5-2001

Distribution List Maker Program with Inter-User Capabilities between Universities and Colleges in the Tennessee Board of Regents School System.

Allan Richard Anderson
East Tennessee State University

Distribution List Maker Program with Inter-User Capabilities between Universities and Colleges in the Tennessee Board of Regents School System

---

A Thesis
Presented to
the Faculty of the Department of Computer Science
East Tennessee State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science

---

by
Allan R. Anderson
May 2001

Dr. Terry Countermine, Chair
Dr. Martin Barrett
Dr. Phil Pfeiffer

Keywords: Pegasus Mail, Netscape Messenger, Outlook, Eudora Lite,
Outlook Express, E-mail, Distribution Lists

ABSTRACT

Distribution List Maker Program with Inter-User Capabilities Between Universities and Colleges in the Tennessee Board of Regents School System

by

Allan R. Anderson

E-mail is an important tool for faculty and staff at the university, college, department, and instructor levels. E-mail is a useful medium in the academic setting for corresponding at all levels. Instructors e-mail students about assignments, lectures, and urgent information for example, postponed classes and changes in the schedule. In addition e-mail is used to let potential students know about job opportunities. Other routine uses for e-mail include group communications within academic committees and groups of students collaborating on projects.

Most users of e-mail who send messages to multiple recipients enter each recipient's e-mail address into the TO: field individually or enter the name and e-mail address into a distribution list individually. Both methods are time consuming.

This thesis describes a tool for facilitating the use of e-mail for classroom management. This tool, List Maker converts class roll listings to e-mail distribution lists for five common e-mail clients: Pegasus Mail, Eudora Lite, Netscape Messenger, Microsoft Outlook Express, and Microsoft Outlook 97/98. List Maker also converts address books and distribution lists from one e-mail client to another.

The List Maker program has been adopted for use in selected departments within ETSU. A survey of the program’s users indicates that List Maker made it easier for users to create distribution list from class rolls. Efforts to distribute List to other Tennessee Board of Regents (TBR) colleges, unfortunately, have not yet succeeded due to a lack of uniform computing platforms and e-mail policies in the TBR.

ACKNOWLEDGMENTS

I would like to thank my Thesis Committee of Dr. Terry Countermine (Thesis Chairperson), Dr. Marty Barrett, and Dr. Phil Pfeiffer. Without their help this project would not have been possible. I would like to thank Dr. Pfeiffer for his editing and proofing of this paper.

Independent Study Advisors

I would like to again thank Dr. Countermine for approaching me and asking to see if this project was feasible. Along with Dr. Pfeiffer's and Dr. Countermine's advice and guidance, it was determined that this project was very feasible to implement and make a working program.

I would also like to thank the following: Faculty and Staff at Walters State Community College, Faculty and Staff at Northeast State Technology Community College, and Faculty and Staff at East Tennessee State University. A special thanks is due to the Office of Information Technology in making this project possible and in with meeting with representatives from NSTCC.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

E-mail is an important tool for faculty and staff at the university, college, department, and instructor levels. E-mail is a useful tool in the academic setting for corresponding with students about assignments and lectures. E-mail is also a quick and easy method of notifying students of urgent information: for example, postponed classes and changes in the schedule. A third use for e-mail is letting potential students know about job opportunities. Other routine uses for e-mail include group communications within academic committees and groups of students collaborating on projects.

An e-mail mailing list is a set of related addresses. These addresses are typically stored in files created by e-mail clients in response to user commands. Mailing lists can be generated by entering students' e-mail addresses into the message: however, there is an easier way of sending the same message to multiple students; with an e-mail mailing list or distribution list.

Currently ETSU staff and faculty must create e-mail distribution lists manually. An instructor with average typing skills and a basic knowledge of an e-mail client spends approximately an hour per class creating a workable database for communication with students. The present system of common mail clients does not allow users to set up mailing lists automatically, for example from textual lists of user names.

This thesis develops a partial solution to the problem of automated mailing list generation. The software described here creates e-mail distribution lists for commonly used e-mail clients used in the Tennessee Board of Regents (TBR) system. These e-mail clients are Netscape Messenger (part of Netscape Communicator 4.0 and above), Microsoft Outlook Express (obtained with Microsoft Internet Explorer), Microsoft Outlook 97/98 (part of Microsoft Office 97), Pegasus Mail and Eudora Light.

Obstacles to Overcome

Four key obstacles were addressed during the course of this work. These obstacles are the conversion of the information from the Student Information System; incompatibilities between file formats for e-mail clients; design of the user interface; and the lack of ongoing cooperation among schools in the Tennessee Board of Regents.

Student Information System (SIS) File Conversion

SIS is a program used in the TBR system to maintain records of TBR students. Student records are kept in several password-protected databases within the SIS program. A UserID and password determines the user's level of access to SIS data. Students, for example, have access for registration, fees, class schedules, and miscellaneous personal data. Some staff can only *clear holds* (i.e., remove registration restrictions) on a student record, while others can alter other data. For example, the ETSU staff in the Center for Adult Programs and Services (CAPS) can register students and clear student holds but can not add an extra seat for a computer science class or clear the Measles, Mumps, and Rubella (MMR) hold placed on students that must have this vaccination.

Special programs written by ETSU Office of Information Technology (OIT) give SIS users access to data that they need to have but could not otherwise obtain. One such program extracts the class roll information from the various databases that SIS uses and e-mails the class rolls to the requester. One of the programs developed for this thesis reads the text from the body of this class roll e-mail message and converts it to a Microsoft Access database. An example of an e-mail message generated by this program is shown in Figure 2-1.

File Format Conversion

The e-mail clients considered in this thesis (Netscape Messenger, Microsoft Outlook Express, Microsoft Outlook 97/98, Pegasus Mail, and Eudora Light) use different formats for storing distribution lists and e-mail addresses. Netscape Messenger and Microsoft Outlook Express use the Lightweight Directory Access Protocol Data Interchange Format (LDIF) for their address book import. Pegasus Mail and Eudora Light use a text file for their distribution list and address book. However, the layouts of the Pegasus and Eudora files differ. Microsoft Outlook 97/98 uses a binary file that stores the address book and distribution lists.

Interoperability was achieved by developing routines for converting between address book and distribution list formats. Part of this work involved deciphering file formats, when feasible. One format that resisted reverse engineering was Netscape Address Book, which is stored as binary file. Netscape, however, uses the LDIF format to allow imports into its address book.

User Interface Design

One of the project's goals was to determine automatically at program startup what e-mail clients its user might currently be using. The program then offers the user a choice of supported clients in response to a request for a distribution list.

A second issue in user interface design was trying to prevent data entry errors by the user. This goal was accomplished by supplying default values as much as possible and by having the user select information from a dialog box where feasible. For example, when a user creates a distribution list for Pegasus Mail there are three required fields: Long Name for Distribution List, Import File, and Course Number/Dept. Name. If the user selects Import file, the program displays a list of available database files for the user to select. After selecting the database, the Import File and Course Number/Dept. Name fields are filled with the correct information. All that is left for the user to enter is a name for the distribution list in the Long Name for Distribution List. When the user enters a name for this field, the FileName and To: Field are filled in for the user.

Lack of Close Cooperation among Tennessee Board of Regents (TBR) Institutions

All TBR institutions have their own unique political environment and working relations, or lack thereof, with other institutions. One of the problems in working with other TBR institutions is communicating the potential benefits of a project to the school, staff, and students. Another problem is the lack of time and resources to work on additional projects. These problems will examined at in more detail later in this thesis.

What is the E-mail Distribution List Maker Program?

This thesis's key result is an E-mail Distribution List Maker. The E-mail Distribution List Maker is a program that takes information sent to the user from the SIS interface and creates distribution lists. The program provides the user with a web interface for accessing the Internet interface to SIS. After the class roll or departmental list is requested by the user, the information is then e-mailed to the user by the OIT class roll extraction program. The List Maker program uses an e-mail interface to capture this class roll from the user's inbox automatically, as the list maker is running.

Once the List Maker retrieves the class list, the program converts the SIS file to a Microsoft Access database. The database is then used to extract the information needed to create the distribution lists for the e-mail programs considered in this project. The List Maker can make distribution lists for the following e-mail programs: Microsoft Outlook Express, Microsoft Outlook 97/98, Pegasus E-mail, Eudora Lite 4.0, and Netscape Communicator 4.0.

The List Maker can also do format conversions on e-mail clients' address books and distribution lists. Any of the e-mail clients' address books and distribution lists can be converted from any e-mail client to any other e-mail client considered in this project. In some cases, a client's address book must be exported using the mail program's command interface before being converted to a different format. For example a, Netscape Communicator user has to export an address book to a Lightweight Directory Access Protocol Data Interchange Format (discussed in chapter 2) before it can be converted to Pegasus Mail. Also Microsoft Outlook Express and Microsoft Outlook 97/98 users must export their address book to comma-separated value files before they can be converted to another address book format.

Preview of the Remaining Document

Chapter two describes the various file formats used in this thesis project. Some of the file formats to be examined are the SIS file format, the database format used to store the information, and the e-mail client's address book and distribution list formats. Chapter two will also look at the conversion of the various e-mail formats to Microsoft Outlook 97/98 using the Messaging Application Program Interface and the Collaboration Data Objects.

Chapter three describes program design. Some of the topics included in chapter three will be the program's requirements and technologies used in creating the distribution lists and in the conversion routines. In addition, the third chapter will look at the details of how to write directly to the Microsoft Outlook 97/98 Personal Address Book.

Chapter four describes the verification of the design of the program and the time saved, if any, in creating distribution lists with the program compared to the current method of creating distribution lists. Chapter five describes the current status of the program and project. Chapter six projects possible future work on the project, and the last chapter gives acknowledgments.

CHAPTER 2

FILE FORMATS

Overview

This chapter describes the formats of the files and databases manipulated by the List Maker. This includes the file formats of the Student Information System (SIS), and the following e-mail client file formats: Pegasus Distribution list, Pegasus Address Book, Eudora 4.0 Lite, Lightweight Directory Access Protocol Data Interchange Format (LDIF), Netscape Communicator Address Book, Microsoft Outlook Express and Microsoft Outlook 97/98. This chapter also examines the specific problems presented by these e-mail clients along with the limitations of the Eudora 4.0 Lite mail program.

SIS File Format

The E-mail Distribution List Maker program transforms class rolls generated by OIT's web interface into e-mail address books. Figure 2-1 shows a sample SIS file.

An OIT generated class roll, referred to here as an SIS file, is a text file sent to the requester via e-mail. List Maker converts SIS files to Microsoft Access databases before generating e-mail distribution lists. Table 2-1 shows the schema for the converted SIS file. Each record in the conversion database represents one record in the SIS file. This conversion simplifies List Maker design by allowing the use of Structured Query Language (SQL) for list generation.

