid Mail. If Qasid Mail is started with the, *run. bat*, then Windows will be able to find the DLL files.

7.8 Running Qasid Mail

Starting the Mail Automatically to start the applications double-click the Qasid Mail jar file, “qasidmail.jar” and start the application.

7.8.1 Run-time Environment

Qasid Mail automatically generates all property files it needs and places them in the home directory. All properties are configured via dialog boxes when Qasid mail is installed and run for the first time. The Qasid Mail is an email client application written entirely in Java. The application is designed and implemented using the Java Mail API, and therefore inherits some look and feel from the functionality of this package.

Qasid Mail extends the Java Mail API, giving it additional functionality and usability, such as displaying attachments, accessing address books, etc. The possibilities of extending Qasid Mail are end-less as it can be developed further. Qasid Mail has requirements before it can be executed or used successfully.

7.8.2 Run-time Environment

Qasid Mail automatically generates all property files it needs and places them in the home directory. All properties are configured via dialog boxes when Qasid mail is installed and

run for the first time. The Qasid Mail.ldif file contains all personal contact information, including email addresses, addresses, phone numbers, etc. The file is maintained in a format called LDIF (LDAP Interchange Format). The contents of the file can be read with any word processor.

7.8.3 Qasid Mail Configuration

Qasid Mail is configured via a series of dialog boxes. When you first launch Qasid Mail, if it determines that you have not configured the most important properties, it will automatically display the dialogs to configure these items. All of the configuration dialogs are available via the Config menu. Each dialog is discussed via the list below. User Information configures your personal information, such as your email address, personal name, reply to address. Signatures Configures your signatures. Composition Configures where to save messages that you have composed by not sent. Local Manual configures the location of the Qasid Mail on your local disk so you can read the manual. Configures the stores that you wish to use for viewing email. This is where you identify the store that contains your INBOX and other stores you wish to use. SMTP Transport Properties configures the properties used if the SMTP Transport Protocol is used. Temp files configures directory in which temporary files are created.

7.8.4 Creating Folders

You can create new folders in Qasid Mail. To do so, simply select a folder in the folder tree panel, so that it is highlighted. Then, select the "Create Folder" menu command. This will display a dialog asking for the name of the new folder. This is not a folder pathname, just the name of the folder. There are also two check boxes indicating whether the folder can contain messages or folders. Once you have filled in the name, and checked a box, click ok and the folder will appear in the folder panel. folder Paths Folder paths are like file paths. They start with the Store Name, followed by a slash, followed by a folder name. If the folder is inside of other folders, you use a slash separated pathname to the desired folder.

7.8.5 Mail Folders

Mail folders are the things that contain mail messages. Folders can also contain other folders. If a folder can contain other folders; Qasid Mail will display the folder with a folder icon. If the folder can only contain messages and not other folders, then Qasid Mail will display it with a document icon. Folders can not exist on their own. They must be contained by either a store or another folder. Thus, every folder has a path name that uniquely identifies the stores and folders that you must open to get to the folder identified by the pathname.

7.8.6 Mail Stores

Mail stores are the highest level container in the Java Mail API. Stores are contained by nothing. They are the root of all folder trees. In practice, a Mail Store is synonymous with a mail login or user. A store is specified by a URL, and that url typically identifies a mail host and login. Stores are identified by Java Mail using their URL. However, Qasid Mail uses the store's name when referencing a store in a folder pathname or in other cases. When you define a store in the Store Configuration Dialogue you give the store its name. You will use this name any time that Qasid Mail refers to a store name or a folder path. Copy Message In order to copy a message Qasid Mail needs to know which message to copy, and the destination folder. The message that is copied is the currently displayed message. The destination folder is the folder that is selected in the Folder Panel which is the left panel displaying the Store and Folder tree. Moving Messages In order to move a message Qasid Mail needs to know which message to move, and the destination folder. The message that is moved is the currently displayed message. The destination folder is the folder that is selected in the Folder Panel which is the left panel displaying the Store and Folder tree. When you move a message, it is not expunged from the source folder; it is simply marked for delete. Saving Message Compositions If you compose a message, and attempt to close the window before you have sent the message, Qasid Mail will prompt you to save the message. When you save the message, you will need to specify a Folder path name to which you wish to save the message. Composition Configuration This dialog will allow you to configure items related to the composition of email. Specify the store in which to

