OTN OpenMail Technical Note

OpenMail Outlook (MAPI) Support Draft A

Summary

**********PLEASE NOTE THIS IS A DRAFT**********

This OTN describes how OpenMail supports Microsoft Outlook clients using MAPI. The information relates to versions B.05.10 and B.06.00 of OpenMail and versions B.05.20 and B.05.30 of the OpenMail MAPI Service Providers.

The OTN concentrates on how the OpenMail server interacts with the OpenMail MAPI Service Providers and does not attempt to describe the MAPI standard, all the features offered by the Service Providers or the Outlook user interface. You should therefore read this OTN in conjunction with the other documentation listed in the *References* section.

Readership

I have assumed that readers are OpenMail engineers, who need to know how to configure, support and troubleshoot Outlook clients connected to an OpenMail server.

Revision History

June 1998	First issue of this OTN
June 1999	Second issue incorporating new and changed functionality in version B.05.30 of the MAPI Service Providers: delegates, improvements to remote working, changes to the way reminders work.

Comments Please!

I would welcome any comments you may have on this document. Please email them to joyce@pwd.hp.com.

Contents

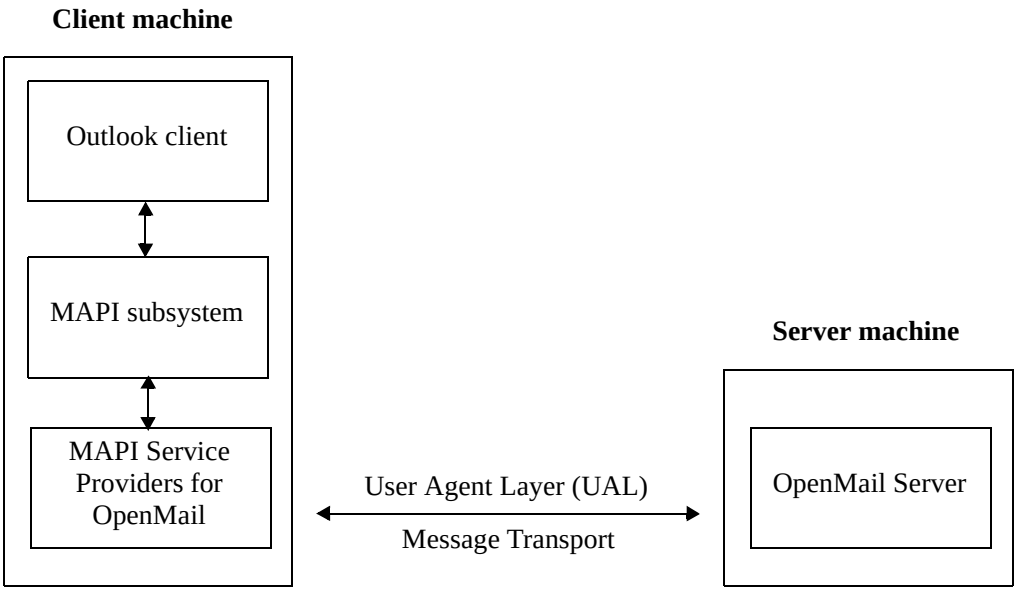
A Brief Introduction To MAPI And Service Providers

MAPI, Messaging Application Programmer's Interface or Messaging API, was developed by Microsoft to allow increasingly feature rich windows messaging clients to integrate more fully with a variety of messaging servers. The API was originally published to enable communication with Microsoft Mail (MS Mail) client. This version of MAPI is now referred to as MAPI-0 or Simple MAPI. It comprised a set of 12 functions that enabled basic messaging features. We initially developed MAPI-0 drivers, so that OpenMail could support Microsoft Mail .

MAPI-0 was then superseded by MAPI-1 or Extended MAPI. This is the messaging API used by the Outlook client, an integrated desktop information manager that supersedes Microsoft Mail, Microsoft Exchange client and Microsoft Schedule + by combining, extending and enhancing the features of these products.

To support the Outlook client, we developed new *Service Providers* based on this new version of MAPI and called the product *Outlook (MAPI) Service Providers for OpenMail.*

Figure 1 MAPI subsystem and OpenMail Service Providers



The MAPI architecture comprises three separate layers; the client application (e.g. Outlook), the MAPI subsystem and the Service Providers that communicate with specific messaging systems. All three layers reside on the client workstation.

The MAPI subsystem comprises runtime interfaces and functions and a MAPI spooler. The runtime libraries are responsible for initializing the MAPI subsystem, starting a MAPI session and administering profiles. The spooler is an independent process that manages the flow of messages in and out of the system. It sets the delivery order of both inbound and outbound messages and routes them to the appropriate message store or transport. The client developer is generally responsible for providing the MAPI subsystem.

Service Providers are replaceable modules that facilitate communication with a specific messaging system component, such as a message store or directory. Service Providers speak a specific communication protocol and provide a specific set of functionality. For a messaging server to support MAPI clients, the developer of the server is generally responsible for producing the necessary Service Providers. The Service Providers are the component of the MAPI architecture responsible for ensuring that commands generated by a MAPI client are properly translated into the protocol used by the messaging server.

Services are usually classified as Address Book, Message Store and Transport. Service Providers can provide any combination of these three principle services. A client mail application may require multiple Service Providers to access all the directories and message stores needed in a single messaging session. The Windows system keeps Service Providers configuration details in the file `\WINNT\SYSTEM32\MAPISVC.INF` (`\WINDOWS\SYSTEM32\MAPISVC.INF` on Windows 95).

A profile is a MAPI construct that lists the messaging services available to a user during a single messaging session. Configuration data associated with each listed messaging service is also stored in the profile. Once a profile has been created, a MAPI client user can communicate with local and remote message stores and directories using the Service Providers listed in the profile. The availability of a particular messaging service during a session is entirely dependent on which services are defined in the profile being used for that session.

Profile management is one of the few elements of the MAPI subsystem that is exposed to the end user. Profile information is stored in the Windows registry and should only be modified through one of the standard Windows interfaces to the MAPI subsystem. The most common way to access MAPI is through the Mail icon (Mail and Fax icon in Windows 95) in the Windows Control Panel.

This, then, is where the OpenMail MAPI Service Providers fits into the overall MAPI architecture. The Service Providers consists of three services:

- Address Book - This is used to access local and remote instances of OpenMail directories.
- Message Store - This allows a MAPI client to access folders and messages in the OpenMail Message Store using a set of MAPI interfaces. A MAPI profile requires a Message Store to be nominated as the default for the Message Store service to use. The user's Inbox, Outbox and special Outlook folders, Calendar, Tasks etc., will all be located in the Default Message Store.
- Transport - This works with the MAPI spooler to send messages to, and receive messages from, the OpenMail Message Store. The spooler and Transport service are mainly used when the OpenMail Message Store is not the default store.

  As I have mentioned, a part of native MAPI is the MAPI spooler - the file `MAPISP32.EXE`. The role of the spooler is to handle the communication between the client and the transport service, so that the client is able to continue processing without interruption. Outgoing messages are sent from the spooler to the Transport service, which converts them into an OpenMail recognizable form before sending them. Incoming messages are received by the Transport service, which notifies the MAPI spooler. The spooler then sends an empty MAPI message container for the Transport service to fill with the OpenMail message details converted into MAPI properties. The client is therefore not speaking directly to the Message Store service but using the spooler for communication; the client tells the spooler that it wants to send a message and the spooler then takes the message and decides which transport service to use to send it

  However, if the OpenMail Message Store is the default store, both the MAPI spooler and Transport service will be used very little, because the OpenMail message store can send messages directly (through the UAL) from the client's outbox and to the client's inbox .

Figure 1 shows a MAPI client (Outlook) communicating with an OpenMail server. Note the OpenMail Service Providers reside on the client machine.The Service Providers are responsible for converting commands issued by the MAPI client into OpenMail UAL commands.

Microsoft refer to a client using MAPI as working in *corporate workgroup mode*. When the same client is using Internet protocols, such as IMAP4, instead of MAPI for message retrieval and submission, it is said to operate in *Internet only mode*. Outlook 97 uses MAPI only. Outlook 98 can use MAPI or IMAP4. With IMAP4, features available are limited to e-mail access and personal information management, that is, workgroup features are not supported. Note that you cannot have Outlook MAPI and IMAP4 profiles on one client machine.

This OTN focuses on OpenMail supporting the Microsoft Outlook client in corporate workgroup mode.

Throughout the OTN, HP/UX is used as the reference server platform and Outlook 98 on Windows NT is used as the client environment. Outlook 97 and Windows 95 may have subtle differences.

References

The section above gives you a brief overview of MAPI architecture and where the Outlook (MAPI) Service Providers for OpenMail fit in. For more information on MAPI, see the MSDN Online Library on the Microsoft website, or refer to the MAPI Software Developer's Kit (SDK).

This OTN is not intended as a tutorial on Outlook. There is a useful white paper (ID: 100-1320) on the OpenMail website (http//:www.hp.com/go/OpenMail), which provides end user instructions for the new and changed features available with the B.05.30 Service Providers. For more general information on how to use the Outlook client, refer to the online help. There are also a number of tutorial publications available from most bookshops and information on Microsoft's website.

For detailed technical specifications for the B.05.30 version of the OpenMail MAPI Service Providers; system requirements, supported operating system platforms, languages, fixes, limitations and known problems, see the following documents on the OpenMail website:

Release 5.30 At a Glance (ID 100-1316)

Focus on Release 5.30 (ID: 100-1323)

B.05.30.00 OpenMail Outlook (MAPI) Service Providers Release Notes (ID: 100-1349)

For details on how to install, configure and use the OpenMail Outlook (MAPI) Service Providers, refer to the *OpenMail MAPI Service Providers Technical Guide* (ID: 100-1332)

Further information on installing and upgrading the OpenMail server is given in *OpenMail Installation Instructions for Unix Systems* (ID: 100-1330).

Reference information on the OpenMail server is given in the *OpenMail Technical Guide* (ID: 100-1330).

For details of OpenMail software licensing, refer to the *OpenMail Licensing Guide* (ID: 100-1330).

The OpenMail UAL is described in the OTN: *UAL and OpenMail Clients* (ID: 300-0003).

For details of OpenMail support of IMAP4 clients, see the OTN: *IMAP4* (ID: 300-0134).

An Overview of the MAPI Service Providers Releases

Previous OpenMail releases have incrementally added functionality to the MAPI support we offer:

- B.04.00 provided the MAPI-1 transport and address book Service Providers for OpenMail.
- B.05.00 added the MAPI-1 message store Service Provider.
- B.05.10 was the first release to support the Microsoft Outlook client, which replaced the Exchange Client as Microsoft's strategic MAPI mail/scheduling client.

