Developing Mail Enabled Applications with HP OpenMail

Tony Jones
Hewlett-Packard Company
20 New England Avenue
Piscataway, NJ 08854-4107
(908) 562-6248

HP OpenMail provides enterprise wide, client/server based electronic messaging. It also has the capability of allowing applications to use it as an information transport. This paper describes how to develop mail enabled applications that access OpenMail servers. Mail enabled applications can be client based or server based. For example, a client based application can extract information from a daily report, summarize it, and then mail it to a selected group of end-users. A server based application can provide a means for any end-user to request specific information.

The topics discussed in this paper include:

- Introduction to HP OpenMail architecture
- Introduction to using mail API calls (MAPI, UAL) within client applications
- Introduction to the OpenMail Request Server
- Techniques used to build mail enabled applications.

Upon completion of the session, the participants will have a better understanding of how to:

1] Use OpenMail as a foundation element in their current system architecture
2] Develop applications that interact with end-users via OpenMail
3] Use either client or server based mail enabled applications to streamline information flow

Introduction to HP OpenMail architecture

OpenMail is a software application that provides electronic mail and other messaging services. It is compliant with the X.400 Recommendations and provides facilities based on international standards to exchange and manage information.

OpenMail consists of a user agent Application Programming Interfaces (API) for communication with the client applications, a message store that holds messages, messaging-optimized Directories and connection to X.500 Directory systems, interfaces that connect to transport systems, and gateways that link to other systems.

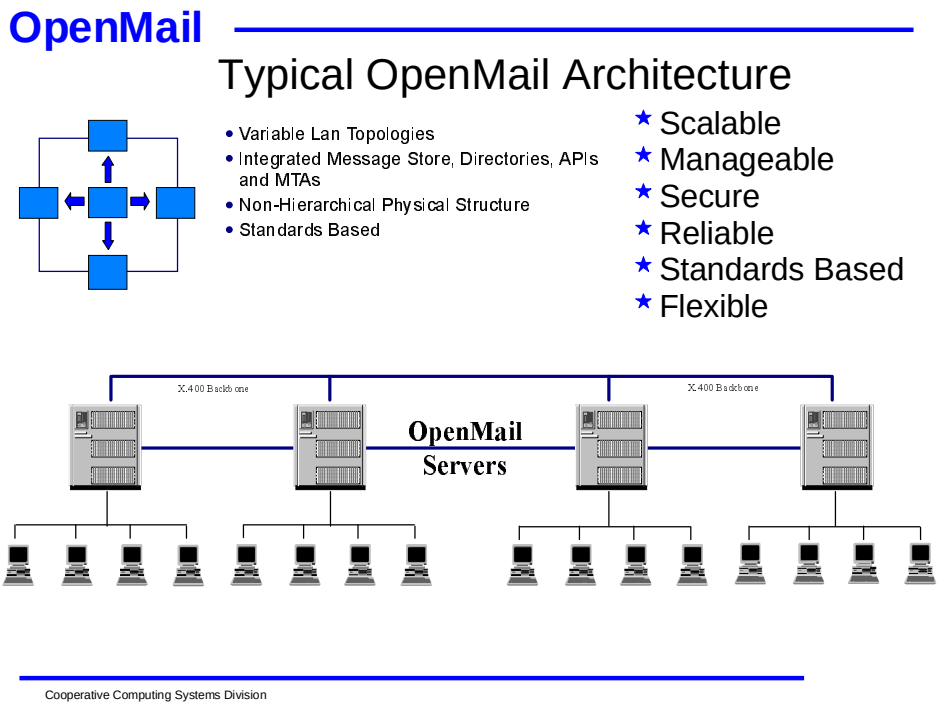

Figure 1. HP OpenMail

OpenMail uses a client-server architecture to provide electronic mail facilities. The OpenMail architecture consists of three main parts:

1] Interfaces to the clients, the X.400 Message Transfer agent, and to Sendmail
2] Gateways to other messaging systems
3] Remaining services, such as local delivery and message routing.