save mail messages that have not been sent yet. This allows you to save a message that you are composing for later edit and sending. To send a message that you have saved, open the store that you configure here, and select the message in the list on the right, and use the "Edit Message" command in the "Mail" menu. The mail message will be displayed in a compose window in the state that you saved it. The store name used in this configuration must be a full path name to completely describe the mail folder to use. For this reason, the tree of mail folders is displayed at the bottom of the dialog. Clicking on any of the mail folder icons in the tree will cause the folder's full path name to be entered for you automatically.

7.8.7 Transport Configuration

The transport configuration determines what Qasid Mail uses to send outgoing email. If you never send email with Qasid Mail, and only use it for reading email, then you do not need to worry about this configuration. In WinXP SMTP protocol provider is available to send email. if file contains all personal contact information, including email addresses, addresses, phone numbers, etc. The file is maintained in a format called LDIF (LDAP Interchange Format). The contents of the file can be read with any word processor.

7.8.8 Qasid Mail Configuration

Qasid Mail is configured via a series of dialog boxes. When you first launch Qasid Mail, if it determines that you have not configured the most important properties, it will automatically display the dialogs to configure these items. All of the configuration dialogs are available via the Config menu. Each dialog is discussed via the list below. User Information configures personal information, such as your email address, personal name, reply to address. Signatures configure signatures. Composition Configures where to save messages that you have composed by not sent. Local Manual configures the location of the Qasid Mail on your local disk so you can read the manual while you are off-line. Mail Stores Configures the stores that you wish to use for viewing email. This is where you identify the store that contains your INBOX and other stores you wish to use. SMTP Transport Properties Configures the properties used if the SMTP Transport Protocol is used. Temp files configures directory in which temporary files are created.

7.8.9 Creating Folders

New folders can be created in the Qasid Mail. To do so, simply select a folder in the folder tree panel, so that it is highlighted. Then, select the "Create Folder" menu command. This will display a dialog asking for the name of the new folder. This is not a folder pathname, just the name of the folder. There are also two check boxes indicating whether the folder can contain messages or folders. Once you have filled in the name, and checked a box, click ok and the folder will appear in the folder panel.

7.8.10 Folder Paths

Folder paths are like file paths. They start with the Store Name, followed by a slash, followed by a folder name. If the folder is inside of other folders, slash separated pathname is used to the desired folder.

7.8.11 Mail Folders

Mail folders are the things that contain mail messages. Folders can also contain other folders. If a folder can contain other folders; Qasid Mail will display the folder with a folder icon. If the folder can only contain messages and not other folders, then Qasid Mail will display it with a document icon. Folders can not exist on their own. They must be contained by either a store or another folder. Thus, every folder has a path name that uniquely identifies the stores and folders that you must open to get to the folder identified by the pathname.

7.8.12 Mail Stores

Mail stores are the highest level container in the Java Mail API. Stores are contained by nothing. They are the root of all folder trees. In practice, a Mail Store is synonymous with a mail login or user. A store is specified by a URL, and that url typically identifies a mail host and login. Stores are identified by Java Mail using their URL. However, Qasid Mail uses the store's name when referencing a store in a folder pathname or in other cases. When you define a store in the Store Configuration Dialogue you give the store its name. You will use this name any time that Qasid Mail refers to a store name or a folder path.

7.8.13 Copy Message

In order to copy a message Qasid Mail needs to know which message to copy, and the destination folder. The message that is copied is the currently displayed message. The destination folder is the folder that is selected in the Folder Panel which is the left panel displaying the Store and Folder tree. Moving Messages In order to move a message Qasid Mail needs to know which message to move, and the destination folder. The message that is moved is the currently displayed message. The destination folder is the folder that is selected in the Folder Panel which is the left panel displaying the Store and Folder tree. When you move a message, it is not expunged from the source folder; it is simply *marked* for delete.

7.8.14 Saving Message Compositions

If a message is to be composed and attempt to close the window before it have sent the message, Qasid Mail will prompt to save the message. While saving the message, it will need to specify a Folder path name to which it is wished to save the message.