  The B.05.10 drivers provided the options for a "with server store" and "without server store" profile, allowing users to store and access their inbox, filing cabinet and bulletin boards on the server or to download their mail to a local personal store (.PST file). Storage of other information such as Calendar, Tasks, Contacts and Notes was restricted to the .PST file. This necessitated that there always be a .PST file in the user's profile.

  Put another way OpenMail could not be the default store for Outlook. Whilst you could store personal calendar information locally, you could not access other people's free/busy time or calendar details across the enterprise.

Release B.05.20 Overview

The B.05.20 release allowed the OpenMail Message Store to be the default store, offering messaging and calendaring functionality across the enterprise.In addition, Tasks, Contacts and Notes could be used for Personal Information Management with the data stored on the server.

The Outlook 98 client was supported with the service providers, but only to the level of Outlook 97 functionality.

In summary, this release added two major pieces of functionality:

- The OpenMail Message Store as default store.
- Enterprise Calendaring across the supported profile types:

  OpenMail as the default store

  .PST as the default store

  (OpenMail and .PST are supported in the same profile, but only for migration purposes).

B.05.20 was, in marketing terms, an OpenMail release. In technical terms the release consisted of two things:

- A server patch, released initially as Periodic Patch No. 4
- The Service Provider component for the PC client

Release B.05.30 Overview

Release B.05.30 delivered:

- Delegate functionality
- Significant improvements to Remote Working
- Defect fixing

As with B.05.20, release B.05.30 comprises both an OpenMail server component and the Service Provider component for the PC. There are two ways of upgrading to release B.05.30:

1. On the server - upgrade to OpenMail B.05.10 with PP-March99 (or later) installed.

   On the PC - install the B.05.30 MAPI Service Providers.

2. On the server - upgrade to OpenMail version B.06.00.

On the PC - install the B.05.30 MAPI Service Providers.

The B.05.30 MAPI Service Providers are available on CD or can be downloaded as a zip file from the OpenMail web site.

Important! You need a username and password to download the Service Providers. Only customers with a Media Subscription Service contract, or a current support contract for software updates, and authorized personnel, are issued with a download username and password.

As a general rule, the version of the MAPI Service Providers in a client must be the same or lower than the version of OpenMail on the connected server. So B.05.20 or B.05.30 Service Providers with a B.05.30 compatible or B.06.00 OpenMail server are supported combinations but B.05.30 Service Providers with a B.05.20 cmpatible server is not supported.

Using Different Versions of Outlook

You can use Outlook 98 or Outlook 97 with the B.05.30 Service Providers. To use the OpenMail Outlook delegate functionality with Outlook 97, you will need version 8.04 of Outlook 97 from Microsoft. This version is available from the Microsoft Office 97 Service Release 2 (SR-2). See the Microsoft website, http://www.microsoft.com, for more information.

In Outlook 98, we support the Outlook 97-level MAPI functionality but not the new Outlook 98 features, such as Outlook Today, HTML mail editing, Rules Wizard, Junk E-mail Manager and Automatic Formatting (Colors).

Installing the OpenMail MAPI Service Providers

An InstallShield setup program is provided for the installation of the Service Providers. The image contains the directories "disk1" and "disk2" and the `setup.exe` program can be found under "disk1".

This installs the necessary DLLs into the appropriate Windows system directory for the platform you are running on (NT or Win95):

`omaclx32.dll`	handles MAPI permissions
`omfb32.dll`	provides free/busy functionality
`omlink.dll`	provides server link functionality
`ommapi.cnt` and `ommapi.hlp`	help files
`ommapi32.dll`	
`omname32.dll`	address parsing and mapping
`omooa32.dll`	out of office assistant
`omtf.dll`	handles OpenMail transaction files
`omual.dll, omualo32.dll`	handle UAL commands
`ualwsd32.exe`	the client UAL process
`omvers.dll`	version information
`omdlgt32.dll`	provides delegate functionality
`omext32.dll`	provides OpenMail MAPI extensions to menus etc.
`ommotd32.dll`	provides message of the day functionality

It will also make the necessary changes to the Registry. The Registry settings for the OpenMail MAPI Service Providers are under the following paths:

```
HKEY_LOCAL_MACHINE\Software\Hewlett-Packard\OpenMail\MAPI\Local
HKEY_CURRENT_USER\Software\Hewlett-Packard\OpenMail\MAPI\LogFile
```

To check the providers have been installed:

- Go to Settings -> Control Panel and click on the Mail icon.
- Click the Show Profiles button.
- Click Add and you should see

  `HP OpenMail (with server store)`

  `HP OpenMail (without server store)`

  listed as possible information services (see Figure 2 on page 10). If you don’t see these, then the OpenMail MAPI Service Providers have not been correctly installed.

If you want to check which version of the Service Providers are installed:

- Locate `ommapi32.dll` in the Windows System directory.
- Right click on this and select Properties.
- Click on the Version tab.

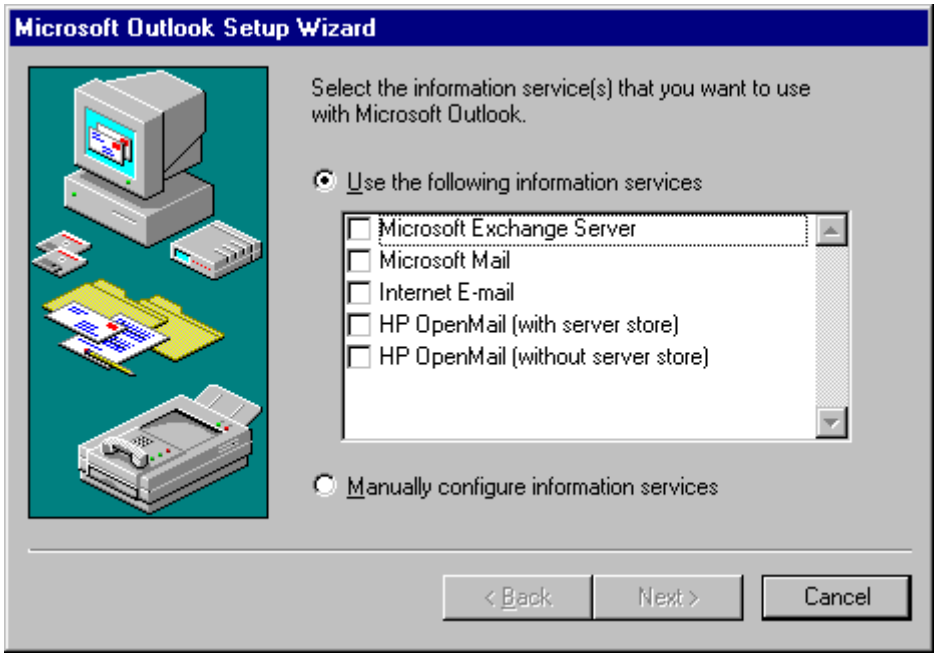
Alternatively, if you have an OpenMail profile already set up, select this (via the Mail icon, then Properties). Highlight the OpenMail information service and click About.

Service Providers Configuration Details

When the Service Providers are installed, the configuration details for each service provider are written to the file `mapisvc.inf` in the `system32` folder. You cannot edit this file. When you create a new profile, details of the information services available (see the screen below) are obtained direct from the `mapisvc.inf` file.

If this file becomes corrupt, reinstall the Service Providers.

Figure 2



Below is an excerpt from a `mapisvc.inf` file. Note the OpenMail service provider configuration details at the end:

```
[MSPST MSP]
PR_PROVIDER_DLL_NAME=mspst.dll
PR_RESOURCE_TYPE=MAPI_STORE_PROVIDER
PR_RESOURCE_FLAGS=STATUS_DEFAULT_STORE
PR_DISPLAY_NAME=Personal Folders
PR_PROVIDER_DISPLAY=Personal Folders
PR_MDB_PROVIDER=4e495441f9bfb80100aa0037d96e0000
34140102=4e495441f9bfb80100aa0037d96e0000
[MSPST MS]
PR_DISPLAY_NAME=Personal Folders
PR_SERVICE_DLL_NAME=mspst.dll
PR_SERVICE_ENTRY_NAME=PSTServiceEntry
Providers=MSPST MSP
PR_SERVICE_SUPPORT_FILES=mspst.dll
PR_RESOURCE_FLAGS=SERVICE_NO_PRIMARY_IDENTITY
PR_MDB_PROVIDER=4e495441f9bfb80100aa0037d96e0000
.
.
.
.
[OpenMail]
PR_SERVICE_NAME=HP OpenMail (with server store)
PR_DISPLAY_NAME=OpenMail
Providers=OM_AB, OM_MS, OM_XP <---OM_AB=Address Book, OM_MS=Message Store,
                                  OM_XP=Transport Provider
PR_SERVICE_DLL_NAME=ommapi.dll
PR_SERVICE_SUPPORT_FILES=ommapi.dll
```

```
PR_SERVICE_DELETE_FILES=ommapi.dll
PR_SERVICE_ENTRY_NAME=ServiceEntry
PR_RESOURCE_FLAGS=SERVICE_SINGLE_COPY | SERVICE_PRIMARY_IDENTITY
WIZARD_ENTRY_NAME=WizardEntryWithStore
[OpenMail (without server store)]
PR_SERVICE_NAME=HP OpenMail (without server store)
PR_DISPLAY_NAME=OpenMail (without server store)
Providers=OM_AB, OM_XP
PR_SERVICE_DLL_NAME=ommapi.dll
PR_SERVICE_SUPPORT_FILES=ommapi.dll
PR_SERVICE_DELETE_FILES=ommapi.dll
PR_SERVICE_ENTRY_NAME=ServiceEntry
PR_RESOURCE_FLAGS=SERVICE_SINGLE_COPY | SERVICE_PRIMARY_IDENTITY
WIZARD_ENTRY_NAME=WizardEntryWithoutStore
[OM_XP]
PR_PROVIDER_DISPLAY=HP OpenMail Transport
PR_DISPLAY_NAME=OpenMail
PR_PROVIDER_DLL_NAME=ommapi.dll
PR_RESOURCE_TYPE=MAPI_TRANSPORT_PROVIDER
PR_RESOURCE_FLAGS=STATUS_PRIMARY_IDENTITY
[OM_MS]
PR_PROVIDER_DISPLAY=HP OpenMail Message Store
PR_DISPLAY_NAME=Mailbox
PR_PROVIDER_DLL_NAME=ommapi.dll
PR_RESOURCE_TYPE=MAPI_STORE_PROVIDER
PR_RESOURCE_FLAGS=STATUS_PRIMARY_IDENTITY | STATUS_DEFAULT_STORE
PR_MDB_PROVIDER=e0a88720fa6411ceb0160800098b9b76
[OM_AB]
PR_PROVIDER_DISPLAY=HP OpenMail Address Book
PR_DISPLAY_NAME=OpenMail
PR_PROVIDER_DLL_NAME=ommapi.dll
PR_RESOURCE_TYPE=MAPI_AB_PROVIDER
PR_RESOURCE_FLAGS=STATUS_PRIMARY_IDENTITY
[OM_MS_DELEGATE]
PR_PROVIDER_DISPLAY=HP OpenMail Delegate Message Store
PR_PROVIDER_DLL_NAME=ommapi.dll
PR_RESOURCE_TYPE=MAPI_STORE_PROVIDER
PR_RESOURCE_FLAGS=STATUS_NO_DEFAULT_STORE
PR_MDB_PROVIDER=007828048099d111b51d080009b149da
```

Removing the OpenMail MAPI Service Providers

Warning! It is not recommended that you remove the OpenMail MAPI Service Provider software simply by deleting the component files. Many of these, especially those with the extension.DLL, may be used by other local applications their removal can cause problems.