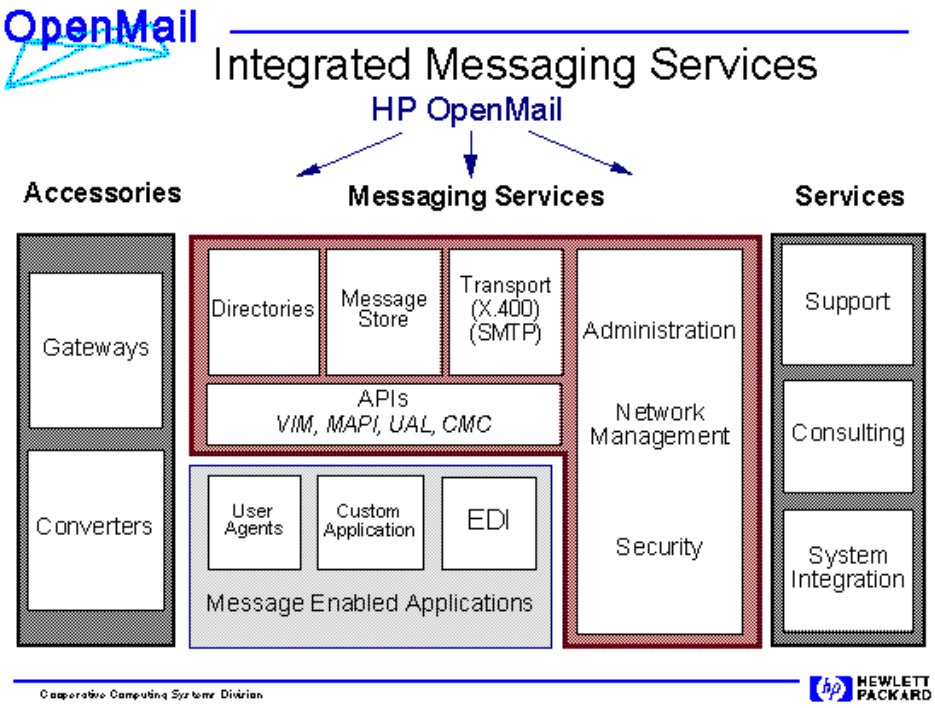

Figure 2. HP OpenMail Messaging Services

Every user in OpenMail is addressed in the following manner:

GivenName SurName/OrgUnit1,OrgUnit2,OrgUnit3,OrgUnit4

The user's name is actually made up of up to four items: GivenName (first name), MiddleInitial, SurName (last name) and generation (Jr, III, etc.). The four organizational units make up the mailnode. A user can have from one to four organizational units.

Example:

Tony Jones/hp,piscataway

Users who are on external mail systems such as the Internet can be reached via the Unix gateway. To address users on external systems, use the Foreign Address field area which consists of the "name@domain.organization" convention surrounded by parentheses after the mailnode.

Example:

Jim Smith/hp,unix(jsmith@compusa.com)

Users on external mail systems such as the Internet can also reach an OpenMail user via the Unix gateway. To address OpenMail users from external systems (using default configuration), use the "name/mailnode@domain.organization" convention. The name field is made up of SurName_GivenName (separated by an underscore), and the mailnode field is made up of the org units (separated by an underscore).

Example:

Jones_Tony/hp_piscataway@hosta.hp.com

In addition, OpenMail users can have more traditional looking electronic mail addresses when messages are sent through the Unix gateway. OpenMail can be configured such that the contents of each user's "INTERNET-ADDR" field will be used instead of the default method OpenMail uses to produce an e-mail address. Therefore, an OpenMail address to the Internet could appear as:

Tony_Jones@om.hp.com

OpenMail supports a command line interface to access various functions including:

Command	Description
omsend	Send an electronic message
omread	Read an electronic message
omnew	List new messages
omdelete	Delete a message
omsearch	Search a directory for user(s)

Programs can use the command line interface to electronically send program output. For example, to send a report file "report.out" to a user called Tony Jones:

```
omsend -t "Tony Jones" -s "FYI report" -a "report.out" -u $USER -p $PASS  -q
```

The -s option is the subject, the -u and -p options are for the OpenMail user and password. The -q option is for quiet mode (no informational messages are generated).

Introduction to using mail API calls (MAPI, UAL)

Client application communication to OpenMail servers is done via application programming interfaces (API). The APIs provide access to the various OpenMail server features including the message store, directory, and system management. The original API for OpenMail was the User Agent Layer (UAL). This API exposes all of the features of the OpenMail server.

A popular set of APIs are the MAPI 0 and MAPI 1 from Microsoft. MAPI 1 (Extended MAPI) is the Microsoft Exchange API. It is used by Microsoft on the Windows family of operating systems to implement Microsoft Exchange, their client/server e-mail system. MAPI 1 is also called X-MAPI.