7.8.15 Composition Configuration

This dialog will allow configuring items related to the composition of email. Specify the store in which to save mail messages that have not been sent yet. This allows to save a message that are composing for later edit and sending. To send a message that have saved, open the store that configure here, and select the message in the list on the right, and use the "Edit Message" command in the "Mail" menu. The mail message will be displayed in a compose window in the state that saved it. The store name used in this configuration must be a *full path name* to completely describe the mail folder to use. For this reason, the tree of mail folders is displayed at the bottom of the dialog.

7.8.16 Transport Configuration

The transport configuration determines what Qasid Mail uses to send outgoing email. If you never send email with Qasid Mail, and only use it for reading email, then you do not need to worry about this configuration. In WinXP SMTP protocol provider is available to send email.

8 Results and Analysis

8.1 Introduction

Testing of software is a very important part of the software development activity. During this stage many modifications are also carried out. As a result of this phase, a programmer may be required to completely rewrite the code. If the results are not as per the expectations of the user, the software may not be accepted by the user. Testing is carried out throughout the development phase. Once developed, software is integrated and a new series of testing is required to be carried out. As such the application software was also passed through a series of tests starting with the testing of the first module. This continued till the software was completed. After the completion of the software, it was tested on line as well.

8.2 Testing Model Used

Testing was carried out using the 'waterfall' model. Here as soon as the very first model was written, it was tested. Subsequently all other modules were tested and then integrated in the overall code and tested. Once the software was ready, testing on LAN and workgroup was also planned and executed. The use of this model did consume lot of time however, most importantly, at each stage of the development, the software was checked and in the end no extra effort was required to correct any error. Thus it reduced the risk factor. Also it was possible to accommodate the changes in the in the design and the functionalities. This also allowed changes to technology and style of coding.

8.3 Stages of Testing.

During the development and integration of the software three levels testing was carried out. In the first level, each module was tested for errors and the output was observed. Any changes required were made. This consumed a lot of time but the time spent did pay in the end. This also allowed refining the overall design of the java programmed. This highlighted the powers and shortcomings of the programming languages and helped in the choice of an appropriate programming language. The second level of testing was carried out to see if the

individual modules fit in the overall coding. This in fact had a direct bearing on the efficiency of the software and needed more care. Any changes required to be made to the modules as a result of this level of testing was carried out immediately. This testing caused a delay in the development of software but the risk of error was reduced to a minimum. Till this level the testing was carried out on a single PC. In the next level, the software was tested on workgroup and also on the LAN in the computer laboratory. By this time the software was completed and integrated. This level of testing not only tested the output and performance of the software but also checked the hardware and software requirements of the machines on which the software was to be run.

8.4 Results and Analysis

Once the software was tested on a single PC, it met all the required parameters. Its performance was satisfactory and the output was as per the expectations. Once the test was carried out in the computer laboratory, the results were not really encouraging. The reason was that the strict administrative rights were enforced and the software was not really allowed to carry out the database access and manipulation operations. This denial of access means the software was not able to connect to the database stored in a internet running on a remote machine. After obtaining the administrative rights, the software was able to connect to remote machines. Yet problems were faced which was mainly pertaining to the networking and network security. The software was tested on a ‘workgroup’ and the results were very much satisfactory. It was able to access and connect register running on remote machines and carrying out the data manipulation operations. The software can give good results if it is placed on the server machine and the administrative and network security restrictions are not a hurdle in the way of the user forbidding the software to access DBMS running on a remote machine and stopping the user from connecting to remote databases. The best way to use the application is to place it on web so that it can be accessed and run from anywhere. This kind of testing was not carried out due to the fact that web hosting was a costly affair and also needed time to carryout the necessary coordination. However the results shown by the software once tested in a workgroup was very encouraging and the same result is expected once the software is run from a website hosting the software.