If you need to remove the OpenMail MAPI Service Provider software from a PC, use Add/Remove Programs option from the Control Panel. Ensure you remove the appropriate Registry settings as well.

Adding Outlook Users and Profiles

When you have ensured the version of OpenMail running on the server is correct and have successfully installed the Service Providers on the client, you can create an OpenMail user and profile. The OpenMail users are created on the server using `omaddu` and the profiles are created on the client PC.

A Brief Look At Some FREEBUSY Issues

We'll look at free/busy information in more detail later, but for the moment it is worth explaining what you need to do at setup, if users are to be able to share calendaring information.

The `FREEBUSY` directory is a normal Openmail directory. Users must have an entry created in this directory, on their local server, if they are to be able to publish or view Free/Busy information.

At release B.05.20, if this directory did not already exist on the server, it was created automatically by a script that was run from the server patch you needed to install. At B.05.30, you need to use the `omaddfb` command on the server to create a `FREEBUSY` directory. Just typing:

```
omaddfb
```

will create the `FREEBUSY` directory and populate it with an entry for every local user on the local system.Typing:

```
omaddfb -n
```

creates the directory but does not populate it. You would then need to add entries manually by using the `-F` option in the `omaddu` or `ommodu` command:

```
omaddu -n "Mr Mapi/canberra,class" -p mapi -F
 omaddu : The user was added successfully
omsearch -d FREEBUSY -e "S=mapi/g=mr"
 S=Mapi/G=Mr/OU1=canberra/OU2=class
```

Each entry in the `FREEBUSY` directory must include all the addressing attributes that appear in the `SYSTEM` default directory (as defined by the hidden directory `USERLIST`). All addressing attributes, as returned by `-m @X400-ATTR@`, are included in the entry.

If your `SYSTEM` default directory contains additional addressing attributes for your users, which would not appear in the `USERLIST` entry, for example X.400 CAPO (Country, ADMD, PRMD, Org) attributes or DDA (Domain Defined Attribute) values, you will need to update the `FREEBUSY` entries accordingly. If you do not do this, publishing and lookup of Free/Busy information will not work.

If, for example, there is a local user:

```
omshowu -n "Mr X400/canberra,class"

   User Name : Mr X400
   MailNode : canberra,class
   System Login : mrx400
   Password : set
   Admin Capabilities : NO
   Language : C
   Access : Normal
   Mail Account: Unlocked
   Last Signon : 01.01.70 00:00:00
```

The `USERLIST` entry for this user would be:

```
S=X400/G=Mr/OU1=canberra/OU2=class
```

However, the SYSTEM default directory entry is:

```
omsearch -e "S=x400" -m @X400-ATTR@
   S=X400/G=Mr/OU1=canberra/OU2=class/C=GB/A=400NET/P=HP/O=HP
```

If an entry for this user was created automatically, the entry in the FREEBUSY directory would read:

```
S=X400/G=Mr/OU1=canberra/OU2=class
```

When a request is made for the free/busy times of this user, the SYSTEM directory is accessed first to get the fully qualified address of the user, Mr X400:

```
S=X400/G=Mr/OU1=canberra/OU2=class/C=GB/A=400NET/P=HP/O=HP
```

Then the FREEBUSY directory is searched for that user. As Mr X400's entry in the FREEBUSY directory differs:

```
S=X400/G=Mr/OU1=canberra/OU2=class
```

this entry will be regarded as belonging to a different user and Mr X400 will not be found in the FREEBUSY directory.

To fix this, either add the required attributes to the FREEBUSY entry (using ommodent), or delete this entry and re-add it, as follows:

```
omdelent -d FREEBUSY -e "S=X400/G=Mr/OU1=canberra/OU2=class"
```

```
omsearch -e "S=X400/G=Mr/OU1=canberra/OU2=class" -m @X400-ATTR@ \
   > | omaddent -d FREEBUSY -f -

omsearch -d FREEBUSY -e "S=X400/G=Mr/OU1=canberra/OU2=class"
   S=X400/G=Mr/OU1=canberra/OU2=class/C=GB/A=400NET/P=HP/O=HP
```

It is important to understand this relationship between USERLIST, SYSTEM and FREEBUSY, as it effects both the publishing and lookup of Free/Busy information.

If you use omshowlog to look at what the omaddfb command does, you will see it gives update permissions to all users:

```
REPORT                          Administration(ommodacln     ) 06.15.99
15:02:55
[OM 4630] Admin command run :
ommodacln -t dir -l FREEBUSY -g default -c +update
```

The reason for this is that in the above environment, where there is a mismatch between the USERLIST and FREEBUSY entries, "modifyself" permission would not be sufficient to allow a user to update their free/busy information since:

```
S=X400/G=Mr/OU1=canberra/OU2=class
```

does not match in "modifyself" terms

```
S=X400/G=Mr/OU1=canberra/OU2=class/C=GB/A=400NET/P=HP/O=HP
```

There is one more aspect of free/busy that we need to consider at this point; when an Outlook user requests the free/busy time of another user, the other user may reside on the same machine or on a different machine in the network. We, therefore, need a way to map an OpenMail mailnode to the physical machine hosting that mailnode (and therefore the FREEBUSY directory). The file that does this, ~openmail/sys/mnMapFile, is created and updated automatically as required.

We will look into this whole area in more detail when we look at the calendaring functionality.

You will also see the following file created:

```
~openmail/sys/mapiorn.ids
```

You need not worry about this file. Its purpose is to identify MAPI properties that contain ORNs (Originator/Recipient Names), as these require particular character set conversion handling.

Profiles

Now that we have created a user on the server, we shall create a profile for this user on the PC.

Outlook profiles and the functionality offered by different combinations of information services in a profile is well documented in the *OpenMail 5.30 White Paper* and Chapter 3 of the *OpenMail MAPI Service Providers Technical Guide*. The table below just outlines some combinations and the types of users who might want to use them.

Information Services in a Profile	**Examples of Use**
HP OpenMail (with server store)	Office based users
HP OpenMail (with server store) as default + Personal Folders (.pst)	Users migrating from .pst to server based store. Users who want to restore data from a .pst file.
Personal Folders (.pst) configured as default store + HP OpenMail (with server store)	Occasional mobile users. Users who want to retain their calendar data locally.
Personal Folders (.pst) configured as default store + HP OpenMail (without server store)	Travelling users. Remote users. Users who want to retain message store data locally.

When you create an OpenMail (with server store) profile, OpenMail information about that user's Special folders is stored in the profile. This means you cannot subsequently modify the Principal Name and Server assigned to that profile. However, if you create an OpenMail (without server store) profile, you can use the same profile to logon to a different server as a different user.

Use the following steps to create a profile with the OpenMail server as default store:

1. Go to Settings -> Control Panel
2. Select the Mail icon
3. Click on Show Profiles
4. Click Add
5. Check only the box for "HP OpenMail (with server store)"
6. Click "Next >"
7. Give the profile a name
8. Enter the Server name for the user
9. Enter the Principal Name (i.e. user)
10. Enter the password for the user
11. Finished

On the PC user profiles are held in the registry can be seen using Regedit on NT under:

```
HKEY_CURRENT_USER
          \Software
            \Microsoft
              \Windows NT
                \Current Version
                  \Windows Messaging Subsystem
                    \Profiles
                      \<user name>
```

and on Windows 95 under:

```
        HKEY_CURRENT_USER
          \Software
            \Microsoft
              \Windows Messaging Subsystem
                \Profiles
                  \<user name>
```

Each profile has a `.FAV` file, `<profile name>.FAV`, associated with it, which is normally found under `C:\WINNT` or in Windows 95, `C:\WINDOWS`. This contains the Outlook Bar Shortcuts. Under certain circumstances, if you delete the `.FAV` file, you get a message when you restart Outlook telling you that it is going to rebuild the shortcuts for you.

If you delete a profile, it is a good idea to delete the associated `.FAV` file.

Automatic Profile Generation

Two utilities are available for automatic creation of Outlook profiles:

- Microsoft's Automatic Profile Generator (Newprof) utility, which is supplied with Outlook. With Newprof, users do not need to create their own profiles after Outlook has been installed. The Newprof application takes as input a `.PRF` file that describes the services to be included in the profile. This file also contains the settings for these services. The result of running Newprof on this `.PRF` file is a profile that contains the required services with the defined settings. A custom `.PRF` file can also be used as part of the installation of Outlook to create automatically a custom default profile.

  You can run Newprof from the command line with a variety of distribution mechanisms, such as SMS, logon scripts and e-mail. If you use it with logon scripts, it can be used to create default profiles for "roving" users when they logon to a different computer.

  There is a section in the *OpenMail MAPI Service Providers Technical Guide*, which gives advice on using Newprof to generate OpenMail profiles automatically. This needs to be read together with information on the Microsoft website :
  `http://www.microsoft.com/outlook/adminguide/CH02.htm#ch2 05d`

- Profile Maker from Automatic Profile Management, LLC. The OpenMail 5.30 White Paper contains an overview of this utility. For more information on Profile Maker, see

  `http://www.autoprof.com/profmkr.htm`

  For details on how Profile Maker supports MAPI configurations for OpenMail, see the online configuration guide:

  `http://www.autoprof.com/guide/hp_openmail.htm`

MAPI Properties

Any MAPI object, such as a message, is a collection of MAPI properties. Properties are just data structures containing the raw data and information on how to interpret it. Taking a message as an example, the subject and the creation date are regarded as properties. The message also contains a table (list) of recipients. Each row in the table is a collection of properties that describe that recipient. Similarly, all the attachments for a message are held in an Attachment Table and each row in the table is a collection of properties that describe that attachment. So, as far as MAPI is concerned, there is no other data apart from properties.