OpenMail supports the MAPI 1 interface to the OpenMail server. MAPI 1 sits alongside cc:Mail Desktop API, cc:Mail Mobile API, CMC, MAPI 0 (the Microsoft Mail 3.x API), VIM, and P7 as client APIs currently supported by OpenMail. (This range is in addition to OpenMail's native UAL API, which is in effect a superset of the other APIs.) This set of different client APIs enables OpenMail to support a wide range of client, mail-enabled, and mail-aware applications. Supporting MAPI also enables OpenMail to work with MAPI 1 clients and provide the same sort of client connections as the Microsoft Exchange server.

The MAPI 0 functions are:

Command	Description
MAPILogon	Begins a session with the messaging system
MAPIFindNext	Returns ID of the next (or first) mail message of a specified type
MAPIReadMail	Reads a mail message
MAPISaveMail	Saves a mail message
MAPIDeleteMail	Deletes a mail message
MAPISendmail	Sends a mail message, allowing greater flexibility than MAPISendDocuments in message generation
MAPIAddress	Addresses a mail message
MAPIResolveName	Displays a dialog box to resolve an ambiguous recipient name
MAPIDetails	Displays a recipient details dialog box
MAPILogoff	Ends a session with the messaging system

An application developer can focus on using MAPI to access a messaging backbone. It should not matter to the application developer that the MAPI services are provided by MS-Mail for OpenMail instead of the regular MS-Mail. The code should simply bind to the MAPI Dynamic link library (DLL) and call the regular MAPI functions. The fact that MS-Mail for OpenMail is installed will result in the OpenMail version being called. All MAPI-0 calls will be invoked within the context of the OpenMail driver. The added

benefit to this approach is other programs that take advantage of MS-Mail via MAPI now take advantage of OpenMail.

Refer to Appendix A for a sample program that uses MAPI function calls.

Introduction to the OpenMail Request Server

The OpenMail Request server allows a user to mail a request to execute a script and get the results in an electronic mail message. These scripts can provide functions such as return the amount of free disc space on the OpenMail server. Access control lists (ACLs) are also available to control who can access the scripts. To access the request server, replace the GivenName with the request (name of the script), and use "+req" as the SurName:

Request +req/ou1, ou2,ou3,ou4(Foreign Address Field)

Note: The scripts are stored in the "/usr/openmail/req" directory on the OpenMail server.

The Request server provides five items to a script:

- Four environment variables:

Variable	Description
LANG	Language used by the sender (C, english, etc.)
OMSUBJECT	Subject field of the message
OMSENDER	Sender's name (Given & SurName) and mailnode
OMRECIPFA	Foreign address field of recipient's information

- Last body part of the message

For example, suppose you want to access the Request server to find out the amount of disc space consumed by a particular user. To execute a script called *discusage* use the following format:

From: Tony Jones/hp,piscataway
To: discusage +req/hp,piscataway(sbutler)
Subject: Hi there from the gang
Text:

This is line 1
This is line 2

The items seen by the script would include:

LANG=C
OMSUBJECT=Hi there from the gang
OMSENDER=Tony Jones/hp, piscataway
OMRECIPFA=sbutler

The Foreign address field typically looks like: name@host.domain.organization. In this example, we provide a user's name (sbutler).

Reading from "standard in", the script would get "This is line1, This is line 2,..." (on separate lines).

The output of the script (via "standard out") is returned to the sender in an e-mail message.

If a copy of the script resides on separate OpenMail servers, the sender could specify a mailnode on a different server. The script would then execute on the computer that services that mailnode and the results would be returned to the sender.

Techniques used to build mail enabled applications

An application can extract information from a daily report, summarize it, and then mail it to a selected group of end-users. The application can invoke a separate program to mail the information. The distribution list of end-users interested in the information can be stored in the OpenMail server. The application can send messages to the distribution list and let the list be managed at a higher level. This allows for further modularization.

Refer to the Visual Basic program in appendix A to see a sample of how to mail enable applications. This program can be called by other programs. This program can be changed to have a hard-coded file name attachment, and a hard-coded distribution list.

References

Technical Reference: Microsoft Mail Electronic Mail for PC Networks Version 3.2
Programmer's Guide: Microsoft Visual Basic Version 3.0
Microsoft KnowledgeBase Article Q140447

Appendix A.Visual Basic Program that uses MAPI-0 to send mail

The information in this article applies to:
- Microsoft Mail for PC Networks, versions 3.0, 3.2, and 3.5

SUMMARY

The Simple Messaging Application Programming Interface (MAPI) includes functions that allow developers to logon, send, and logoff of Microsoft Mail programmatically. The following Microsoft Visual Basic code illustrates successful calls to these functions.