The algorithm for doing the conversion is as follows. The List Maker program searches the file to find COURSE:. After finding the COURSE: field the program uses the text following the COURSE: as the file name for the database. The List Maker program also uses the COURSE: field for the table name. Next, the program searches the file to find the file's column headers and uses these headers for the field names in the database. The program then reads the rest of the file to extract the information to be entered into each field. The use of a fixed format layout makes it easy to break the report into distinct data items. See figure 2-1.

SELECTION = TERM: 1999 FALL , COLLEGE: ALL, CLASS: ALL, MAJOR: ALL , COURSE: CSCI5957201

STUDENT ID	NAME	EMAIL ADDRESS	COLL	CLASS	MAJOR	N/S	REG STATUS
111-11-1111	OWENS, JOHN CIDRIC	ZJCO5@etsu.edu	UD	SO	UDEC	none	Dropped-Non Payment
111-11-1111	GIBSON, MARY LYNN	ZMLG16@etsu.edu	NU	FR	PBSN	none	Enrolled
111-11-1111	YOUNG, MELISSA JAMELLE	ZMJY1@etsu.edu	AS	JR	PSYC	none	Enrolled
111-11-1111	SHORTT, CHRISTOPHER BRADLEY	ZCBS9@etsu.edu	AT	FR	ENTC	none	Enrolled
111-11-1111	NASH, SANDRA LYNN	ZSLN4@etsu.edu	BU	SR	ACCT	none	Enrolled
111-11-1111	HANCOCK, NEOYSHI UMEKA	ZNUJ11@etsu.edu	ED	SR	ECDV	NAME	Enrolled
111-11-1111	MONTGOMERY, HEATHER NICOLE	ZHNM3@etsu.edu	AS	FR	PMED	none	Enrolled
111-11-1111	FINCH, TERESA ELAINE	ZTEF4@etsu.edu	AT	SR	AHSC	none	Dropped; Retain
111-11-1111	REYNOLDS, TRENT WARDEN	ZTWR1@etsu.edu	AS	SR	ENGL	none	Dropped; Retain
111-11-1111	PILK, LUCAS ALLEN	ZLAP15@etsu.edu	UD	FR	UDEC	none	Enrolled
111-11-1111	CAMPBELL, ADAM CHARLES	ZACC15@etsu.edu	AS	JR	CJCR	none	Enrolled
111-11-1111	CARDER, AMANDA DAWN	ZADC17@etsu.edu	AS	FR	PSYC	none	Enrolled

Figure 2-1 File Format from SIS. Field breaks are at columns 12, 47, 73, 78, 83, 89, 95, and 115.

Table 2-1 Microsoft Access 97 Database File Layout

Field Name	Data Type	Field Size	Required	Allow Zero Length
Class-ID	AutoNumber	Long Integer		
ID	Text	11	No	Yes
Name	Text	40	No	Yes
Email	Text	30	No	Yes
Coll	Text	2	No	Yes
Class	Text	4	No	Yes
Major	Text	4	No	Yes
N/S	Text	4	No	Yes
Reg	Text	40	No	Yes

Pegasus Distribution List File Format

Each Pegasus Distribution list is stored as a separate text file and formatted as shown in Figure 2-2. Pegasus allows a user's name to be paired with each e-mail address if enclosed in quotation marks, with at least one space after the e-mail address.

```
\TITLE UIT Section 564 Bristol
\SENDER "UIT Section 564 Bristol" <@>
\NOSIG Y
ZMJJ2@etsu.edu              "BLANKENSHIP MICHELLE JESSEE"
ZMTB12@etsu.edu             "BOATWRIGHT MICHAEL THEODORE"
ZNWC1@etsu.edu              "COLE NATHAN W"
ZCGC4@etsu.edu              "COSTAGLIOLA CLARA GIUSEPPINA"
ZTDH8@etsu.edu              "HINSHAW TOMMY D."
ZDLT1@etsu.edu              "TRIPLETT DANNY LEN"
```

Figure 2-2 File Format of Pegasus Distribution List

Field Descriptions

The \TITLE line gives the title of the distribution list. This title appears in the distribution list dialog box in Pegasus Mail. The optional \SENDER line suppresses the display of all target e-mail addresses in the receiver's e-mail. This example uses the fake e-mail address of "UIT Section 564 Bristol" <@> to suppress the address listing. The \NOSIG Y field tells Pegasus mail

not to use a signature in the e-mail. These three fields are the only fields that the distribution list maker puts in the file by default. However, Pegasus mail only requires the \TITLE and \NOSIG fields.

Four other optional fields may appear in the header. The \REPLYTO field allows the sender to preset the e-mail address for message replays. For example, \REPLYTO zara1@etsu.edu requests a reply at zara1@etsu.edu. The \READING field, if present, requests that the receiver return a message confirming that the receiver has accessed the mail. The \DELIVERY field, if present, requests that the receiver return a message confirming that the message was delivered. Finally, the \URGENT field, if present, causes Pegasus to send at the highest available priority. Pegasus Mail displays urgent messages at the top of the new mail folder in red text and sends a different new mail notification to the recipient to show that the message is urgent.

Pegasus Address Book File Format

Pegasus maintains its address books in separate files. Every address book is implemented as a pair of files: a .pmr file that contains information in the address book and an index file with an extension of pm!. Both files are in a binary format.

Table 2-2 shows the structure of the index file. A hex number is representing the number of the records for the end of record marker. The NULL is an ASCII 00.

Table 2-2 Pegasus Address Book Index File Format (*.pm!)

Name	Start Hex	End Hex	Start ASCII	End ASCII
NULL	00h	01h	0	1
Key	2h	Ch	2	12
NULL	Dh	Fh	13	15
Name	10h	32h	16	50
NULL	33h	33h	51	51
Phone	34h	3Eh	52	62
End of Record	4Eh	4Eh	78	78
NULL	4Fh	4Fh	79	79
Next Record Starts	50h		80	

Table 2-3 shows the file layout of the *.pmr file or the address book file. This file has a header that consists of the Title and the string of Nulls that follow. This header is only in the file once. Each .pmr file record holds 458 bytes. Empty fields are filled with nulls to maintain record size.

Table 2-3 Pegasus Address Book File Format (*.pmr)

Name	Start HEX	End HEX	Start ASCII	End ASCII
Title	0h	2Bh	0	43
NULL	2Ch	81h	44	129
Name	82h	A8h	130	168
NULL	A9h	A9h	169	169
Of	Aah	D0h	170	208
NULL	D1h	D1h	209	209
Key	D2h	DCh	210	220
NULL	Ddh	DDh	221	221
Street	DEh	118h	222	280
NULL	119h	119	281	281
Postal	11Ah	154h	282	340
NULL	155h	155h	341	341
Phone	156h	16Ch	342	364
NULL	16Dh	16Dh	365	365
Fax	16Eh	184h	366	388
NULL	185h	185h	389	389
Notes	186h	1D4h	390	468
NULL	1D5h	1D5h	469	469
E-mail	1D6h	238h	470	568
End of Record	239h	24Bh	569	587
Next Record Starts	24Ch		588	
End of File 10 NULLS				

Eudora Lite 4.0 File Format

Eudora Lite 4.0 (Eudora) captures the entire mail database in a text file named nndbase.txt. Special characters in nndbase.txt serve as field and record delimiters. Eudora stores this file in the directory where the user installed Eudora. Eudora also supports an index file, nndbase.toc, which is rebuilt every time that the address book gets saved. See Figure 2-3 for a sample of the Eudora file structure.

```
alias csci1101001
ZPAA1@etsu.edu,ZICM1@etsu.edu,ZJAB22@etsu.edu,ZRMB11@etsu.edu,ZLEC6@etsu.e
du,ZABC6@etsu.edu,ZACD1@etsu.edu,ZADF1@etsu.edu,ZSNF10@etsu.edu,ZSMG3@ets
u.edu,ZJMH8@etsu.edu,ZAJH6@etsu.edu,ZCAH5@etsu.edu,ZJMH13@etsu.edu,ZJLJ7@ets
u.edu,ZKCK1@etsu.edu,ZCSL2@etsu.edu,ZMBL1@etsu.edu,ZARL5@etsu.edu,ZDCM11@
etsu.edu,ZAKM4@etsu.edu,ZFMM12@etsu.edu,ZANO2@etsu.edu,ZJEP21@etsu.edu,ZTL
Q2@etsu.edu,ZKNR12@etsu.edu,ZSRS13@etsu.edu,ZSCS12@etsu.edu,ZMBT1@etsu.edu,
ZJLW20@etsu.edu
note csci1101001 ARRIARAN ♥AUCHTERLONIE ♥BATES ♥BLEVINS ♥CAMPBELL
♥COE ♥DEELY ♥FEATHERS ♥FENTRESS ♥GRAY ♥HAASE ♥HEATH ♥HEFNER
♥HOFFMAN ♥JENKINS ♥KNODE ♥LADD ♥LIRIO ♥LUCAS ♥MEREDITH
♥MILLER ♥MULLINS ♥OVERBAY ♥PATRICK ♥QUESENBERRY ♥RILEY ♥SIMS
♥SMITH ♥TISDALE ♥WINSTON
alias "Allan Anderson" zara1@etsu.edu
note "Allan Anderson" Student at ETSU
alias "Sharon Anderson" sharona@xyz.net
note "Sharon Anderson" Wife of Allan Anderson
alias "New Group" zara1@etsu.edu,sharona@xyz.net
```

Figure 2-3 Eudora Address book File Format **NOTE:** ♥ is ASCII character 03

Every record in Eudora Address Book stores an alias for an individual user or a mailing list. The first record in Figure 2-3 is a distribution list for a csci1101 class. The list itself consists of a series of comma-separated e-mail addresses. The optional note field that follows the list associates optional information with each e-mail address on the list. If the user creates the alias using the distribution list maker program, the note contains the last names of the individuals in the alias field. These names, furthermore, appear in the same order as Eudora lists the e-mail addresses in the alias field. The ASCII character 03 (♥) separates each name. This allows the name to be on a separate line to match with the alias field. Each Eudora entry is limited to 273 items.

Lightweight Directory Access Protocol Data Interchange Format

The Lightweight Directory Access Protocol Data Interchange Format (LDIF) file format is used by Netscape Communicator and Microsoft Outlook Express to import addresses into their address books. An LDIF file contains individual e-mail addresses and distribution lists. The address books for Communicator and Outlook Express must have the individual's e-mail address in the address book before the e-mail address can be included in a distribution list. See Figure 2-4 for a sample of the LDIF file format.