There are many MAPI properties defined by MAPI. These are listed in the MAPI SDK documentation. In addition, clients and service providers can extend these using custom properties.

Properties can be persistent, in that they are stored and extend across sessions, or temporary. They can be read only or read/write, as defined by the implementor.

A property can be a single value or an array of values. For a single value, three pieces of information are stored:

- The *value* of the property
- The data *type* of the value, such as boolean or integer
- A numeric value that uniquely identifies the property (*identifier*)

The type and identifier are combined to form the property tag and it is this which is used to refer to the property.

An example of a MAPI Property is

`0x0037001e`

which is a combination of the type and identifier of the property. The type, `0x001e`, indicates the property type is a string (PR_TSTRING). The identifier, `0x0037`, indicates this is a Subject property. The mnemonic description given to this property is `PR_SUBJECT`.

The type/identifier order will vary depending on where you are looking at these properties. For example, in the registry you will see a number of settings starting `001e` (PR_TSTRING):

Tag	**Value**
`001e6612`	`Mr Mapi/canberra,class`

`0x6612` is the identifier for PR_TR_FULL_PRINCIPAL_NAME in the profile.

In a UAL Trace you might see the following returned as part of `ListMAPIProps`

Tag	**Value**
`3613001e`	`^]IPF.Appointment`

`0x3613` is the identifier for PR_CONTAINER_CLASS. `^]` is a representation of the group separator character `0x1d`).

Property Storage

In the OpenMail server, MAPI properties are stored in various places; sometimes in container extension files, sometimes as an object file and sometimes in a transaction file, for example, when the message is being mailed. In general, OpenMail does not care what any of this data is, although Local Delivery will generate some MAPI properties, such as PR_DISPLAY_TO, when attaching a message to a mailbox.

Generically you will hear people talk about MAPI *blobs*. There a number of blobs:

- *List Properties*

  These properties are generally held on a per message basis, and exist at the list, or content table, level. An example of a list property is the message subject.

- *Relationship Properties*

  These properties define the relationship between the user and a message. This is important because, in OpenMail, the same message can be shared by more than one mailbox. If one user makes a change to the message, sets a read flag, for instance, it must not affect all the other users who are sharing it. These properties are, therefore, held on a per user basis and exist at the parent level.

- *Computed Properties*

  These are properties that are created “on the fly”. For example, when delivering a message, local delivery computes the property `PR_DISPLAY_TO` to enable easy access to this information by the Service Providers.

All of the above properties are held in container extension files in the OpenMail message store. We can use `omcontain` to display them using the `lm` (List MAPI) option.

Other MAPI properties, such as the recipient and attachment tables and properties describing icon placement and OLE attachments, are held in an object file attached to the OpenMail message. This was the `WINMAIL.DAT` attachment in previous releases, also known as the TNEF (Transport Neutral Encapsulated Format) file. It is not possible to view this using `omcontain`. We will look at configuration options for this file later in the OTN.

When a message is mailed, the list and relationship properties and the properties in the TNEF object file, are mailed with the message.

A message in the OpenMail message store has attributes associated with it. Where there is a correlation between a MAPI property and an OpenMail attribute these are mapped.

There may be cases where the value of a MAPI property may differ depending on whether you are looking at a message on the server or the client.

Also, there are instances where different MAPI properties are shown depending on which view you are using. For example, when listing the Intray, the "From" column shows the property `PR_SENT_REPRESENTING_NAME` , when you open the message the "From:" field is shown as just `PR_SENDER` or `PR_SENDER` "on behalf of "`PR_SENT_REPRESENTING_NAME` if these two properties are different.

OpenMail As The Default Store

If we look on the server at the user we have created, then we see the normal OpenMail message store structure. For example

```
# omcontain
WARNING: This is a diagnostic tool for use by HP trained personnel.
    Improper use can cause serious damage.
    If you do not wish to continue, hit return now.

.

.

.

Option?u
User:mr mapi
User Name = Mr Mapi /canberra,class
User Number = 104
Ctr Handle = 3001
Done
Option?l
Ctr Handle (3001):
   1)User 104 Intray                        (IN TRAY)
   2)User 104 OutTray                       (OUT TRAY)
   3)User 104 Pending Tray                  (PENDING TRAY)
   4)User 104 Filing Tray                   (FILING AREA)
   5)User 104 ListArea Tray                 (LIST AREA)
Done
```

Certain Outlook folders are mapped directly onto exisiting OpenMail folders:

Outlook	**OpenMail**
Inbox	Intray
Outbox	OutTray
Sent Items	Pending Tray
Deleted Items	WASTE BASKET in the Filing Tray

The other Outlook special folders:

```
Calendar
Contacts
Tasks
Journal
Notes
Drafts (Outlook 98)
```

are created under the Filing Tray when the user first logs in with Outlook.

So, if we take a look in the Filing Tray now:

```
Option?o
Ctr Handle (3001):
Entry Number:4
Ctr Handle = 3002
Done

Option?l
Ctr Handle (3002):
Done
```

we can see that there is currently nothing created under this area.

On the server, ensure that the following services are running: Remote Client Interface (rci), Client Directory Access server (cda), Background Search Server (search) and the Directory Relay Server (drs).

Now login to this user from Outlook using the default store profile we created. If we take another look in the Filing Tray:

```
Option?o
Ctr Handle (3001):
Entry Number:4
Ctr Handle = 3002
Done

Option?l
Ctr Handle (3002):
   1) WASTE BASKET                          (FOLDER)
   2) Search Area                           (FOLDER)
   3) MAPI Root Folders                     (FOLDER)
   4) Calendar                              (FOLDER)
   5) Contacts                              (FOLDER)
   6) Journal                               (FOLDER)
   7) Notes                                 (FOLDER)
   8) Tasks                                 (FOLDER)
   9) Drafts                                (FOLDER)

Done
```

WASTE BASKET, Search Area and MAPI Root Folders are created by OpenMail. The other folders (Outlook special folders) are created on the server by the MAPI Service Providers to hold the special folder contents for this user.

MAPI Properties are written to the User Folder, the Intray and the folder itself. Basically the absence of MAPI Properties on the User Folder and Intray indicate that the special folders still need to be created. When the special folders are created, their own MAPI Properties are updated. In addition, the MAPI Properties of the User Folder and Intray are updated to reflect the Direct References of the special folders.

Direct References

Direct References, also sometimes referred to as *Direct Indices*, refer to an OpenMail feature used a lot in the OpenMail MAPI implementation. A Direct Reference is a unique 16 character hexadecimal number that corresponds to a single item (not a file) in the OpenMail message store. It allows direct access to the item without having to work through the hierarchy of folders and messages to get to it.

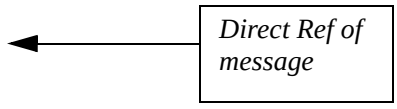
The mapping of Direct References to items is held in *Direct Index Tables* (dits). The files containing these tables are on the server, in the `~openmail/dits/` directory.

Warning! Do not change anything in this directory.

In the example below, I have used `omcontain` to show the Direct Reference of a message in Mr Mapi's Intray:

```
Option?u
User:mapi

User Name = Mr Mapi /canberra,class
User Number = 104
Ctr Handle = 3001
Done

Option?o
Ctr Handle (3001):
```

```
Entry Number:1
Ctr Handle = 3002
Done

Option?l
Ctr Handle (3002):
    2) from remote                                  (MESSAGE)
    3) Messages                                     (MESSAGE)
Done

Option?r
Ctr Handle (3002):
Entry Number:2

Item Type: MESSAGE
Subject: from remote
Subject Charset: IA5
T61Subject:
T61Subject Charset:
Creator: Remote/Mr///canberra/class
Creator Id: 105
File Name: ~/data/0000004/00001e7:2
Item Number: 1396
Num Components: 2
Created: 06.18.99 16:39:50
Attached: 06.18.99 16:39:56
Received: 06.18.99 16:39:55
Last Recorded Modify: 06.18.99 16:39:55
Content Flags: 0x00000020, 0x00001480
DEL_ENV_PRESENT
NO_DN_REQUESTED
NO_MAPI_ATTACH
NO_SUB_FOLDER
Priority: 0
Category: 1
Requested Ack: 0
Sent Ack: 0
Entry Status: 3
Msg ID:
H0000069000005a1.0929720388.kuda.pwd.hp.com
Ack ID:
Direct Index: 70684 (67584) 0001141c34937c44
Item Charset: IA5
MAPI List Props Offset 42, Len 709
MAPI Comp Props Offset 43, Len 37
MAPI Rel  Props Offset 44, Len 54
Done
```



Using the new `omdref` command, in `/opt/openmail/diag`, you can identify the item in the message store from the Direct Reference, for example:

```
root@kuda[dits] #omdref 0001141c34937c44
USER FOLDER: Mr Mapi / canberra, class
  IN TRAY: User 104 Intray                                : RecNo : ItemNo
    MESSAGE: from remote                                    :     2 : 1396
```

`RecNo` is the container record number and `ItemNo` is this item's unique number within the message store.This is the same as the Item Number field displayed when an item is read (`r`) in `omcontain`.

MAPI Special Folders In The OpenMail Message Store

We can use the listMapi (`lm`) option in `omcontain` to list the MAPI properties in the folders. To list the properties (and Direct References to the special folders) in the User Folder:

```
Option?u
User:Mr Mapi

User Name = Mr Mapi /canberra,class
User Number = 104
Ctr Handle = 3001
Done

Option?lm
Ctr Handle (3001):
Entry Number:0
===============
MAPI List props
===============
VERSION=B.05.30.01
CHARSET=ISO8859_1
36e41102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@0001000d6610809f\@==
36d70102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@0001000a69f9a10f\@
36d50102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@0001000911adc859\@
360a000b=1
36d40102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@00010008292ed833\@
36d30102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@0001000703e8973e\@
36d20102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@000100063ee1d1cf\@
36d10102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@000100055863d082\@
36d00102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@00010004767fde45\@
GUID=\B \F\@\@\@\@\@À\@\@\@\@\@\@F
#b/8223=0
GUID=\H \F\@\@\@\@\@À\@\@\@\@\@\@F
#b/8503=0
36010003=0
34140102=à¨‡ úd\QÎ°\V\H\@\I‹›v
===============
MAPI Comp props
===============
VERSION=B.05.30.01
CHARSET=ISO8859_1
ffa0102=S_öï·$Ó\Q„¿\@`°)\^t
ff90102=\@\@\@\@ÃÂN Ñ\Qµ\]\H\@\I±IÚ\B\@\@\@USER_FOLDER\@
ffe0003=3
Done
```