8.5 Limitations and Future Work

Despite showing good results, the software has few inherent weaknesses. These weaknesses have not been catered for in the design of the software. Moreover few additional features could not be incorporated due to time limitations. First on the list are the key board shortcut keys. All software, whether operating system or any other utility software or special software, always contains key board shortcut keys, which make the job of the user very easy and in certain cases if the mouse stop functioning, the user is not paralyzed and can continue working by making use of the key board shortcut keys. For example while typing a letter in MS Word, a person can save his work by pressing 'CTRL +S' and thus saves time and effort which is required for saving the document by using the mouse. In this way the person ensures that his tempo of work is not broken. This aspect has not been catered for in the and is a shortcoming of the software. This can be addressed to in future works. If this is done it will add to the performance of the software and will help the user to carryout the job very smoothly.

Another limitation of the software is the inability to convert from one DBMS format to the other. The software can convert data stored in any format to a standard format but it cannot convert a table from MS Access format to a table in Oracle or vice versa. Routines for the inter-conversion are available. One need to first convert the data from one format to an intermediate standard format and then this intermediate format is converted to the required format. Since all the data are exported to a standard format by this software the job of inter-conversion is half done. In a future work, the inter-conversion of various data storage formats may be implemented. This will allow easy implementations of the database operations.Yet another limitation have been the lack of the database security. The software is password protected and the DBMS is also secure. The built-in security measures of the .NET ensure security of the database. Need is to enhance the security level to the table and the field level. Thus in any future work this aspect of the database security may be implemented making the database secure up to the table and the field level. This aspect of database security is very helpful in ensuring secure data access and manipulation in case the software is hosted on web. The basic database operations have been implemented and

are well functioning. The result of a search on multi DBMS is stored in a simple format and is printed as such. This results in data redundancy. Duplicity of data is not required. This can be avoided by incorporating the basic functionalities of 'JOIN', 'DIVIDE' and 'INTERSECT'. Also the functionality of 'UNION' is required to be implemented in the software. Any future work may map these four main functions along with the derivative functions.This will deny any chance of duplicity of data. In this software the communication module is responsible to map the queries generated by the SQL Engine to three DBMS i.e. MS Access, SQL Server and Oracle. The allows for extension of the software and in any future work more DBMS can be added to the software. This also depends upon the requirement of a particular user.

8.6 Conclusion

Qasid Mail is Software developed by the team. It is a platform independent product, which is capable of manipulating different databases of various vendors. There was a great need to design such software. Nowadays mail security development can be done using Microsoft Access, SQL Server and SQL/PLSQL and Oracle etc. Each platform has its own features. Access is very user friendly and it takes very less time to build the back-end of database. SQL/PLSQL, Oracle and SQL Server are more secure and handle millions of records. The decision of using any backend platform depends upon the requirement. A small database where security is not of great importance, using ORACLE is a waste of resources, therefore, all these 3 have their own respective applications.

Our application makes the job of learning and implementing the mail security operations much easier. The user of this application will no longer have to learn each of these technologies separately. The user will perform all his operations on this generic platform. All his operations will be mapped to the respective back-end database development platform. How the operations of the platform are mapped, is not the concern of the end user. He just selects the database application onto which he wants to map his operations and all of his operations will be mapped to the selected platform. Moreover, implementing these operations is much straight forward as same interface is used for all database

applications and the platform is very user friendly. By mapping all the desired operations onto any of the back-end platform using a common interface user does not have to learn all the backend development tools. Only this software can do their job much quicker as it is a very user-friendly interface. Moreover the software has been developed using a very powerful and emerging technology, the MS Visual Studio.NET. The software can be web hosted and can be accessed and run from anywhere.

BIBLIOGRAPHY

[1] ASP.NET Database Programming Weekend Crash Course by Jason Butler and Tony Caudill.

[2] Progrmming Microsoft.NET By Prosise Jeff.

[3] SAMS, Teach Yourself .NET Windows Forms in 21 Days, by SAMS series

[4] Essential ADO.NET By Bob Beauchemin explains the database access part of the .NET framework.

[5] ASP.NET by Example by Steven A. Smith

[6] Professional ASP.NET 1.0 Special Edition a wrox series book.

[7] Dietel-C Sharp How to programme by Dieteland Dietel

[8] *http://www.c-sharpcorner.com/*

[9] *http://www.codeguru.com/*

[10] *http://www.dotnet-fr.org/*

[11] *http://microsoft.com*

[12] *http://msdn.microsoft.com/net/*