List Maker makes the following assumptions about the LDIF address book format. The LDIF file must include fields for distinguished name, common name, mail, objectclass: top, and objectclass: person, surname. The distinguished name consists of the common name and mail separated by a comma (i.e., dn: cn=Allan Anderson,mail=zara1@etsu.edu). The common name consists of the first name and the last name (i.e., cn: Allan Anderson). The mail field is the e-mail address of the person (i.e., mail: zara1@etsu.edu). The objectclass field designator is used twice: once to specify that the record represents a top level object (objectclass: top) and once to specify that the record represents a person (objectclass: person).

List Maker assumes that every personal record (i.e., object of type person) has at least two attributes: common name and surname. The surname is the last name of the person in the address book (i.e., sn: Anderson). All other fields are optional.

Finally, List Maker assumes that each distribution list has at least five attributes: distinguished name, common name, objectclass: top, objectclass: groupOfNames, and member. The distinguished name for a distribution list is the common name (i.e., dn: cn=Thesis). The common name for a distribution list is the name of the distribution list (i.e. cn: Thesis). The objectclass field designator is used twice: once to specify that the record represents a top level object (objectclass: top) and once to specify that the record represents a group of names (objectclass: groupOfNames). The class groupOfNames requires the attribute common name. The member field contains the common name and mail field. The common name and mail entries must be in the address book before they can be in a distribution list. The format for the member field is (member: cn=Allan Anderson,mail=zara1@etsu.edu).

```
dn: cn=Allan Anderson,mail=zara1@etsu.edu
modifytimestamp: 20000328211110Z
cn: Allan Anderson
mail: zara1@etsu.edu
xmozillausehtmlmail: FALSE
givenname: Allan
sn: Anderson
objectclass: top
objectclass: person

dn: cn=Display Name,mail=e-mail
modifytimestamp: 20000328211047Z
cn: Display Name
mail: e-mail
xmozillausehtmlmail: TRUE
givenname: First
sn: Last
xmozillanickname: nickname
o: Organization
locality: City
st: State
description: Notes
title: Title
streetaddress: Address
postalcode: Zip
countryname: Country
telephonenumber: work
homephone: home
facsimiletelephonenumber: fax
pagerphone: pager
cellphone: cellular
ou: Dep
homeurl: URL
xmozillaanyphone: work
objectclass: top
objectclass: person

dn: cn=Thesis
cn: Thesis
objectclass: top
objectclass: groupOfNames
member: cn=Display Name,mail=e-mail
member: cn=Allan Anderson,mail=zara1@etsu.edu
```

Figure 2-4 LDIF File Format

Netscape Address Book File Format

Netscape Address Book uses a binary file format to store the data for the address book. For this project, the precise format was not deciphered. Rather, Netscape's import and export features were used to access and update address books.

Microsoft Outlook Express File Format

Microsoft Outlook Express address book uses a binary file format to store the data for the address book. For this project, the precise format was not deciphered. Rather, Microsoft Outlook Express import and export features were used to access and update address books.

Microsoft Outlook 97/98 File Formats

Microsoft Outlook 97/98 uses a binary file format to hold its address files. Outlook 97/98 also supports two different address books. The one that most people see is the Contacts. However, distribution lists cannot be placed in the Contacts' folder. Another address book, the Personal Address Book (PAB), allows the use of distribution lists and individual addresses.

Decoding the file format for these two address books proved difficult, even with a binary file viewer. However, Microsoft Visual Basic 6.0 allows users to add individual addresses and distribution lists to the PAB. For VB-based PAB operations to work, the user of Outlook 97/98 must have a PAB installed on their computer, which, in turn, requires administrative privileges. Creating a distribution list or an individual address in the PAB is done via the Messaging Application Program Interface (MAPI) and the Visual Basic 6.0 Collaboration Data Objects (CDO).

Messaging Application Program Interface (MAPI)

MAPI is a layer between user programs and Messaging Providers that lets programmers write programs without concern for the underlying Message Provider type. Microsoft divides Messaging Providers into three basic categories: Message Store Providers, which supply message storage, organization, and retrieval facilities for a message system; Address Book Providers, which supply message addressing and distribution lists facilities to the messaging client, and Message Transport, which moves messages between messaging clients. Before using

the MAPI functions, the MAPILogon Application Program Interface (API) function must be used to connect to a valid message store. (Bockmann, Klander, and Tang, (1998) and MAPI)

Collaboration Data Objects (CDO)

Collaboration Data Objects is a technology for building messaging or collaboration applications. Microsoft has designed CDO to simplify the creation of applications with messaging functionality, as well as to add messaging functionality to existing applications. CDO library version 1.2.1 exposes messaging objects for use by Microsoft Visual Basic, C/C++, and Microsoft Visual C++. The CDO library allows the programmer to quickly and easily add to Visual Basic applications the ability to send and receive mail messages, and to interact with folders and address books. However, before these features can be used, a MAPI client must be installed. The CDO library is a layer built on top of MAPI to allow Microsoft Visual Basic and C/C++ to have access to the MAPI features. The CDO library, however, does not offer access to all of MAPI. Microsoft designed CDO primarily for clients and it is not suitable for service providers. (CDO - Microsoft Platform SDK Messaging and Collaboration (CDO 1.2.1)) An example of how CDO accesses Outlook 97/98 Personal Address Book can be found on Microsoft web site at http://support.microsoft.com/support/kb/articles/q178/7/87.asp.

Conversion of E-mail File Formats from One Program to Another

E-mail clients evaluated for this project have varying import abilities. Table 2-4 shows the import features of the e-mail clients examined during this project. For the programs that do not import data files generated by other e-mail clients, a file conversion routine was written to provide interoperability. The balance of this section discusses strategies used for List Maker's conversion routines, along with any limitations imposed by the clients proper.

Eudora Lite

Eudora Lite limits the number of entries in each address book to 273. If the user converts a distribution list with more than 273 entries to Eudora Lite, List Maker's conversion routine breaks the list into multiple address books. The user of Eudora Lite can identify the multiple address books by the name of the address book and a number after the name of the address book.

A set of related address books, for example, might have names like “csci,” “csci 1", and “csci 2". Csci and csci 1 would have 273 entries in each address book and csci 2 would have the remainder of the address up to 273 entries.

Eudora Lite neither imports any other e-mail clients’ address books or allows the export of its own address book. The List Maker Program allows the user to import and export Eudora Lite address books to other e-mail clients considered for this project.

Outlook 97/98

The CDO and MAPI interfaces allow the conversion of any e-mail address book in a known format into Outlook 97/98 - accessible PAB. This is how the List Maker program converts unsupported e-mail programs to Outlook 97/98. The CDO and MAPI interface can also be used to convert from Outlook to another e-mail address book.

Netscape

Netscape’s address book format was not decoded for this project. The user must first export the Netscape address book to the LDIF file format to be imported into the other e-mail program or for The E-mail Distribution List Maker Program to convert it to another e-mail program.

Pegasus Mail

Pegasus Mail does not recognize standard e-mail address book formats. Pegasus, however, does import and export a tagged file format and a tabbed file format. The e-mail programs considered in this project do not support Pegasus file formats. The E-mail Distribution List Maker Program converts the Pegasus address book to other e-mail programs’ address book formats.

Other troublesome features of Pegasus include its use of a separate file for distribution lists and its lack of import or export support for distribution lists. The List Maker program allows conversion of the distribution list to other e-mail programs considered in this project.

Outlook Express

Outlook Express exports a comma-separated file. The List Maker program is used to convert the comma-separated file to the other e-mail address books. Outlook Express imports a LDIF file. The user needs to use the List Maker program to convert the e-mail programs' address books that Outlook Express do not import to a LDIF file and then import this file into Outlook Express.

Table 2-4 Import Formats Supported by Email Programs

Import Data Format	Outlook 98	Outlook 97	Outlook Express 5	Pegasus	Eudora Lite	Netscape Comm.
Comma Separated Values (DOS and Windows)	Yes	Yes	Yes	No	No	Yes
Eudora Light and Pro (mail and address book)	Yes	No	Yes	No	No	Yes
LDIF	No	No	Yes	No	No	Yes
Microsoft Access	Yes	Yes	No	No	No	No
Netscape Mail (mail and address book)	Yes	No	Yes	No	No	NA
Netscape Messenger (mail and address book)	Yes	No	Yes	No	No	Yes
Outlook 97/98	NA	NA	No	No	No	Yes
Outlook Express (mail and address book and rules)	Yes	No	NA	No	No	Yes
Personal Address Book	Yes	Yes	Yes	No	No	No
Pegasus	No	No	No	NA	No	No

(Leonhard, Hudspeth, and Lee,1998)

CHAPTER 3

PROGRAM DESIGN

Overview

This chapter discusses the requirements for the E-mail Distribution List Maker and Conversion Program. It also describes how the database is created and the Structured Query Language (SQL) is used to extract the information from the database. In addition, it discusses some techniques used in retrieving the e-mail from the SIS Internet interface. Last, there is a brief look at the technology used to create a distribution list in the Personal Address Book of Microsoft Outlook.

Typical User

The requirements for the List Maker program were developed by assessing the needs of List Maker's typical users. The typical users of the List Maker program during the evaluation phase of this project have been secretaries of departments, professors, and directors of centers. These users know how to use basic programs on their computer, including their e-mail client, word processing programs, and the SIS program. A few users, professors who teach computer science classes, are considered expert computer users.

The typical user in the academic environment would most likely be using an e-mail client that their Information Technology office would have installed on his/their computer or the e-mail client that came with the computer. They could also be using one of the free e-mail clients that are available. Common free e-mail clients include Microsoft Outlook Express (free with Microsoft Internet Explore), Netscape Messenger (free with Netscape Communicator), Pegasus Mail (www.pegasus.usa.com), and Eudora Lite (www.eudora.com). A final client, Microsoft Outlook, is included in the Microsoft Office suite, which is typically installed on new computers.

User Requirements

The following top level requirements for List Maker are dictated by the profile of the typical user as well as the needs of the academic environment.

- An access e-mail account (ETSU only) for authenticating the user. Other schools might use other forms of authentication

- Simple and intuitive operation of menus and toolbars for function execution
- Ability to select the e-mail program for which to create a distribution list
- Ability to maintain the username, e-mail server, and password of the user e-mail account in a text file and encrypted binary file (if wanted)
- Ability to launch Web Browser from within the program to request the class rolls and departmental listings
- Ability to receive SIS generated e-mail from within the program
- Ability to create distribution lists for the different e-mail clients
- Ability to name the distribution lists so that the user can find them in their e-mail client
- Ability to convert address books and distribution lists from one e-mail program to another of the listed e-mail programs

Program Requirements

The following supplemental requirements for List Maker are dictated by the computing platforms in use in the TBR.