The lines underlined above contain the Direct References to the MAPI special folders. The first 8 digit number on these lines is the hex tag for the MAPI Property. As with other MAPI Property tags, this tag is made up of the Property identifier and type information. Take the tag `36d70102`, for example:

`0x36d7` is the identifier of the Property, `PR_IPM_DRAFT_ENTRYID`

`0x0102` indicates this is a binary Property

The table below gives the description for the identifiers in the output above:

Hex MAPI Property Identifier	MAPI Property Description
36d0	`PR_IPM_APPOINTMENT_ENTRYID`
36d1	`PR_IPM_CONTACT_ENTRYID`
36d2	`PR_IPM_JOURNAL_ENTRYID`

Hex MAPI Property Identifier	MAPI Property Description
36d3	PR_IPM_NOTE_ENTRYID
36d4	PR_IPM_TASK_ENTRYID
36d5	PR_REM_ONLINE_ENTRYID (Reminders Folder in MAPI Root Folders)
36d7	

At the end of each underlined line is the Direct Reference to the special folder:

Direct Reference	Special Folder Referenced
0001000a69f9a10f	Drafts folder
0001000911adc859	Reminders Folder in MAPI Root Folders
00010008292ed833	Tasks
0001000703e8973e	Notes
000100063ee1d1cf	Journal
000100055863d082	Contacts
00010004767fde45	Calendar

If we look at the Properties in the Intray, we will see these Properties duplicated:

```
Option?l
Ctr Handle (3001):
    1) User 104 Intray                      (IN TRAY)
    2) User 104 OutTray                     (OUT TRAY)
    3) User 104 Pending Tray                (PENDING TRAY)
    4) User 104 Filing Tray                 (FILING AREA)
    5) User 104 ListArea Tray               (LIST AREA)
Done

Option?o
Ctr Handle (3001):
Entry Number:1
Ctr Handle = 3002
Done

Option?lm
Ctr Handle (3002):
Entry Number:0
===============
MAPI List props
===============
VERSION=B.05.30.01
CHARSET=ISO8859_1
36eb0102=\@\@\@\@
GUID=\B \F\@\@\@\@\@À\@\@\@\@\@\@F
#b/8223=0
GUID=\H \F\@\@\@\@\@À\@\@\@\@\@\@F
#b/8503=0
34140102=à¨‡ úd\QÎ°\V\H\@\I‹›v
36010003=1
36d00102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@00010004767fde45\@
```

```
36d10102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@000100055863d082\@
36d20102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@000100063ee1d1cf\@
36d30102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@0001000703e8973e\@
36d40102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@00010008292ed833\@
36d50102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@0001000911adc859\@
36d70102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@0001000a69f9a10f\@
36e41102=\@\@\@\@S_öï·$Ó\Q„¿\@`°)\^t\B\@\@\@0001000d6610809f\@==
360a000b=0
Done
```

We believe Outlook duplicates the Properties on the User Folder and Intray to enable access to the Mailbox and Intray independently.

Using the `omdref` command, you can identify the item in the message store from the Direct Reference, for example:

```
root@kuda[] #omdref 00010004767fde45
USER FOLDER: Mr Mapi / canberra, class
  FILING AREA: User 104 Filing Tray                    : RecNo : ItemNo
    FOLDER: Calendar                                   :     4 :1205
```

Running the `lm` option in `omcontain` on the specific special folders, (e.g. Calendar) you see:

```
Option?l
Ctr Handle (3002):
    1) WASTE BASKET                              (FOLDER)
    2) Search Area                               (FOLDER)
    3) MAPI Root Folders                         (FOLDER)
    4) Calendar                                  (FOLDER)
    5) Contacts                                  (FOLDER)
    6) Journal                                   (FOLDER)
    7) Notes                                     (FOLDER)
    8) Tasks                                     (FOLDER)
    9) Drafts                                    (FOLDER)
Done

Option?o
Ctr Handle (3002):
Entry Number:4
Ctr Handle = 3003
Done

Option?lm
Ctr Handle (3003):
Entry Number:0
===============
MAPI List props
===============
VERSION=B.05.30.01
CHARSET=ISO8859_1
3613001e=IPF.Appointment
GUID=\B \F\@\@\@\@\@À\@\@\@\@\@\@F
#b/8223=0
GUID=\H \F\@\@\@\@\@À\@\@\@\@\@\@F
#b/8503=0
360a000b=0
36010003=1
34140102=à¨‡ úd\QÎ°\V\H\@\I‹›v
Done
```

In the line underlined, `3613` is the identifier for `PR_CONTAINER_CLASS` and `001e` indicates that this is a string property. In each of the special folders this tag is set as follows:

Reminder: `3613001e=Outlook.Reminder`

Calendar:	`3613001e=IPF.Appointment`
Contacts:	`3613001e=IPF.Contact`
Journal:	`3613001e=IPF.Journal`
Notes:	`3613001e=IPF.StickyNote`
Tasks:	`3613001e=IPF.Task`
Drafts	`3613001e=IPF.Note`

The folder called MAPI Root Folders in the Filing Tray contains two further folders:

```
Option?o
Ctr Handle (3002):
Entry Number:3
Ctr Handle = 3003
Done

Option?l
Ctr Handle (3003):
   1) MAPI Common Views                          (FOLDER)
   2) Reminders                                  (FOLDER)
Done
Option?
```

Both of these folders are for Outlook's own use and are not visible to the user through the client. The Common Views folder is used to store the user's personal view created by selecting View -> Define Views. The reminders folder is not used in release B.05.30. In release B.05.20, it was used to store the search specification file used to trigger reminders. We shall look at reminders later.

MAPI Properties for the User Folder, Intray and Special Folders are stored in the profile you used to login to the user. If you remove the MAPI properties from the User Folder and Intray, this Direct Reference information will be lost. Similarly, if you delete and recreate a user on the server, the Direct References will be different for the new user. If you then use the original profile to login to the newly created user, the Direct References on the server and in the profile will be out of sync and you will see the following message.

```
"The location messages are delivered to has changed for this user profile.
To complete this operation, you may need to copy the contents of the old
Outlook folders to the new Outlook folders. For information about how to
complete the change of your mail delivery location, see Microsoft Outlook
Help. Some of the shortcuts on the Outlook Bar may no longer work. Do you
want Outlook to recreate your shortcuts? All shortcuts you have created
will be removed."
```

Deleting the profile on the PC and recreating it does not have the same effect; in this case, there is no change on the server, that is, the Direct References will be the same.

Mailing A Message

Let’s look at an example of mailing a simple message - the body of the message is in Rich Text Format (RTF) rather than plain text.

We’ll turn the Service Router off, so that we can look at the message in transit using `omqdump`.

In the transaction file there are some new records....

```
HEADER            (DN)  1  0  0  0  0x2000000  0x3  0  0x0
  ** Warning - Insufficient fields present  (Record No. 1)
CREATOR           (DN)  0  102  0  0  0  0  "S=Admin/G=Mr/OU1=canberra/ \
OU2=class"  ""  ""
SUBJECT           (DN)  "golf tomorrow?"  ""  "IA5"  ""  ""
MSG_CLASS         (DN)  "IPM.Note"
CREATE_DATE       (DN)  99/6/22 15:07.15+60
MSG_INT_ID       (DN)  0  "H0000066000005d1.0930064033.kuda.pwd.hp.com" \
"H00\00066000005d1.0930064033.kuda.pwd.hp.com"
MSG_OBJECT_FILE   (DN)  595  0  "MAPI.ObjectProperties"
MSG_OBJECT_DATA   (DN)  0x1  1  0  "789f3e22e4b401069008000400.........
.........................
.........................
.........................73000b000f0e000004805767"
MSG_MAPI_PROPS_INFO (DN)  0  0  0  375
MSG_MAPI_PROPS_DATA (DN)  0x1  1  0  "56455253494f4e1.........
.........................
.........................
.........................01b4060b0291b5e74"
MSG_MAPI_USER_PROPS_INFO (DN)  0  0  0  54
MSG_MAPI_USER_PROPS_DATA (DN)  0x1  1  0
"56455253494f4e1d422e30352......0303030331d6666666666666666"
ORIGINAL_EITS     (DN)  0  0  0  0  1  "1.2.840.113543.4.2130"
ITEM_CREATE_DATE  (DN)  99/6/22 15:07.14+60
CONTENT_FILE     (DN)  1166  1166  0x2  131  0  "MAPI-1 Distribution List"
""  ""  ""  ""  "IA5"  ""  ""  ""  ""  ""  ""  ""  ""  ""
ITEM_CREATE_DATE  (DN)  99/6/22 15:07.14+60
CONTENT_FILE      (DN)  2130  2130  0x0  189  0  "golf tomorrow?"
"BDY.RTF"  \
""  ""  ""  "IA5"  ""  ""  ""  ""  ""  ""  ""  ""  ""
RECIPIENT         (DN)  0x2020  0  1  ""  ""  "S=Mapi/G=Mr/OU1=canberra/\
OU2=class"
OPERATION_TRACE   (DN)  99/6/22 15:07.15+60  5  ""
****
```

We can see here the TNEF object file:

```
MSG_OBJECT_FILE   (DN)  595 0  "MAPI.ObjectProperties"
```

and the size of the Object File:

```
MSG_OBJECT_DATA   (DN)  0x1  1  0 "789f3e22e4b401069008000.........
```

The List properties:

```
MSG_MAPI_PROPS_INFO (DN)  0  0  0  54
```

and the size of the data that follows:

```
MSG_MAPI_PROPS_DATA (DN)  0x1  1  0  "56455253494f4e.........
```

The relationship properties:

```
MSG_MAPI_USER_PROPS_INFO (DN)  0  0  0  54
```

and the size of the data that follows:

```
MSG_MAPI_USER_PROPS_DATA (DN)  0x1  1  0  "56455253494.........
```

It is also worth noting that you can see the RTF bodypart (2130), but no `WINMAIL.DAT` (1734) attachment. From B.05.20, the default action is that all the MAPI properties and TNEF information is held as shown above. The Service Providers no longer produce a separate `WINMAIL.DAT` attachment unless you configure them to do so.

This behaviour can be changed by setting

```
[Mail]
Compatible Messages=1
```

in the `mapi.cfg` file, which we will look at in more detail later.

With this set, the message on the queue would be similar, but you in addition to the above you would also see

```
ITEM_CREATE_DATE  (DN)  99/6/22 15:07.14+60
CONTENT_FILE      (DN)  1734  1734  0x0  1147  0  "WINMAIL.DAT"    \
"WINMAIL.DAT" ""  ""  ""  "IA5"  ""  ""  ""  ""  ""  ""  ""
```

Message Delivery

When the message is delivered and opened by the user, the MAPI Properties must be presented back to the client. Computed properties are generated by local delivery and stored as a container extension file, along with the List and Relationship properties that were mailed with the message.