MORE INFORMATION

The following Microsoft Visual Basic code uses:

- MAPILogOn function to initiate a mail session or run with an existing session if one exists.
- MAPIRecip structure to set up the recipients of the mail message.
- MAPIMessage structure to set up the content of the mail message.
- MAPISendMail function to send the message.
- MAPILogOff function to end the mail session.

IMPORTANT NOTE: Make sure a MAPI declaration module, such as MAPILIB.BAS, is included in the project.

```
'Example VB code of sending mail via Simple MAPI using MAPI.DLL:
'********** Important ******************************
'make sure MAPILIB.BAS or a module that contains
'MAPI functions declaration is already included in the project
' ************************************************
'       Simple MAPI Declarations
' ************************************************

' Set up the message structure and recipient structures
Dim M As MAPIMessage            ' dimension new message structure
Dim Mo As MapiRecip             ' dimension originator structure

M.RecipCount = 1&               ' set RecipCount property of new message to be 1
M.FileCount = 0&                ' set FileCount property of new message to be 0
MsgId$ = ""                     ' set MsgID string to ""
MsgType$ = "IPM.Microsoft Mail.Note"   'set MsgType string to default MS Mail type

M.Reserved = 0&                 ' set Reserved property of new message to be 0&
M.MessageType = MsgType$        ' set MessageType property of new message to  be MsgType$
M.DateReceived = ""             ' set DateReceived property of new message
M.Flags = 0&                    ' set Flags property of new message to be 0&

ReDim mr(0 To 0) As MapiRecip           ' dimension recipient array structure for 1 recipient only
ReDim MF(0 To 0) As MapiFile    ' dimension file attachment array structure
MF(0).Reserved = 0&             ' set Reserved property of file structure to  be 0&
MF(0).Flags = 0&                ' set Flags property of file structure to be  0&
MF(0).Position = -1             ' set Position property of file structure to be -1
```

```
MF(0).FileType = ""                    ' set FileType property of file structure

'You may not need this if a session already is established 'i.e. Mail or S+ is running.
' ***************************************************
' Login and start the MAPI Session
' ***************************************************
rc& = MAPILogon(Form1.hWnd, "", "", MAPI_LOGON_UI, 0&, lhSession&)
If rc& <> SUCCESS_SUCCESS Then
    MsgBox "Error logging in"
    End
End If

' you can replace InputBoxes with strings of text to eliminate the need of  user-interface
M.Subject = InputBox("Enter a subject line:")
M.NoteText = InputBox("Enter some body text:")
aPathName$ = InputBox("Enter a file to attach:")
If Trim(aPathName$) <> "" Then
   MF(0).PathName = aPathName$
   MF(0).FileName = InputBox("Enter the file name to include as:")
   M.FileCount = 1
Else
   MF(0).PathName = ""
   MF(0).FileName = ""
   M.FileCount = 0
End If

'You can replace the InputBox with a full name
who = InputBox("Enter a recipient's alias: ")
If Not IsEmpty(who) Then
   mr(0).Name = who
   mr(0).RecipClass = MAPI_TO
   X = MAPIResolveName(lhSession&, 0, mr(0).Name, MAPI_DIALOG, 0, mr(0))
   X = SUCCESS_SUCCESS
   If X <> SUCCESS_SUCCESS Then
      MsgBox ("The address for this message is not valid.")
      Screen.MousePointer = 0
      rc& = MAPILogoff(lhSession&, 0&, 0&, 0&)
      If rc& <> SUCCESS_SUCCESS Then MsgBox "Error logging off """
   Else
      rc& = MAPISendmail(lhSession&, Form1.hWnd, M, mr(0), MF(0),
      MAPI_DIALOG, 0&)
      If rc& <> SUCCESS_SUCCESS Then MsgBox "Error sending message"

      rc& = MAPILogoff(lhSession&, 0&, 0&, 0&)
      If rc& <> SUCCESS_SUCCESS Then MsgBox "Error logging off """
   End If
Else
      rc& = MAPILogoff(lhSession&, 0&, 0&, 0&)
      If rc& <> SUCCESS_SUCCESS Then MsgBox "Error logging off """
End If
```

End