- Ability to determine if Microsoft Outlook 97/98 and the cdo.dll (dynamic link library) files are installed on the user's computer
- Ability to find and remember the directory of Eudora Lite and Pegasus Mail Programs
- Ability to maintain the location of the URL of the Web interface in a text file
- Ability to encrypt/decrypt the user e-mail password if the user wants to store his/her e-mail password
- Ability to request information from using the web interface
- Ability to retrieve information from the SIS program
- Ability to convert a SIS file to a Microsoft Access Database
- Ability to extract the information needed to create the distribution lists for the following e-mail programs: Pegasus Mail, Eudora Lite, Netscape Communicator Messenger, Microsoft Outlook Express, Microsoft Outlook 97, and Microsoft Outlook 98

Detailed Discussion of the User Requirements

The user requirements are discussed below. The discussion is organized according to the steps involved in using List Maker to create a distribution list. A more detailed explanation of how these requirements were met is given in the design section of this chapter.

Startup

On startup the List Maker program verifies that Microsoft Outlook and the cdo.dll are installed on the user's machine. Microsoft Outlook needs to be installed to use the MAPI functions to gain programmed access to Microsoft Outlook. The List Maker program uses the MAPI functions to login to Microsoft Outlook. If Microsoft Outlook is not installed and the user tries to create a distribution list or convert an address book to Microsoft Outlook, an error is generated when the List Maker program tries to login to Microsoft Outlook. The error causes the List Maker program to lock up, forcing the user to shut it down through the task manager. In the worst case, this error could lock up the user's computer, forcing the user to turn off his/her computer and restart the computer to clear the error.

The cdo.dll needs to be installed to use the Personal Address Book of Microsoft Outlook. If this dll is not installed and the List Maker program tries to create a distribution list or convert a distribution list to Microsoft Outlook, an error is generated. The error causes the List Maker program to lock up, forcing the user to shut it down through the task manager. In the worst case, this error could lock up the user's computer, forcing the user to turn off his/her computer and restart the computer to clear the error. This search is performed whenever List Maker is started because Microsoft's new security patch for Microsoft Outlook 98 and 2000 removes cdo.dll. After the List Maker program determines whether the files are installed, the user can select which e-mail client he/she wishes to make a distribution list

E-mail Client Selection

The List Maker program allows the user to select the type of e-mail client for which they are going to make a distribution list. In the List Maker program the user selects Pegasus Mail, Netscape Communicator Mail & Microsoft Outlook Express, Eudora Lite, or Microsoft Outlook

97, 98, or 2000. The choices that the List Maker provides for the user vary, according to the files List Maker discovers on program startup. A specific e-mail client must be specified because each of these clients uses a different file format for storing distribution lists.

The List Maker program can write the distribution list directly to the following e-mail clients: Pegasus Mail, Eudora Lite, and Microsoft Outlook. These e-mail clients were selected because of their common use in the ETSU academic environment.

Setup

The List Maker program allows the user to store information about the location of Pegasus Mail and Eudora Lite, the e-mail server name, username, the e-mail password, the Uniform Resource Locator (URL) of the SIS web interface, and any combined classes. This information is entered only once, except when creating a single list for two or more sections of the same or related classes (i.e. CSCI4717 and CSCI5717, undergraduate and graduate computer architecture). Mailing lists for these combined classes may change from semester to semester and should, as a rule, be re-entered. This stored information allows the user to avoid reentering same information every time they use the List Maker program.

If the user chooses to supply an e-mail password, entering an e-mail password is optional. List Maker encrypts the password before storing the password in a binary file on the user's computer. Otherwise, if the user decides not to store his/her e-mail password, he/she must enter the password each time he/she uses the built-in e-mail client to receive the information generated by the web interface of SIS.

A more detailed discussion on how the location of Pegasus Mail and Eudora Lite is determined and how the e-mail password is encrypted/decrypted is given later in this chapter.

Requesting Class Rolls

After the setup step is completed, List Maker allows the user to request class rolls from SIS via an Internet access program. At ETSU, List Maker invokes the SIS Internet access program using a built in web browser based on Microsoft Internet Explorer. Microsoft Internet Explorer was chosen because of its integration with Microsoft Visual Basic and availability: IE is normally installed on every personal computer. When List Maker displays the Internet access

program's web page, the user must enter their staff e-mail username and password before submitting requests. This is to keep unauthorized individuals from gaining access to student information.

A more detailed explanation of why Microsoft Internet Explorer was used is given later in this chapter.

Receiving the Information

Information from SIS is returned to the user's e-mail inbox via e-mail. List Maker retrieves this e-mail but does not delete it. This is done in the interest of maintaining the integrity of the inbox. The List Maker program allows the user to receive SIS information via an e-mail client built into the List Maker program. This was done to allow the user of the List Maker program to request the information, receive the information, convert the e-mail, and create the distribution list all in one program. A more detailed explanation of List Maker's interaction with the e-mail client is discussed later in this chapter.

Creating the Database

List Maker program converts information obtained from SIS into a database in order to simplify list generation. Database creation is done as another step in List Maker operation and requires no intervention by the user. Once the database is created, the program allows the user to create a distribution list by querying the database via the Structured Query Language (SQL). The SQL is hidden from the user; the SQL statements needed to query the database are created through the user's choice on the query database screen of the List Maker program.

What is SQL?

> SQL is an abbreviation for "Structured Query Language." It is a specialized language for updating and requesting information from databases. SQL is an ANSI and ISO standard, and is the de facto standard database query language. SQL works with a variety of established database products such as those from Oracle and Sybase. It is widely used in both industry and academia, often for enormous, complex databases.

In a distributed database system, a program often referred to as the database's "back end" runs constantly on a server, interpreting data files on the server as a standard relational database. Programs on client computers allow users to manipulate that data, using tables, columns, rows, and fields. To do this, client programs send SQL statements to the server. The server then processes these statements and returns replies to the client program.

To illustrate, consider a simple SQL command, SELECT . SELECT retrieves a set of data from the database according to some criteria. For example, to retrieve from a table called Class all records that have a value of Smith for the column Last_Name, a client program would prepare and send this SQL statement to the server back end:

SELECT * FROM Class WHERE Last_Name='Smith';

The server back end may then reply with data such as this:

E_Mail	Last_Name	First_Name
jsmith@xyz.net	Smith	John
dsmith@xyz.com	Smith	David
msmith@abc.com	Smith	Matthew
3 rows in set (0.05 sec)		

(Indiana University)

Why use SQL?

SQL was chosen to make retrieving the information easier. A single SQL statement can accomplish as much work as pages of code in standard programming language. The following SQL statement, for example, returns the Name, Reg Code and E-mail address for a department distribution list from a database table named csci.

SELECT DISTINCT Name, Reg, Email FROM csci WHERE class in ("FR", "SO") and major in ("CSCI").

This statement extracts freshman and sophomore computer science majors from a table of class rolls. This information is stored in a temporary query for later use.

The SQL query for a class distribution is similar. The following SQL statement sorts a class roll in a table csci4957201 by name:

SELECT * FROM csci4957201 ORDER BY Name

Creating the Distribution List

Once the database is created, the user then can create distribution lists for the e-mail client selected. There are two types of distribution lists that the List Maker program creates: lists for an individual class and lists for a department.

Class Distribution List

Creating a class distribution list is straightforward. The following is an example of an SQL query that would be used:

SELECT NAME, EMAIL, REG FROM Csci1101001 WHERE REG="Enrolled"

The output from this list-building query is then written to a distribution list, in the required format, with a user-specified name.

Departmental Distribution List

Creating a departmental list is a little more difficult, thanks to the variety of criteria that a user might wish to specify when creating such a list. List Maker program allows the user to specify a list based on student classification, college, and the major. After the required query is created, the distribution list is created in the same way as a class list. In an effort to keep the program user friendly and error free, the List Maker program extracts the different colleges, classifications, and majors from the departmental database into preset queries. The SQL for each is as follows:

College = SELECT DISTINCT Coll FROM database table

Classification = SELECT DISTINCT Class FROM database table

Major = SELECT DISTINCT Major FROM database table

After these queries are built, the user can generate a distribution list by selecting colleges,

classifications, and majors from a drop down list box. This drop down box eliminates the need to remember the SIS codes for the different choices. Say, for example, that the user wanted to create a distribution list for all computer science majors, both undergraduate and graduate. The user would select CSCI and CSCM from the Major drop down box. The SQL statement for this query would be:

SELECT DISTINCT Name, Reg, EMail FROM csci WHERE major IN ("CSCI", "CSCM").

The DISTINCT qualifier eliminates duplicates entries from the final list. Name, Reg, and Email are the fields extracted from the database. The database name for this example is csci. The WHERE clause selects only those students that are a CSCI major (undergraduate) or a CSCM major (graduate). Once the requested information is extracted, the creation of the distribution list is the same as creating a distribution list for a class. The user can name the distribution list so that he/she can find the distribution list in the e-mail client.

Detailed Explanation of the Program Requirements

Determining if Microsoft Outlook and the CDO.DLL are Installed

The List Maker program needs to know if Microsoft Outlook is installed on the user's computer. The List Maker program uses Microsoft Outlook MAPI service for programming access to Microsoft Outlook. With Microsoft Outlook, a user needs to establish a MAPI session to use Microsoft Outlook. On normal startup of Microsoft Outlook the MAPI login is automatic. Because the List Maker program accesses Microsoft Outlook programmatically, it needs to know if Microsoft Outlook is installed to start the MAPI session. Appendix one gives details on how to start the MAPI session.

The List Maker program also needs to know if the cdo.dll is installed. The List Maker program uses cdo.dll to gain access to the Personal Address Book (PAB) of Microsoft Outlook. A search for this dll must be done each time the List Maker program starts. This is because the security update that Microsoft has published for Microsoft Outlook 98 and 2000 removes this dll for the user's computer. Microsoft advises that cdo.dll can be used to allow a hacker to use a user's PAB to do harm.

Location of E-mail Clients

The List Maker program needs to know the location of Pegasus Mail and Eudora Lite. The List Maker program uses this information to write the distribution lists directly to the program directory or address book. Because Pegasus Mail distribution lists are in separate files, the List Maker program writes the distribution list to the Pegasus directory for use by the e-mail client. Eudora Lite stores all address and distribution lists in an address book. The List Maker program appends the new distribution lists to the address book of Eudora Lite. Thus, it needs to know the location of the Eudora Lite address book.

Maintain URL of Web Interface

The List Maker program maintains the URL of the web interface so that the user of the program does not have to remember the cryptic URL. The URL for ETSU is https://www2.etsu-tn.edu:443/cgi001/rwaaa015e. This URL would be different for each school that uses the List Maker program. The built-in web browser uses this URL to reach the web interface that the ETSU OIT department has created.