In a UAL Full Command/Reply trace you will see the reply to a UAL LIST command present back the List, Relationship and Computed properties:

```
REPLY 121       LIST
{
 ..............
ListMAPIProps    = ......
RelMAPIProps     = ......
CompMAPIProps    = ......
 ..............
 }
```

Unfortunately this information is presented as raw data, and if like me you are used to doing a "`tail -f`" on the Full Command/Reply trace this has a tendency to mess up the display due to the embedded control characters.

```
# tail -f OM<OM user ID>U.log | cat -v
```

gets round this.

You might also see the fields

```
MAPIFlags1
MAPIFlags2
MAPIRelFlags
```

populated. These can store 32-bit boolean MAPI properties. Flags1 and 2 can store List or Computed properties.

You will also see the client request the TNEF object file:

```
COMMAND 192     GET_OBJFILE     16:54:48
{
        UARef           = 2704
        SeqNo           = 16
        ListRef         =
        ItemRef         =
        Flags           = 2
        ObjectIndex     = MAPI.ObjectProperties
        TargetFileName  = /var/opt/openmail/user/u000036/TMPA2704/0000301
        TargetFileId    =
```

```
        DirectRef        = 00010603172701c9
}
```

At this point in time there is no easy way of interpreting the data representing the MAPI properties at the server level. In the future we are considering providing some diagnostic utilities to do this.

Notifications and Server Push

What follows here is a cut-down and slightly modified description of the Notification process that appears in *The Essential Mix* OTN, Web ID: 300-0008.

For various historical reasons, there are two daemons that handle notifications. There is the Notification Server (ns) and the Notification Monitor (nfd). Both can be seen using `omstat -a`:

```
root@kuda[tmp] #omstat -a
PC Monitor                  Started              NON-STOP       0
Notification Server         Started              05.28.99       0
Shared memory daemon        Started              NON-STOP
Notification Monitor        Started              NON-STOP
Container Access Monitor    Started              NON-STOP
Item Structure Server       Started              05.28.99
Database Monitor            Started              05.28.99
Licence Monitor Daemon      Started              NON-STOP
LDAP Daemon                 Started              05.28.99
Queue Manager               Started              NON-STOP
Item Delete Daemon          Started              NON-STOP
IMAP Server Daemon          Started              05.28.99
```

As from version B.05.20, the Notification Monitor is a NON-STOP daemon. The notification process is now a key element in supporting clients. Some older clients will still use the notification server for checking if new mail has arrived, but newer clients, including Outlook and the IMAP server, use the Notification Monitor only.

Notifications provide a way for UAL clients to find out that various *events* have occurred, and what they are. There are essentially two categories of event:

1. Notification that some event has occurred
2. New Mail has arrived

First of all, I shall describe briefly how notifications work.

A UAL Client can tell the Notification Monitor (nfd) that it wants to be informed when certain events happen on an object in the Message Store. For example, tell me when someone appends an item to this folder. Indeed, it can register its interest in a list of events on a list of objects.

Registration can be done either using the explicit `ADD_NOTIFICATION` command or using the UAL `LIST` command, which can do an implicit `PREPARE_LIST` of the item and, at the same time, register for notifications. For example, in the listing below, the presence of `PrepListFlags`, `PrepListFilter` and `NfFlags` shows that an implicit `PREPARE_LIST` has been requested:

```
COMMAND 121     LIST    16:54:46
{
        UARef            = 2704
        SeqNo            = 9
        ListRef          =
        ListFlags        = 65544
        StartRecord      = 0
        MaxNumRecords    = 32767
        Fields           = 547406079
        DirectRef        = 0001180416a669f5
        MAPIFields       = 15
        PrepListFlags    = 3096           <---------
        PrepListFilter   = 0              <---------
        NfFlags          = 4              <---------
        NfEvents         = 9992
        NfKey            = 2
        FilterFile       =
```

```
}
```

The nfd keeps track of who has expressed interest in what.

Central to the heart of OpenMail is the *ct* library. All access to the containers in the OpenMail Message Store is via this library. So, when someone makes a change to a container, the ct library looks to see if anyone has registered interest in this container. If so, it sends details of this event across to the nfd. The nfd then looks through its tables and determines exactly who was interested in this event. For each interested user, it writes (appends) the information about this event to file(s) in the `~openmail/notifs` directory.

Note: This directory has changed from `~openmail/temp/safe/notifs`

Each file will be named: `<omuid>.<pid>`

where

`<omuid>` is the OpenMail UID of that interested user

`<pid>` is the PID of that user's `ual.remote` process.

There is one file per user session and all notifications for this user session will be written here.

How does the client get to see that an event has occurred?

Previously (and for the older clients) it was up to the client to *poll* the server every so often and ask if an event had occurred. The way it asked was as follows:

- The client periodically issues a GET_NOTIFICATION to the Notification Server (ns). This will actually send a datagram (udp) to port 5766 (openmailns) asking if the `Notification Occurred` flag is set for this user.
- If the response is Yes, the UAL client library will then talk directly to `ual.remote` and ask it to retrieve from the Notification Monitor (nfd) the details of the notification(s) that have occurred.
- `ual.remote` then reads directly from the `<omuid>.<pid>` file and passes back the info to the client. It reads the whole file and truncates it, so that next time it will not re-read the same info.

With 5.20 there was an addition to this process, known as *Server Push*, whereby the server can actually nudge the client into issuing a GET_NOTIFICATION. This is done via a second file in `~openmail/notifs:`

```
<omuid>.<pid>.pipe
```

A low level datacomm bit is set in the `ual.remote` process. This gets passed to the client either during the current client/server exchange, or as an unsolicited message if there is no current activity.

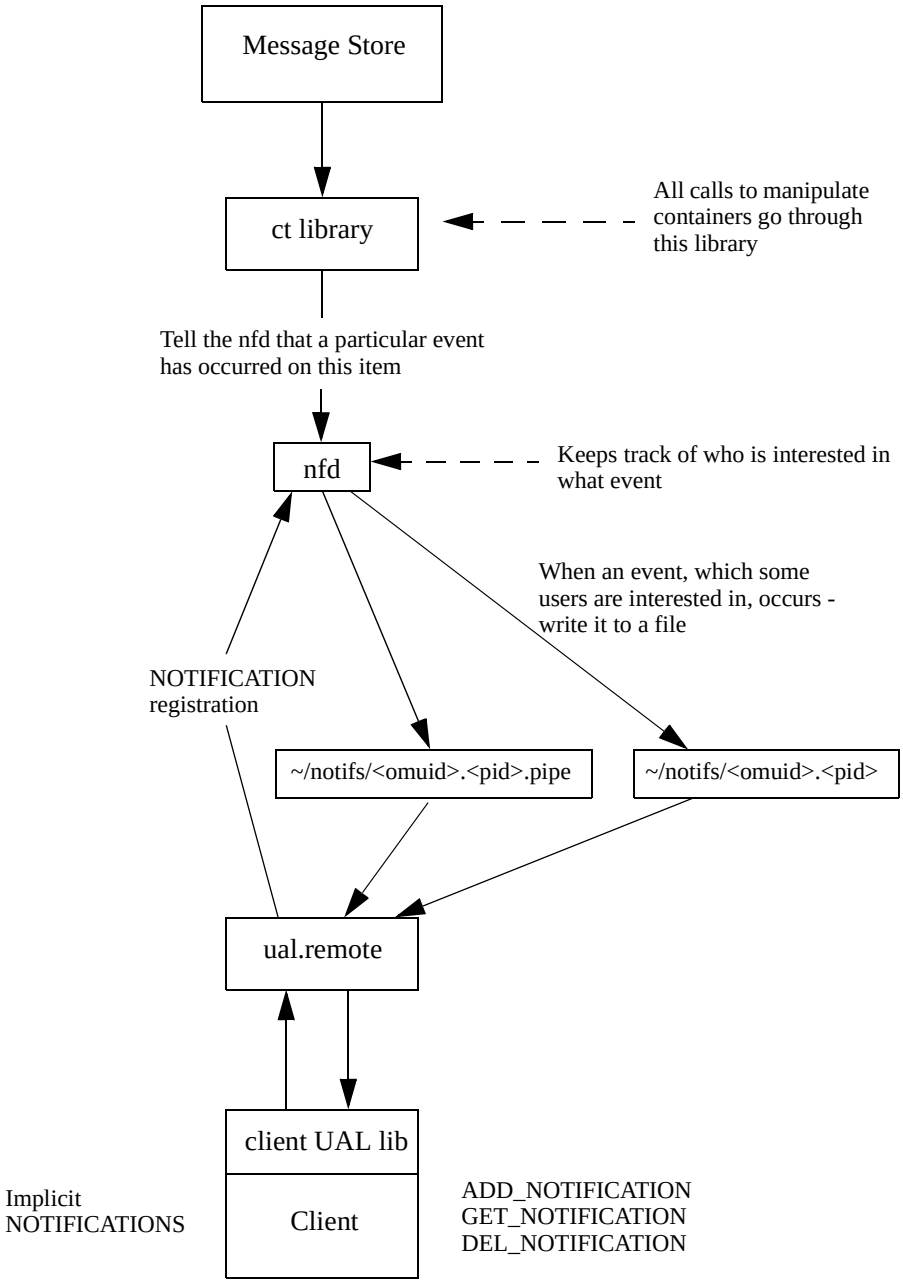
This is the process Outlook uses exclusively; the client does not poll the server or talk to the Notification Server (ns) at all.

If you are using Service Providers that don't support Server Push, there is a tweak to disable it:

```
UAKD_SERVER_PUSH_NOTIFS=TRUE
```

If this teak is set to `FALSE`, the client polls the server to see if there are any new notifications. If the Notification Server (ns) is running, it will handle the client requests for event information. If the Notification Server is not running, the `ual.remote` process will handle the requests for event information.

The notification process looks something like this:



You can see the effect of this behaviour by running Outlook with a default store profile and leave it logged in. Then using another client, for example OMGUI, create a new folder in the filing cabinet. This will appear as another folder in the Outlook display.

There is one other type of notification, called a Server Notification. This type of notification is used when the server needs to inform the client of a server event, such as an impending shutdown of the system. It is issued at a lower level than the other notifications we have been considering.

The table below shows the bit settings for the different types of notifications you might see in the Event field of a UAL trace. Combinations of notifications can be set. As the value shown in a UAL trace is in decimal, I have included this value where appropriate.