Encrypt/Decrypt E-mail Password

If the user wants to store his/her e-mail password for the List Maker program, it must be encrypted so that unauthorized persons cannot see it. The encryption algorithm is based on an algorithm created by Rodney Thayer. This algorithm can use keys of various sizes. The algorithm can use as many as 2048 bits for the key. The List Maker program currently uses a 40-bit key, in keeping with old guidelines of exporting cryptography. This algorithm is compatible with RSA's RC4. (Shea, 1999)

What is RC4?

RC4 is a stream cipher designed by Ron Rivest for RSA Data Security (now RSA Security). RC4 is a variable key-size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation. Analysis shows that the period of the cipher is overwhelmingly likely to be greater than 10^{100}. Eight to sixteen machine

operations are required per output byte, and the cipher can be expected to run very quickly in software. Independent analysis have scrutinized the algorithm and its is considered secure. (RSA Security)

Retrieving Information via the Internet

List Maker uses an Internet interface designed by the Office of Information Technology at East Tennessee State University to access SIS. List Maker's built-in web browser assumes the use of Microsoft Internet Explorer version 4.01. List Maker, when started, checks for IE v4.01 or later, prompting the user to install the program as required.

List Maker's dependency on IE is a transitive dependency, mediated by List Maker's use of Microsoft Visual Basic, and VB's dependency on IE. The need for IE v4.01 or later was determined by testing: OIT's web interface seemed to fail sporadically when used with older versions of Internet Explorer. A user who does not have version 4.01 installed could use a web browser program outside the List Maker program to access SIS data and then switch back to the List Maker program to finish creating the distribution lists.

The decision to use Microsoft Internet Explorer was based on three considerations. First, IE is readily interoperable with Microsoft Visual Basic. Second, IE is installed on just about every personal computer that runs Microsoft Windows 95, 98, Millennium Edition, Windows NT, and Windows 2000. If the user is using one of the Microsoft Office Suites, Microsoft Internet Explorer is installed when the suite is installed. Finally, Microsoft Internet Explorer can also be obtained free of charge from the Microsoft web site. The Netscape Communicator web browser cannot be integrated with Microsoft Visual Basic without purchasing additional third party software.

Receiving the Information via E-mail

The List Maker program uses the Post Office Protocol 3 (POP3) to retrieve the e-mail generated by the web interface to SIS. The List Maker program simply receives e-mail messages and does not delete any messages from the user's inbox. This was by design to reduce the chance of accidently deleting the wrong e-mail. The drawback is that users must delete SIS generated e-mail by hand.

An Overview of the Post Office Protocol 3 (POP3)

> POP (the standard version is POP3, described by RFC 1725) is a standard protocol for retrieving email from Internet mail servers. Like most Internet protocols, POP is command-line-based: all operations are performed by building text commands and sending them to the server. (Vivtek)

How is the Correct E-mail Message Determined?

The List Maker searches for e-mail messages with the subject OUTPUT FILE FROM WEB CLASS ROLL/EMAIL REQUEST. When such a message is found, the List Maker extracts course data from the message body, saving this data as a text file with the course number for the File_Name (see Figure 2-1). The file is stored in the List Maker program directory. If there are duplicate messages on the e-mail server, the last message with the required course number is the message that List Maker program will use. This should not cause a problem for the user, as long as messages with the same subject are ordered chronologically on the server.

Creating the Database

The List Maker program uses Microsoft Data Access Objects, the Microsoft Jet database engine, and the Microsoft Jet workspace to convert SIS e-mails to database tables. The layout for the table is shown in Table 2-1. After the program creates the database and the table, the program parses each line of the text file to extract the individual elements for each item in the database table.

An Overview of the Three Technologies Used

DAO Overview. DAOs (Data Access Objects) enable the programmer to use a programming language to access and manipulate data in local or remote databases and to manage databases, their objects, and their structure. For more information on the Data Access Objects, refer to http://msdn.microsoft.com/library/default.asp

Microsoft Jet Database Engine. Microsoft's Jet database engine is a database management system that retrieves data from and stores data in user and system databases. The Microsoft Jet database engine can be thought of as a data manager component with which other data access systems, such as Microsoft Access and Microsoft Visual Basic, are built. For more information on the Microsoft Jet database engine, refer to http://msdn.microsoft.com/library/default.asp

Microsoft Jet Workspace. Microsoft's Jet workspace is a workspace that uses the Microsoft Jet database engine to access a data source. The data source can be a Microsoft Jet database file (.mdb), an ODBC database, such as a Paradox database, or an ISAM database. For more information on the Microsoft Jet workspace, refer to http://msdn.microsoft.com/library/default.asp

Converting the SIS File to a Microsoft Access Database

SIS e-mail messages are converted to access databases using a two-step process: e-mail message to comma-delimited file, then comma-delimited file to an Access database. This use of a two step procedure serves two purposes. First, the intermediate step that converts comma-delimited files to access database can also be used to support Outlook and Outlook express; these programs each export their address book as a comma separated file. The use of a comma-delimited file also enhances data integrity by allowing incomplete List Maker operations to be rolled back at need. Rollback, which is not implemented in the current version of List Maker, would allow List Maker to recover from a failed attempt to update an existing database by restoring the original. The List Maker database, once created, can be used to generate grade sheets and other course-related list-based databases.

Creating the Distribution Lists

Querying the Database

Once the program has created the database, the user then can use the database to generate distribution list for a class or department. The user need not have Microsoft Access installed on their computer to use the database to create the distribution lists. The List Maker program setup

process installs the necessary dynamic link libraries and ActiveX components needed to create and access a Microsoft Access database.

Using the Distribution Lists

Once the program creates the distribution lists, the user can use them in their e-mail clients as usual. The distribution lists for Eudora Lite, Outlook 97/98, and Pegasus Mail are stored in the appropriate e-mail client ready to be used. Users of Netscape Communicator and Outlook Express must import the LDIF file created by List Maker into the program before using the distribution list.

Some Design Details of the List Maker Program

Locating Pegasus Mail and Eudora Mail

The List Maker program locates the Pegasus Mail home directory by searching for a file named pmail.ini. The search is accomplished with a Win32 API call that does a recursive directory search on the caller's behalf. If pmail.ini is not found, the List Maker program gives the user the option of entering the location of Pegasus Mail. The logic for locating pmail.ini must also support PC configurations that support two or more users of Pegasus Mail. Because Pegasus Mail allows multiple users and the distribution lists and address books are stored in the users' profiles, a way had to be found to store the generated distribution lists and address books in the correct place. The setup dialog for the List Maker program allows the user to enter their user name for the Pegasus User Name.

A similar strategy is used determine the home directory for of Eudora mail. Here, List Maker searches for eudora.exe, and, in the case of Eudora Lite, assumes that only one user is using an e-mail client.

The List Maker Program E-mail Client

The List Maker program uses an e-mail client based on the POP3 protocol. POP3 was used instead of one of the other e-mail protocols, because no other software had to be installed on the user's computer. A MAPI client could have been used instead of POP3, but a MAPI client

like Microsoft Outlook would have to been installed on the user's computer. If one was assured that a MAPI client was installed, writing a MAPI client is a fairly straightforward process.

Conversion Routines

The address book conversion routines were added to the List Maker program as an enhancement to the basic program. Because the OIT department at ETSU was phasing out support for the Pegasus Mail program and because the users of this program were converting to a different e-mail client, they would have to re-create all of their address books and distribution lists. With the conversion routines, the users would be able to convert their Pegasus address books and distribution list to any of the e-mail clients looked at for this project.

Routines for converting SQL query outputs to mailing list were straightforward to design and code, once address book and distribution list formats were known. See chapter two for the file formats of the e-mail clients considered for this project and for explanation of how the Microsoft MAPI and CDO API's eliminate the need for decoding Outlook's address book format. Further details of the logic for using MAPI to update address books is given in Appendix One.

Limitations of the Conversion Routines

The conversion routines created for this project are not totally automated.

1. Users of Netscape Messenger must export Messenger's address book as an LDIF file for conversion. This is required because data files were not decoded for this project. Also, any address books and distribution lists converted to Netscape Messenger will have to be imported from the LDIF file created by the conversion routines.
2. Users of Microsoft Outlook Express must export their address book as comma separated file to support conversion. This conversion will not keep any distribution lists, and the user will have to recreate the distribution lists in the new e-mail client. Also, the conversion of another e-mail client to Microsoft Outlook Express will have to be imported, because the file format for this program was not

looked at for this project. Microsoft Outlook Express imports the LDIF file format.

3. Users of Microsoft Outlook must export the PAB and the Contacts address books to a comma separated file for conversion. Although the MAPI and CDO functions were used to write distribution list to the PAB, it should also be able to read the PAB and convert the PAB to other e-mail client address book. However, this was not looked at for this project at this time. This could be used as a possible future enhancement to the program.

CHAPTER 4

USABILITY

Overview

This chapter considers List-Maker-related usability issues. In particular, it focuses on time saved by List Maker, and users' perceptions of interface quality. Data for this chapter were gathered through a survey of List Maker users. (Appendix Two)

Determining Skill Levels

The skill levels for List Maker users were determined by asking each user to evaluate his/her abilities as a user subjectively. They were not determined by any objective tests.

Survey Results

E-Mail Clients in Use

Currently, nine ETSU employees are using List Maker. Of those, five use the Microsoft Outlook Express mail client; two use Netscape Communicator; and the remaining two use Pegasus.

Length of Time to Create Distribution Lists by Hand

Six of the nine users responded to the request for data on how long mailing list take to create by hand. Five reported times between 30 minutes and an hour; one reported more than an hour, on average.

Length of Time to Create Distribution List Using List Maker Program

The List Maker program apparently reduced the time to create distribution lists. Of the six users who responded to the request for List Maker timing data, all reported that lists were generated in no more than thirty minutes.

Skill Level of Users

The computer skill level of the user's varied. The skill levels novice, some what skilled,

knowledgeable, and advanced users.

Skill Level Versus Time to Create Distribution Lists

There was little correlation between the skill levels of the users and the amount of time needed to create the distribution lists either by hand or using the List Maker program.

Quality of the List Maker Program

Participants in the survey offered no suggestions to improve the interface of the List Maker program. One user requested support for generating distribution lists based on other SIS data. For example, the user wanted the ability to generate lists of students with thirty hours or fewer or a list of students in a particular major.

CHAPTER 5
CURRENT STATUS

Overview

This chapter examines the current status of the project, including the status of the program and the cooperation between Walters State Community College (WSCC) and ETSU. It also discusses the as yet unsuccesseful efforts to promote the use of List Maker at other schools, including Northeast State Technical Community College (NSTCC).

Status of the E-Mail Distribution List Maker and Conversion Program

Distribution List Maker

Currently the E-Mail Distribution List Maker and Conversion Program is complete. The program is able to correctly request the information by using the Internet Interface designed by OIT. It is able to retrieve the information requested by the user using the List Maker built-in e-mail client. After the information has been retrieved, the program converts the text file to a Microsoft Access database that is then used to extract the information needed to create the distribution list.

The E-mail Distribution List Maker and Conversion program also creates distribution lists for the following e-mail clients: Pegasus Mail, Eudora Lite, Microsoft Outlook, and Netscape Messenger 4.0 and Microsoft Outlook Express. The E-mail Distribution List Maker and Conversion program can write directly to all the e-mail programs, with the exception of Netscape Messenger and Microsoft Outlook Express. These two programs must import the lightweight directory access protocol data interchange format (LDIF) file using the programs import features.

Conversion Routines

The conversion routines also can read from and write to the e-mail programs chosen for this project. However, there are exceptions to this. If the user converts a distribution list or address book to Netscape Messenger or Microsoft Outlook Express, the LDIF file will have to be imported into the appropriate program using its import feature. The other exception involves the

conversion of Netscape Messenger, Microsoft Outlook Express, and Microsoft Outlook. In order to convert the Netscape Messenger address book, the user must first export the address book to the LDIF format and then convert this file to e-mail program that the user is using. Also, if the user wants to convert the Microsoft Outlook Express or Microsoft Outlook address book to another program, the user must first export the address book to a comma separated file for the conversion routines. Once the user has exported the file, the user then can convert the comma separated file to the appropriate e-mail program. If the user were to export the address book of Microsoft Outlook Express or Microsoft Outlook, he/she will lose their distribution lists, because these two programs do not export the distribution lists.

Walters State Community College

On the 5th of November, 1999, Dr. Countermine and I met with representatives of Walter State Community College (WSCC). We explained how this project could help the faculty and staff at WSCC. We also explained that some work would be required on their part to make this project work at WSCC. This work would be creating the web interface with the SIS program to extract the class rolls. However, that would not have to start from scratch; the OIT office at ETSU was willing to give them the code used to create the interface at ETSU.

The atmosphere of the meeting was very negative at the start. By the end of the meeting, the atmosphere changed and the feeling was that they might approve the project. There was great interest from the computer science department at WSCC and, in fact, they volunteered the department for testing if the project is approved. During this meeting we found out that the students at WSCC do not use the campus e-mail. The students use one of the free e-mail services on the Internet. The faculty and staff use these e-mail addresses to communicate with the students. Without the students using the campus e-mail system, this project would not work at WSCC.

Outcome of the Project

This project was turned down by WSCC with no explanation. However, there are several possible explanations why the project was turned down. The first possibly is the lack of understanding on how the project could benefit the faculty and staff of WSCC. Another

possibility is the lack of ownership on the part of WSCC personnel. These are perceived feelings and are not meant to be derogatory in any way to WSCC. There could be other valid reasons why WSCC turned down the project.

Northeast State Technology Community College

In October 27th, 1999, Dr. Countermine and I met with Northeast State Technology Community College (NSTCC). We explained how the project would help the faculty and staff at NSTCC and the work required on behalf of NSTCC. The director of computer services at NSTCC agreed to work on the project and change the code of the program that OIT at ETSU is using to generate the web interface. Some of the technical concerns that came up in this initial meeting and their answers are listed here:

1. Where is the secure web server located? (on the VMS)
2. What is the interface between the WEB and SIS? (via a HTML file located on the VMS)
3. What equipment if any is needed? (None)
4. Is there anyone at NSTCC who can change the ETSU code to work at NSTCC? (Yes, the computer service department has a programmer who can re-write the COBOL program.)

In January of 2000, NSTCC raised additional concerns about authentication of faculty and staff. NSTCC does not store the faculty and staff e-mail the same way that ETSU does. These concerns led to a second meeting between NSTCC computer services and ETSU OIT offices with Dr. Countermine and me attending. During this meeting it was decided to give NSTCC a copy of the authentication routine used at ETSU.

As of March 10th, 2000, NSTCC was having the following problems with the implementation.

1. ETSU runs the program on the same server with their SIS system while NSTCC runs it on a system that does not have a web server.
2. NSTCC does not store the e-mail addresses in the same field as ETSU.
3. The authentication routine used at ETSU uses a file that was developed in house. The file was sent to NSTCC for possible use in the implementation at NSTCC.

At this time, however, NSTCC was hopeful that they could get the program to work on their system, because it would save the staff in the computer services department a great deal of time and the instructors much frustration.

On April 26th, 2000, the computer service department at NSTCC was having a problem with the web server on the VMS machine. It would not run the script files needed for this project. They have installed the Netscape VMS server on one of the machines. They are trying to get everything to work. The web server is on one machine the SIS databases are on another machine, and they are working on a SQL query to get the required information.

On October 23rd, 2000, the following update was received from NSTCC. They have had many delays on the project for a variety of reasons and only get to work on it occasionally. The supervisor of the person who was doing the work for this project had to assume another supervisor's responsibilities as well as his/her own and this led to not having enough time to work on this project. Also, the Netscape server that they were trying to use to replace the OSU server was very unreliable, and they had to find an alternative. They are now using the Apache server and it's working much better, but they did have to master new procedures, particularly for getting it to agree to use VMS command procedures as CGI scripts.

As mentioned before, NSTCC's configuration is different from ETSU's in that their SIS files are on a different machine than the web server. They think the best way for them to implement it is to extract the data from SIS that is actually used in the program using FOCUS, and modify the program to read that file. FOCUS is a query language/ report generation type system. Their system manager has been looking at the program and comparing the record layouts and thinks they can use the same extract with just a few modifications.

Outcome of the Project

After looking at the information received from NSTCC it is very difficult to take a program from one institution and make it work on another institution system unless it has the same platforms and data configurations. It looks like what NSTCC is doing would be comparable with other schools within the TBR schools.

The additional work required for a project such as this has to be weighed against the work required to support the school so that the school can function on a day to day basis. This is

a classic example of how you would like to help everybody but there is not enough time in the day. Also, with the additional responsibilities of two different roles take time from any additional projects. However, given enough time and resources in manpower this project could be made to work at NSTCC. There is hope that this project will be completed and tested at NSTCC before the project comes to a close at ETSU.

CHAPTER 6

FUTURE WORK ON THE PROGRAM

Enhancements to the List Maker Program

One possible enhancement to List Maker would be separating the functions for creating distribution lists and converting address books and distribution lists. A user just might want to use the conversion portion of the program to convert his/her address books and distribution lists to another supported e-mail client.

Another possible enhancement would be the ability to write to and read from the Netscape Messenger and Microsoft Outlook Express address books. Currently the project has to use an intermediate file for the conversions and the creation of the distribution lists. This should make it easier for the user of the program and it should also save some time.

A third possible enhancement would be looking into the possibility of reading the PAB of Microsoft Outlook in order to convert the address book to other e-mail clients.

BIBLIOGRAPHY

BIBLIOGRAPHY

Bockmann. C. J., Klander, L., Tang, L. (1998). Writing a Fax Program in Visual Basic. In Visual Basic Programmer's Library (pg. 773). Las Vegas, NV: Jamsa Press

CDO - Microsoft Platform SDK Messaging and Collaboration (CDO 1.2.1). ("n.d.") MSDN Online Library. [On-line. Retrieved April 27, 2000 from the World Wide Web: http://www.msdn.microsoft.com/library/default.asp

HOWTO: Work with Distribution Lists Using CDO (1.x) from Visual Basic. ("n.d") MSDN Online Knowledge Base. [On-line. Retrieved December 13, 1999 from the World Wide Web: http://support.microsoft.com/support/kb/articles/q178/7/87.asp

Indiana University, ("n.d."), What is SQL?, [On-line], Retrieved August 28, 2000 from the World Wide Web: http://kb.indiana.edu/data/ahux.html

Leonard, W., Hudspeth, L., Lee, T. J. (1998). A Cookbook for Conversion. In R. Petrusha (Ed.), Outlook Annoyances (pg. 186). Sebastopol, CA: O'Reilly & Associates

MAPI - Microsoft Platform SDK Messaging and Collaboration (Messaging API). ('n.d.") MSDN Online Library. [On-line. Retrieved April 27, 2000 from the World Wide Web: http://www.msdn.microsoft.com/library/default.asp

RSA Security, ("n.d.") What is RC4?, [On-line], Retrieved October 26, 2000 from the World Wide Web: http://www.rsasecurity.com/rsalabs/faq/3-6-3.html

Shea, B, Et Al. (1999). Thayer Encryption. In C. Denny, C. Denny, T. Amico, J. Jensen, T. Simpson (Eds.), Visual Basic Source Code Library (pp. 442-444). Indianapolis, IN: Waite Group's

Vivtek, ("n.d.") Topic: POP (Post Office Protocol), [On-line], Retrieved August 28, 2000 from the World Wide Web: http://www.vivtek.com/pop.html

APPENDICES

APPENDIX A: MAPI and CDO

This appendix will explain how a MAPI session is started and how the MAPI session access the Personal Address Book of Microsoft Outlook. It will also give a brief expiation of the purpose for each function in the code segment. The code sample is available from Microsoft Knowledge Base at http://support.microsoft.com/support/kb/articles/q178/7/87.asp.

The first step is to establish a MAPI session. This is accomplished by declaring a public object as a MAPI session (see code segment). After the object is declared, the user must login to create a MAPI session. The function MapiLogon performs the logon and establishes the MAPI session.

Once the MAPI session is established, List Maker verifies that the personal address book (PAB) service is installed on the current computer. This is accomplished with the function AddressBookExists. This function takes the name of the address book. In this case the name of the address book is “Personal Address Book”. If the PAB is installed, MAPI can then be used to add a user or a Distribution List (DL) to the PAB.

The function AddUserToPAB adds a user to the PAB. This sample code takes the AddressList and the name of the new user. It adds the user name and the e-mail address to the PAB database and then performs the update to write the new record.

The function AddDLToPAB adds a distribution list to the PAB. This sample code takes the AddressList and the name of the Distribution List one wants to create. It then creates the entry in the PAB database and performs the update to write the new record.

The function AddUserToDL adds a new entry to the distribution lists. This sample code takes the AddressList and the name of the new entry to be added to the distribution list. Each field in the NewMember record is added as shown in the sample code. After all the fields are populated, an update is performed to save the record.