Value of Event in GET_NOTIFICATION	**Meaning**
0x00000001	New mail notification. If this bit is set to 1 then a notification is produced if a new message is delivered to the container. Local Delivery places new messages in the Intray only.
0x00000002	New ack notification. If this bit is set to 1 than a notification is produced if a new acknowledgement is delivered to the container. Local Delivery places new acks in the Intray or Pending Tray. Use this notification in one or both places, as required.
0x00000004	New item creation notification. If this bit is set to 1 then a notification is produced if a new item is attached to this container. **NOTE**: this event is obsolete from 5.20 onwards.
0x00000008	Item deletion notification. If this bit is set to 1 then a notification is produced if the item referenced in this command is deleted. If the item is a container, deleting a component at any depth does not trigger this event.
0x00000010 (decimal 16)	Item modification notification. If this bit is set to 1 then a notification is produced if the item referenced in this command is modified. If the item is a container, any modification or deletion of a component at any depth triggers this event. **NOTE**: this event is obsolete from 5.20 onwards.
0x00000020 (decimal 32)	Item copy notification. If this bit is set to 1 then a notification is produced if the item referenced in this command is copied by the ual COPY ITEM command. If this item is a container, copying a component within the container does not trigger this event.
0x00000040 (decimal 64)	Item move notification. If this bit is set to 1 then a notification is produced if the item referenced in this command is moved by the ual MOVE ITEM command. If this item is a container, moving a component within the container does not trigger this event.

Value of Event in GET_NOTIFICATION	Meaning
0x00000080 (decimal 128)	Search complete notification. If this bit is set to 1 then a notification is produced when a search completes. The item identified as ListRef/ItemRef in this command is assumed to be a transaction file containing the actual search request definition.
0x00000100 (decimal 256)	A new item has arrived in the container.
0x00000200 (decimal 512)	A new item has departed from the container, either by move or by deletion.
0x00000400 (decimal 1024)	An attribute of the item has changed.
0x00000800 (decimal 2048)	A new child item has arrived somewhere in the hierarchy below this container.
0x00001000 (decimal 4096)	A new child item has departed from somewhere in the hierarchy below this container, either by move or by deletion.
0x00002000 (decimal 8192)	An attribute has changed on an item somewhere in the hierarchy below this container.
0x00000001	A shut-down has been initiated on the server. This event is not valid on a message store item. It can only be registered on one of the system-level direct references.
0x00000002	A shut-down has been cancelled on the server. This event is valid only on one of the system-level direct references described above.

The following example shows that the event indicated is *a new item has arrived in the container*:

```
COMMAND 352     GET_NOTIFICATION        11:54:03
{
        UARef           = 1
        SeqNo           = 0
        Key             =
        GetNFlags       =
        MaxNumRecs      = 100
}

REPLY 352       GET_NOTIFICATION
{
        UARef           = 1
        SeqNo           = 0
        ReplyFlags      = 1
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        NotifRef        = 7
        Key             = 2
```

```
Event           = 256
DateTime        = 930135243
Status1         = 1
Status2         = -101
TopChildDRef    = 0001140109b0faf0
Flags           = 0
ExtraDRefs      = 0001140109b0faf0/0001180416a669f5//
```

The `TopChildDRef` and `ExtraDRefs` indicate the item affected. Where an item has been created, the `TopChildDRef` gives the Direct Ref of the item. The `ExtraDRefs`, in this case, show the Direct Ref of the item followed by the Direct Ref of the parent item. If, for example, the item was a message in the Intray, the parent item would be the User Folder.

Reminders

Outlook reminders are used for

- Calendar Events
- Tasks
- Flags on messages with a time constraint

The way OpenMail handles Outlook reminders has changed considerably between B.05.20 and B.05.30. The first part of this section describes the way reminders work in B.05.30. For those of you who need to know how they worked at B.05.20, go to the section entitled “Reminders In B.05.20”.

Reminders In B.05.30

At B.05.30 the work of checking if there are reminders to process has been moved from the server to the client, to improve performance. Previously, reminders were handled using the background search mechanism of the server. By default this resulted in every client issuing a persistent search request every 5 mins, even if there were no reminders to return. In addition, the notification of the search completing caused the `ual.remote` process to be woken up. On a server with a large Outlook population, this overhead caused some capacity issues.

What happens now is that on startup, the 4 folders on which it is possible to set reminders on items, are opened and listed. These folders are:

Inbox

Calendar

Tasks

Contacts

If any of these folders contained a large number of items, then the initial client startup may be slower than in previous releases. If for any reason one of these folders cannot be opened, you will see the message:

```
Can not start the reminder service. Unable to show reminders. The object
requested was not found
```

Once the folders have been opened, a list is created of only those items for which a reminder is set. This information is then passed in MAPI properties to the MAPI Service Provider. The Service Provider then registers for notifications against the four folders. This way it can keep the information up to date when a reminder is either removed (deleted or dismissed), added or modified (postponed).

On the MAPI client, there is still a Reminders folder under the root folder. This should not be confused with the Reminders folder on the server, which is where we used to hold search specification file. Now the Reminders information is held in memory on the PC. MAPI keeps track of this and displays the the necessary dialogues at the correct time.

On the server, the Reminders folder still exists under MAPI Root Folders, but it is only there in case the user reverts back to using the B.05.20 Service Providers.

On the client, MAPI still sees a list of items that have reminders set. We have simply changed the way this information is gathered.

Reminders In B.05.20

The OpenMail reminder mechanism at B.05.20 was implemented using the background search service, which had to be running (omon -s search) for reminders to be displayed.

In general terms, background searching works by the client

- creating a search specification file
- registering for a notification against this

The search engine then

- picks up the search request and executes it
- places the results in an object file
- sets the notification to inform the client the search has completed

The client can then either directly examine the object file for the search results, or it can use the UAL LIST command against the search specification file to determine the item(s) that matched the search. For Outlook, the latter method was used.

In the "MAPI Root Folders" area of the users message store under the Filing Area, you find the following (as seen by omcontain):

```
Option?l
Ctr Handle (3003):
     1) MAPI Common Views                          (FOLDER)
     2) Reminders                                  (FOLDER)
Done
```

Opening the reminders folder shows

```
Option?o
Ctr Handle (3003):
Entry Number:2
Ctr Handle = 3004
Done

Option?l
Ctr Handle (3004):
   1) Search Specification                    (Search Specification)
Done
```

This, surprisingly, is the search specification file which is held in transaction file format. Reading this entry tells you the filename for this file:

```
Option?r
Ctr Handle (3004):
Entry Number:1

Item Type: Search Specification
Subject: Search Specification
Subject Charset: IA5
T61Subject:
T61Subject Charset:
Creator: Mapi/Mr///canberra/class
Creator Id: 2223
BB Item Owner Id: 2223
File Name: ~/data/000001f/0004uuq<-------------
.........................
```

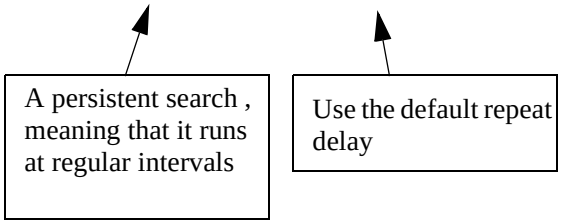
which you can then tfbrowse

```
# tfbrowse -i '~/data/000001f/0004uuq'
 HEADER            (DN)  1000  0  2  1005  0x0  0x0  0  0x0  0x0
 SEARCH_REQUEST    (DN)  0x1  1  1000000  0
 SEARCH_TARGET     (DN)  "000103446533f637"
 SEARCH_TARGET     (DN)  "0001034825a1b1f7"
 SEARCH_TARGET     (DN)  "000118d572169ec0"
 FILTER_START      (DN)  0x0  0x0  0
 FILTER_START      (DN)  0x0  0x0  1
 FILTER_NUM        (DN)  0x0  0x1  29  6  0  0  1
 FILTER_START      (DN)  0x0  0x0  0
```

```
 FILTER_NUM        (DN)  0x0  0x1  29  0  0  0  2
 FILTER_NUM        (DN)  0x0  0x1  29  6  0  0  2
 FILTER_END        (DN)
 FILTER_END        (DN)
 FILTER_END        (DN)
```

The key things to note about this search specification are

```
SEARCH_REQUEST    (DN)  0x1  1  1000000  0
```



The default repeat delay prior to B.05.20 was 10 minutes, at B.05.20 it was 5 minutes (300 seconds). The implication of this is that it is possible for a reminder to trigger up to 5 minutes after the actual time specified on the reminder.

The ~/sys/general.cfg tweaks

```
SE_DEFAULT_DELAY
SE_MAX_OVERDUE_TIME
```

should be reviewed if there are reminder timing problems, check the OpenMail Technical Guide for a description of these.

You can effectively disable the reminder service completely by setting the `SE_DEFAULT_DELAY` general.cfg option to 86400 seconds:

```
SE_DEFAULT_DELAY=86400
```

This will set the persistent search interval to 24 hours.

A search runs against the following Direct Refs of

```
SEARCH_TARGET     (DN)  "000103446533f637"     <----- Calendar
SEARCH_TARGET     (DN)  "0001034825a1b1f7"     <----- Tasks
SEARCH_TARGET     (DN)  "000118d572169ec0"     <----- Intray
```

The implication of this is that reminders will not be triggered on items in other folders. For example if you set a follow-up flag on message in the Intray and then file this in another folder, you will not see the reminder for this message.

The `FILTER_NUM` records detail the search and look more complicated than they actually need to be.

The search itself is looking for either bits 1 or 2 of the `MAPIRelFlags` being set.

An Example

When the user signs on, the client will create a search specification file in the Reminders folder if one does not already exist. If there is already one there then it is left alone.