For this project the code in the sample was changed very little. The biggest change was in the AddUserToDL and AddUserToPAB functions. The declarations were changed so that the information needed for each field can be passed to the function. This allows the programmer to pass the field data to the function instead of hard coding it into the function. For an example of the changed code see the new AddUserToDL function at the end of this appendix.

```
'*********************************************************************
' Distribution List Functions
'---------------------------------------------------------------------
'
' Functions and Procedures in this Module
'*********************************************************************
' 1. AddressBookExists: Determines if a specific AddressList Exists
'                       - for example, PAB, GAL, and etc.
' 2. AddDLToPAB:        Creates a new Distribution List (DISTRIBUTION LIST).
' 3. AddUserToDL:       Used to add a new recipient to a DISTRIBUTION LIST.
' 4. AddUserToPAB:      Add a new user to the MAPI address list.
' 5. DeleteUserFromDL:  Removes a recipient from a DISTRIBUTION LIST. Also
'                       contains
'                       logic to check if a specific recipient is on
'                       a DISTRIBUTION LIST.
' 6. GetDLOwner:        Get the Owner of a Distribution List.
'
'*********************************************************************
'REMEMBER: You still need the pertinent rights to be able to
'manipulate MAPI AddressList objects.
'
'*********************************************************************
Public objSession As MAPI.Session
'*********************************************************************
' A conglomerate function that demonstrates use of several of the
' other functions in this module. Look through these functions for
' the string "TO DO:". The functions will work as is, but will not
' yield meaningful content until they are customized as noted on
' these lines.
'*********************************************************************
Public Sub Main()
   Dim oAddressList As AddressList
   Dim oAddressEntry As AddressEntry
   Dim oNewUserAddressEntry As AddressEntry
   If MapiLogon() = True Then
      Set oAddressList = AddressBookExists("Personal Address Book")
      If Not oAddressList Is Nothing Then
        'TO DO: You will want to customize the string values
        'in the next two lines.
         Set oNewUserAddressEntry = _
           AddUserToPAB(oAddressList, "Custom New User")
         Set oAddressEntry = AddDLToPAB(oAddressList, "My Custom DL")

         If Not oAddressEntry Is Nothing Then
           'TO DO: Recipient to be added on next line should be
           'customized.
           If AddUserToDL(oAddressEntry, "MyNewDLMember") = True Then
              MsgBox "Added new user and DL, and new member in the DL"
           Else
              MsgBox "Adding a user to the DL didn't work"
           End If
         Else
            MsgBox "Custom DL wasn't added to the Personal Address Book"
         End If
         Exit Sub
```

```
      Else
          MsgBox "There is no PAB Service installed on this machine"
          Exit Sub
      End If
   End If
End Sub

'**********************************************************************
' FUNCTION     - AddressBookExists
'----------------------------------------------------------------------
' PARAMETERS   - sAddressBookName as string and can be one of two
'                values:
'               "Personal Address Book" or "Global AddressList".
' DESCRIPTION  - Determines if the Personal Address Book Service is
'                installed on the current computer.
' RETURN VALUE - An AddressList object pointing to the addresslist or
'                nothing if it does not exist.
'**********************************************************************
Public Function AddressBookExists(sAddressBookName As String) _
   As AddressList
   Dim oAddressList As AddressList

   For Each oAddressList In objSession.AddressLists
      If oAddressList.Name = sAddressBookName Then
         Set AddressBookExists = oAddressList
         Exit Function
      End If
   Next oAddressList
   Set AddressBookExists = Nothing
End Function

'**********************************************************************
' FUNCTION     - AddDLToPAB
'----------------------------------------------------------------------
' PARAMETERS   - oAddressList as AddressList, sDLName As String (The
'                name of the new Distribution List to create.)
' DESCRIPTION  - Add a new Distribution List to the MAPI address list
'                specified in first parameter.
' RETURN VALUE - An AddressEntry object pointing to the address entry
'                or nothing if it does not exist.
'**********************************************************************
Public Function AddDLToPAB(oAddressList As AddressList, _
   sDLName As String) As AddressEntry
   Dim oAddressEntries As AddressEntries
   Dim oNewAddressEntry As AddressEntry

   On Error GoTo Trp_AddDLToPAB:
   Set oAddressEntries = oAddressList.AddressEntries
   Set oNewAddressEntry = oAddressEntries.Add("MAPIPDL", sDLName)
   oNewAddressEntry.Update
   Set AddDLToPAB = oNewAddressEntry
   Exit Function
Trp_AddDLToPAB:
   Set AddDLToPAB = Nothing
   Exit Function
```

```
End Function

'*********************************************************************
' FUNCTION     - AddUserToDL
'---------------------------------------------------------------------
' PARAMETERS   - oDL As AddressEntry (This is the object reference to
'                the Distribution List the user is being added to.),
'                sUsername As String (The name of the new member to
'                be added to the Distribution List.)
' DESCRIPTION  - Adds a new recipient to a Distribution List.
' RETURN VALUE - A Boolean value specifying whether or not adding the
'                user was successful.
'*********************************************************************
Public Function AddUserToDL(oDL As AddressEntry, sUserName As String) _
   As Boolean
   Dim oNewMember As AddressEntry

   On Error GoTo Trp_AddUserToDL:
   'TO DO: Customize for your new DL member.
   Set oNewMember = oDL.Members.Add("SMTP", "MyAddressName")

   'You need to fill in all your own data here. The data below is
    'for sample purposes. This is not an inclusive list of potential
    'fields, and of the fields represented here, not all must be
    'populated for a viable address entry.
    With oNewMember
      'TO DO: Obviously all of these should be modified.
      .Address = "MySMTPAddress@company.com"
      'Generic Comments Field.
      .Fields(ActMsgPR_COMMENT) = "My notes here."
      'Business Generic Fields.
      .Fields(ActMsgPR_COMPANY_NAME) = "Microsoft Corporation"
      .Fields(ActMsgPR_DEPARTMENT_NAME) = "Microsoft Technical Support"
      .Fields(ActMsgPR_MANAGER_NAME) = "John Doe"
      .Fields(ActMsgPR_ASSISTANT) = "Jane Doe"
      .Fields(ActMsgPR_GIVEN_NAME) = "Bill"
      .Fields(ActMsgPR_MIDDLE_NAME) = "A."
      .Fields(ActMsgPR_SURNAME) = "Smith"
      .Fields(ActMsgPR_TITLE) = "Mr."
      .Fields(ActMsgPR_EMAIL_ADDRESS) = "mymail@Microsoft.com"
      'Business Address Fields.
      .Fields(ActMsgPR_OFFICE_LOCATION) = "Building 13"
      .Fields(ActMsgPR_BUSINESS_ADDRESS_STREET) = "1 Microsoft Way"
      .Fields(ActMsgPR_BUSINESS_ADDRESS_CITY) = "Redmond"
      .Fields(ActMsgPR_BUSINESS_ADDRESS_STATE_OR_PROVINCE) = "WA"
      .Fields(ActMsgPR_BUSINESS_ADDRESS_POSTAL_CODE) = "98052"
      .Fields(ActMsgPR_BUSINESS_ADDRESS_COUNTRY) = "USA"
      'Telephone Number Fields.
      .Fields(ActMsgPR_BUSINESS_FAX_NUMBER) = "425-555-0329"
      .Fields(ActMsgPR_BUSINESS_TELEPHONE_NUMBER) = "425-555-8080"
      .Fields(ActMsgPR_BUSINESS2_TELEPHONE_NUMBER) = "425-555-8081"
      .Fields(ActMsgPR_CALLBACK_TELEPHONE_NUMBER) = "425-555-8082"
      .Fields(ActMsgPR_CAR_TELEPHONE_NUMBER) = "206-555-0000"
      .Fields(ActMsgPR_ASSISTANT_TELEPHONE_NUMBER) = "425-555-8083"
      .Fields(ActMsgPR_COMPANY_MAIN_PHONE_NUMBER) = "425-882-8080"
```

```
      .Fields(ActMsgPR_MOBILE_TELEPHONE_NUMBER) = "425-555-8084"
      .Fields(ActMsgPR_PAGER_TELEPHONE_NUMBER) = "N/A"
      .Fields(ActMsgPR_PRIMARY_FAX_NUMBER) = "425-555-8085"
      'Home detail fields.
      .Fields(ActMsgPR_HOME_ADDRESS_STREET) = "1234 My Street"
      .Fields(ActMsgPR_HOME_ADDRESS_CITY) = "MyTown"
      .Fields(ActMsgPR_HOME_ADDRESS_STATE_OR_PROVINCE) = "WA"
      .Fields(ActMsgPR_HOME_ADDRESS_COUNTRY) = "USA"
      .Fields(ActMsgPR_HOME_FAX_NUMBER) = "425-555-0001"
      .Fields(ActMsgPR_HOME_TELEPHONE_NUMBER) = "425-555-0002"
      .Fields(ActMsgPR_HOME2_TELEPHONE_NUMBER) = "N/A"

      .Update
     End With

     AddUserToDL = True
     Exit Function
Trp_AddUserToDL:
     AddUserToDL = False
     Exit Function
End Function

'**********************************************************************
' FUNCTION     - AddUserToPAB
'----------------------------------------------------------------------
' PARAMETERS   - oAddressList as AddressList, sDLName As String (The
'                name of the new Distribution List to create.)
' DESCRIPTION  - Add a new user to the MAPI address list specified in
'                first parameter.
' RETURN VALUE - A As AddressEntry object pointing to the address entry
'                or nothing if it does not exist.
'**********************************************************************
Public Function AddUserToPAB(oAddressList As AddressList, sUserName _
    As String) As AddressEntry
    Dim oAddressEntries As AddressEntries
    Dim oNewAddressEntry As AddressEntry

    On Error GoTo Trp_AddDLToPAB:

    Set oAddressEntries = oAddressList.AddressEntries
    Set oNewAddressEntry = oAddressEntries.Add("SMTP", sUserName)
    'TO DO: Change address in next line.
    oNewAddressEntry.Address = "Me@me.com"
    oNewAddressEntry.Update
    Set AddUserToPAB = oNewAddressEntry
    Exit Function
Trp_AddDLToPAB:
    Set AddUserToPAB = Nothing
    Exit Function
End Function

'**********************************************************************
' PROCEDURE    - DeleteUserFromDL
'----------------------------------------------------------------------
' PARAMETERS   - None
```

```
' DESCRIPTION  - Remove a member from a Distribution List, which the
'                user selects from the Phonebook.
'*********************************************************************
Public Sub DeleteUserFromDL()
   'Procedure Level Variables.
   Dim oAddressEntries As AddressEntries
   Dim oAddressList As AddressList
   Dim oRecipients As Recipients
   Dim oRecipient As Recipient
   Dim oAddressEntry As AddressEntry
   Dim oMember As AddressEntry

   'Get the DL from the AddressBook.
   Set oRecipients = objSession.AddressBook(Title:="Select Attendees")
   If oRecipients.Count = 0 Or oRecipients.Count > 1 Then
      MsgBox "Please Select just one member of the Distribution List"
      Exit Sub
   End If
   Set oRecipient = oRecipients.Item(1)
   Set oAddressEntry = oRecipient.AddressEntry
   Set oAddressEntries = oAddressEntry.Members
   'Cycle through the DL until you find the user you seek.
   For Each oMember In oAddressEntry.Members
      If oMember.IsSameAs(oRecipient) Then
      'Or you could say If oMember.Name = "TheRecipsDisplayName"
         oMember.Delete
         Set oMember = Nothing
      End If
   Next oMember
End Sub

'*********************************************************************
' FUNCTION     - GetDLOwner
'---------------------------------------------------------------------
' PARAMETERS   - oSubordinate As AddressEntry (The object reference to
'                the Owner of the DL.)
' DESCRIPTION  - Gets an address entry's Owner. This is relevant only
'                to the Global Address List since you are always the
'                owner of Personal Address Book (PAB) entries.
' RETURN VALUE - A String value of the owners name.
'*********************************************************************

Public Function GetDLOwner(oSubordinate As AddressEntry) As String
   ' IMPORTANT: This code assumes that the Owner of the
   ' Distribution list is stored in PR_EMS_AB_EXTENSION_ATTRIBUTE_3.
   ' This is not always where the Owner is stored, if it is stored
   ' at all. You should verify where the Owner is stored on your
   ' system and change the Field value that is being retrieved by
   ' this code.
   GetDLOwner = oSubordinate.Fields(&H802F001E).Value
End Function

   '*********************************************************************
' FUNCTION     - MapiLogon
'---------------------------------------------------------------------
```

```
' PARAMETERS   - None.
' DESCRIPTION  - Creates and Logs onto a MAPI Session according to the
'                parameters passed when calling the Logon method. The
'                sample below passes no parameters and uses all default
'                values.
' RETURN VALUE - Boolean indicating success.
'*********************************************************************
Public Function MapiLogon() As Boolean
   'Create a session and log on -- username and password in profile
   Set objSession = CreateObject("MAPI.Session")
   'Change the parameters to valid values for your configuration.
   objSession.Logon
   MapiLogon = True
End Function
```

Changed code for the AddUserToDL function.

```
'*********************************************************************
' FUNCTION     - AddUserToDL
'---------------------------------------------------------------------
' PARAMETERS   - oDL As AddressEntry (This is the object reference to
'                the Distribution List the user is being added to.),
'                sUsername As String (The name of the new member to
'                be added to the Distribution List.) sSMTPAddress As
'                String (The e-mail address of the user.) All other
'                fields are optional.
' DESCRIPTION  - Adds a new recipient to a Distribution List.
' RETURN VALUE - A Boolean value specifying whether or not adding the
'                user was successful.
'*********************************************************************
Public Function AddUserToDL(oDL As AddressEntry, sUserName As String,
sSMTPAddress As String, Optional sNote As String, Optional sCompanyName As
String, Optional sDeptName As String, Optional sManager As String, Optional
sAssistMang As String, Optional sFirstName As String, Optional sMiddleName As
String, Optional sLastName As String, Optional sTitle As String, Optional
sEmail As String, Optional sBusOfficeLoc As String, Optional sBusStr As
String, Optional sBusCity As String, Optional sBusState As String, Optional
sBusZip As String, Optional sBusCountry As String, Optional sBusFax As String,
Optional sBusPhone1 As String, Optional sBusPhone2 As String, Optional
sBusCallBack As String, Optional sBusCarPhone As String, Optional
sBusAssitPhone As String, Optional sBusCompMainPhone As String, Optional
sBusMobilePhone As String, Optional sBusPager As String, Optional sBusPriFax
As String, Optional sHomeStreet As String, Optional sHomeCity As String,
Optional sHomeState As String, Optional sHomeCountry As String, Optional
sHomeFax As String, Optional sHomePhone1 As String, Optional sHomePhone2 As
String) As Boolean


   Dim oNewMember As AddressEntry
   On Error GoTo Trp_AddUserToDL:
   Set oNewMember = oDL.Members.Add("SMTP", "MyAddressName")

   'You need to fill in all your own data here. The data below is
   'for sample purposes. This is not an inclusive list of potential
```

```
   'fields, and of the fields represented here, not all must be
   'populated for a viable address entry.
    With oNewMember
      .Address = sSMTPAddress
      'Generic Comments Field.
      .Fields(ActMsgPR_COMMENT) = sNote
      'Business Generic Fields.
      .Fields(ActMsgPR_COMPANY_NAME) = sCompanyName
      .Fields(ActMsgPR_DEPARTMENT_NAME) = sDeptName
      .Fields(ActMsgPR_MANAGER_NAME) = sManager
      .Fields(ActMsgPR_ASSISTANT) = sAssistMang
      .Fields(ActMsgPR_GIVEN_NAME) = sFirstName
      .Fields(ActMsgPR_MIDDLE_NAME) = sMiddleName
      .Fields(ActMsgPR_SURNAME) = sLastName
      .Fields(ActMsgPR_TITLE) = sTitle
      .Fields(ActMsgPR_EMAIL_ADDRESS) = sEmail
      'Business Address Fields.
      .Fields(ActMsgPR_OFFICE_LOCATION) = sBusOfficeLoc
      .Fields(ActMsgPR_BUSINESS_ADDRESS_STREET) = sBusStr
      .Fields(ActMsgPR_BUSINESS_ADDRESS_CITY) = sBusCity
      .Fields(ActMsgPR_BUSINESS_ADDRESS_STATE_OR_PROVINCE) = sBusState
      .Fields(ActMsgPR_BUSINESS_ADDRESS_POSTAL_CODE) = sBusZip
      .Fields(ActMsgPR_BUSINESS_ADDRESS_COUNTRY) = sBusCountry
      'Telephone Number Fields.
      .Fields(ActMsgPR_BUSINESS_FAX_NUMBER) = sBusFax
      .Fields(ActMsgPR_BUSINESS_TELEPHONE_NUMBER) = sBusPhone1
      .Fields(ActMsgPR_BUSINESS2_TELEPHONE_NUMBER) = sBusPhone2
      .Fields(ActMsgPR_CALLBACK_TELEPHONE_NUMBER) = sBusCallBack
      .Fields(ActMsgPR_CAR_TELEPHONE_NUMBER) = sBusCarPhone
      .Fields(ActMsgPR_ASSISTANT_TELEPHONE_NUMBER) = sBusAssitPhone
      .Fields(ActMsgPR_COMPANY_MAIN_PHONE_NUMBER) = sBusCompMainPhone
      .Fields(ActMsgPR_MOBILE_TELEPHONE_NUMBER) = sBusMobilePhone
      .Fields(ActMsgPR_PAGER_TELEPHONE_NUMBER) = sBusPager
      .Fields(ActMsgPR_PRIMARY_FAX_NUMBER) = sBusPriFax
      'Home detail fields.
      .Fields(ActMsgPR_HOME_ADDRESS_STREET) = sHomeStreet
      .Fields(ActMsgPR_HOME_ADDRESS_CITY) = sHomeCity
      .Fields(ActMsgPR_HOME_ADDRESS_STATE_OR_PROVINCE) = sHomeState
      .Fields(ActMsgPR_HOME_ADDRESS_COUNTRY) = sHomeCountry
      .Fields(ActMsgPR_HOME_FAX_NUMBER) = sHomeFax
      .Fields(ActMsgPR_HOME_TELEPHONE_NUMBER) = sHomePhone1
      .Fields(ActMsgPR_HOME2_TELEPHONE_NUMBER) = sHomePhone2

      .Update

     End With

     AddUserToDL = True
     Exit Function
Trp_AddUserToDL:
     AddUserToDL = False
     Exit Function
End Function
```

APPENDIX B: User Survey

1. Are you using e-mail to communicate with your students/staff? ☐ Yes ☐ No

2. If No do you plan on using e-mail to communicate with your students/staff?

☐ Yes ☐ No

3. If you are using e-mail to communicate with your students/staff what e-mail program are you using?

☐ Pegasus Mail ☐ Netscape Communicator ☐ Outlook 98

☐ Eudora Lite ☐ Eudora Pro ☐ Other _____

☐ Outlook Express ☐ Outlook 97

4. Are you using distribution list with your e-mail program? ☐ Yes ☐ No

5. If Yes how are you creating the distribution list?

☐ By Hand ☐ Using a third party package

6. How are you getting the students e-mail address?

☐ From the Student ☐ From SIS ☐ Other _____

7. How long does it take you to create a distribution list for your class?

☐ 10 Minutes ☐ 20 Minutes ☐ 30 Minutes ☐ 40 Minutes

☐ 50 Minutes ☐ 1 hour ☐ Other _____

8. How long does it take to create a distribution list using the E-Mail Distribution List Maker Program?

☐ 10 Minutes ☐ 20 Minutes ☐ 30 Minutes ☐ 40 Minutes

☐ 50 Minutes ☐ 1 hour ☐ Other _____

9. How would you rate yourself on computer skills?

☐ Novice ☐ Some what skilled ☐ Knowledgeable ☐ Advanced User

10. Is the interface understandable? ☐ Yes ☐ No

11. Have you converted your e-mail address book to another e-mail program?

☐ Yes ☐ No

12. If yes how did you convert the address book?

☐ By Hand ☐ Using a third party package

13. If done by hand how long did it take?

☐ 10 Minutes ☐ 20 Minutes ☐ 30 Minutes ☐ 40 Minutes

☐ 50 Minutes ☐ 1 hour ☐ Other _____

14. If done by a third party package how long did it take?

☐ 10 Minutes ☐ 20 Minutes ☐ 30 Minutes ☐ 40 Minutes

☐ 50 Minutes ☐ 1 hour ☐ Other _____

15. How would you rate this program?

☐ Excellent ☐ Very Good ☐Good ☐ Poor

☐ Very Poor ☐ Other _____

VITA

ALLAN R. ANDERSON

Personal Data:	Date of Birth: March, 13 1954
	Place of Birth: Oak Ridge, Tennessee
	Martial Status: Married
Education:	Public Schools, Roane County, Tennessee
	East Tennessee State University, Johnson City, Tennessee; Computer Science Information Science, BS, 1998
	East Tennessee State University, Johnson City, Tennessee; Computer Science Information Science, MS, 2001
Professional Experience:	Technical Sergeant United States Air Force (Ret.) 1974 - 1994
	Graduate Assistant, East Tennessee State University, College of Applied Science and Technology, 1998 - 2000
Honors and Awards:	Unsung Hero Award
	President of Once Again Students in School (OASIS)