In order to activate the search the client issues an add notification against the search specification file.

```
COMMAND 350     ADD_NOTIFICATION        11:42:17
 {
        UARef           = 12490
        SeqNo           = 16
        ListRef         =
        ItemRef         =
        DirectRef       = 0001034a4e0ca784    <---- of the Search
```

```
        AddNFlags       = 4                         Specification file
        AddNEvents      = 128
        Key             = 2
 }

 REPLY 350       ADD_NOTIFICATION
 {
        UARef           = 12490
        SeqNo           = 16
        ReplyFlags      = 0
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        NotifRef        = 10                <----- This is the
                                                  Notification Ref for
                                                   this notification
 }                     11:42:17
```

When the search executes and it yields some results, the client is "nudged", via server push to issue the `GET_NOTIFICATION`

```
COMMAND 352     GET_NOTIFICATION        11:52:18
 {
        UARef           = 1
        SeqNo           = 0
        Key             =
        GetNFlags       =
        MaxNumRecs      = 100
 }
```

In the reply we see

```
REPLY 352       GET_NOTIFICATION
 {
        UARef           = 1
        SeqNo           = 0
        ReplyFlags      = 0
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        NotifRef        = 10<----- Notification Ref
        Key             = 2
        Event           = 128               <----- Search Complete!
        DateTime        = 895747938
        Status1         = 0
        Status2         = 1                 <----- Hit count
        TopChildDRef    = 0000000000000000
        Flags           = 0
        ExtraDRefs      =
 }                     11:52:18
```

The client then LISTs the search specification file to obtain the results from the "SearchResults" object file

```
COMMAND 121     LIST    11:52:18
 {
        UARef           = 12490
        SeqNo           = 38
        ListRef         =
        ListFlags       = 65544
        StartRecord     = 0
        MaxNumRecords   = 1000
        Fields          = 816045567
        DirectRef       = 0001034a4e0ca784    <---- Search Spec File
        MAPIFields      = 15
```

```
        PrepListFlags    = 3096
        PrepListFilter   = 33554432
        NfFlags          = 0
        NfEvents         = 0
        NfKey            = 0
 }
```

and in the `LIST` reply we see the item that triggered the reminder, in this case a calendar appointment.

```
REPLY 121       LIST
 {
 .....
 +
        UARef            = 12490
        SeqNo            = 38
        ReplyFlags       = 0
        ErrorNo          = 0
        Err2             = 0
        Err3             = 0
        ListStatus       = -1
        ItemRef          = 1
        Depth            = 1
        Subject          = let's get reminded
        ItemType         = -100
        CreateDate       = 895747420
        AttachDate       = 895747487
        Contents         =
        AbsoluteRef      = 97282
        Flags1           =
        Flags2           = 128
        MsgFlags         = 133231616
        ExpiryDate       =
        RecipId          =
        ItemClass        = IPM.Appointment
        MsgId            = H00008af00017c02
        AckId            =
        Creator          = Mr Mapi
        ReceiptDate      =
        DeferredDate     =
        ReplyByDate      =
        SubjectCharSet   = IA5
        ContentCharSet   =
        AccessCaps       = 47
        ModifyDate       = 895747487
        TotalKBytes      = 1
 .....
        DirectRef        = 000114876f21237e
        ParentDirectRef  = 000103446533f637
 .....
        MAPIFlags1       =
        MAPIFlags2       =
        MAPIRelFlags     = 1
        ListMAPIProps    = .........
```

Note the `MAPIRelFlags` bit setting which is what our search specification was looking for.

With audit logging set on the background search service you can also see this search complete

```
# omconfaud search 4
 omconfaud : Logging level updated OK.
```

```
# tail -10 ~openmail/logs/audit
 search
 time 895747938 Thu May 21 11:52:18 1998 +60
```

```
 completed-request 0001034a4e0ca784<----- Search Spec File
 hits 1         <----- It found one!
```

As this is a persistent search it will continue to run until the client logs off, this can be seen in the audit file

```
search
 time 895751838 Thu May 21 11:57:18 1998 +60
 completed-request 0001034a4e0ca784
 hits 0
```

When the user dismisses the reminder then you will see the `MAPIRelFlags` reset

```
COMMAND 164     MODIFY_ITEM     13:48:06
 {
         UARef            = 12490
         SeqNo            = 63
         ListRef          =
         ItemRef          =
         ModifyFlags      = 16
         Fields           = 8388608
         ItemType         =
         ContentFile      =
         Subject          =
         Flags1           = 0
         Flags2           =
         MsgFlags         =
         CharSet          =
         ItemClass        =
         MsgId            =
         ExpiryDate       =
         DeferredDate     =
         ReplyByDate      =
         AttribTF         =
         OrigFName        =
         ApplicRef        =
         MIMEType         =
         DirectRef        = 000114876f21237e
         Creator          =
         CreateDate       =
         ModifyDate       =
         ExpiryDelay      =
         ListOrder        =
         PreviewText      =
         MAPIFlags1       =
         MAPIFlags2       =
         MAPIRelFlags     = 0                    <----- Set to zero!
         ListMAPIProps    =
         RelMAPIProps     =
         CompMAPIProps    =
         ContentDisp      =
         ContentID        =
         OldModifyDate    =
 }
```

Although this is not supported, you can request some status information from the search engine by "talking" directly down the pipe that the search engine uses.

```
# cd ~openmail/se
 # ll
 total 0
 prw--w--w-   1 root       hpoffice         0 May 21 11:42 pipe.req

# echo "R/tmp/se.stat" >> pipe.req
# cat /tmp/se.stat
```

```
Search Engine:
        Max 20 search processes
        Default delay: 300 seconds
        Max. overdue time: 300 seconds
One-off searches:
Persistent searches:
        DR='0001034a4e0ca784'   child pid=0     last run at 895749140 \
                                                                delay=300
Search processes:
```

This facility should be used with care as it is very easy to abort the SE if you mess it up!

When the client session finishes, the notification for the search is deleted in order to terminate the search

```
COMMAND 351     DEL_NOTIFICATION        14:09:55
{
        UARef           = 12490
        SeqNo           = 81
        NotifRef        = 10                <----- Notification Ref
        DelNFlags       = 0
        DirectRef       =
}

REPLY 351       DEL_NOTIFICATION
{
        UARef           = 12490
        SeqNo           = 81
        ReplyFlags      = 0
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        Total           = 1
}                      14:09:55
```

Acknowledgements

It is possible to send messages with acknowledgements. However they are handled slightly differently than with other clients (for example, OMGUI).

Note: Before B.06.00 version of the OpenMail server, acknowledgements could be received when the recipient was within the OpenMail environment, but not if the original message was sent out through the Internet gateway. However, at B.06.00 acknowledgements are supported by the Internet gateway. Therefore, if you want acknowledgements supported by the Internet gateway, your server will have to have version B.06.00 installed (B.05.10 with PP-March99 is not enough).

To send a message requesting an acknowledgement, click the Options tab when creating the message. At the bottom of this dialogue are the Tracking Options, where you can set delivery and/or read acknowledgements by clicking the appropriate check box.

If you set these options then you can see the ack requested in the audit log (level 11) as the message is processed by the Service Router.

```
routing
time 930565945 Mon Jun 28 11:32:25 1999 +60
type 0 message
priority 0 normal
sensitivity 0 normal
importance 0 normal
created-locally 1
hop-count 1
originator Mr Mapi / canberra, class
subject retrying a read ack
ua-message-id H0000068000006c1.0930565943.kuda.pwd.hp.com
mta-message-id H0000068000006c1.0930565943.kuda.pwd.hp.com
part-size 135
part-type 1166 DISTRIBUTION LIST
part-size 284
part-type 2130 Microsoft RTF
recipient-to Mr Delegate / brisbane, class
ack-req 7 read                                  <----------
queue SMINTFC:openmail@bean.pwd.hp.com
max-nest-depth 0
message-size 3171
part-count 2
delivered-count 1
```

The ack levels are defined as

0 none

1 transmit

2 receive

3 non-delivery

4 delivery

5 auto-forward

6 delete

7 read

8 auto-answer

9 reply

When the message is delivered, the delivery ack is returned as an acknowledgement message.

```
routing
time 930566255 Mon Jun 28 11:37:35 1999 +60
type 110 acknowledgement
priority 2 urgent
sensitivity 0 normal
importance 0 normal
created-locally 0
hop-count 2
recipient-to Mr Mapi / canberra, class
ack-req 0 none
queue LOCAL
ua-ack-id H0000068000006c1.0930565943.kuda.pwd.hp.com
max-nest-depth 0
message-size 334
part-count 0
delivered-count 1
```

How you see this ack depends on the client:

- In AdvMail you would go to the pending tray.
- In OMGUI you see the acknowledgement status against the distribution list of the message in the OutTray.
- With Outlook you see the message in the Inbox with an icon indicating this is an acknowledgement message.

You can configure Outlook to automatically deal with such messages:

Tools -> Options -> E-Mail Options -> Advanced E-Mail Options

You can specify whether delivery and read receipts should be processed on arrival. In the Tracking Options dialogue box you can say whether to delete these receipts after processing. The deletions occur using a background thread, so don't expect them to be removed immediately.

In the Sent Items folder, if you open the message that you sent with the ack, then you should see a Tracking tab which gives the status for each recipient. This tab is only visible once the first ack is returned.

The OpenMail acknowledgement message is reworked into a Receipt Message which is what Outlook expects in order for the above to work. The key MAPI property here is `PR_REPORT_TAG`.

You will notice also that the acknowledgement does not have a subject associated with it, yet when you see the acknowledgement in the Inbox it does have the correct subject. From an OpenMail perspective the mapping between a message and acknowledgements against that message are done using the msg id.

(mta-ack-id in the audit log), `H0000068000006c1.0930565943.kuda.pwd.hp.com` in this case.

In order to be able to map the subject and generate the correct `PR_REPORT_TAG` for Outlook we use the registry to store information about a message that has acks set.

You can see this using REGEDIT and following

```
HKEY_CURRENT_USER\
  Software\
    Hewlett-Packard\
      OpenMail\
        MAPI\
     Default\
       <machine>\
         <user>\
```

```
                Sent Message Details\
            <mta-ack-id>
```

In my example:

<machine>	=	kuda.pwd.hp.com
<user>	=	Mr Mapi
<mta-ack-id>	=	H0000068000006c1.0930565943.kuda.pwd.hp.com

In here you will find values for

Cc

Report Tag

Sent Date

Subject

To

Originally the reasons for implementing this in this fashion were that we needed some way to tie the msg id back to the original subject. In the release B.05.10, with a .PST profile, there was no guarantee that you could see the server store. For this release, we did consider mapping the OpenMail msg id to `PR_REPORT_TAG` but there were some size considerations here which meant this was not possible.

The ack information stored in the registry remains there for 30 days, after which time it is removed by a client process. If the information is not in the registry then the OpenMail ack is still generated, but it arrives with no subject. Opening it you see

```
----------------------------
 Your Message

    To:Unknown
    Subject:
    Sent:01/01/01 00:00

 was read on <actual date>
 ----------------------------
```

i.e. you can see a message you sent generated an acknowledgement. You just cannot see who the message was to, what the message was about and when it was read. Obviously, in this case the Tracking tab can not be updated since there is no `PR_REPORT_TAG` property.

If you see an ack like this then it means the ack information could not be found in the registry. The most likely cause for this will be when the person logs on to the server but from a different/new PC.

Calendaring

Before B.05.20 it was possible to store personal calendar events in the .PST file and to send meeting requests to other users. With the 5.20 release this functionality was dramatically enhanced. Firstly, with OpenMail as the default store, calendaring appointments can be stored on the server. Secondly, Free/Busy time can be stored on the OpenMail server and accessed across the Enterprise.

With the correct permissions, a user can also see the meeting details for appointments shown in the Free/Busy map.

Appointment Storage

What we are providing here is a repository for Outlook to store appointment details. We have no control over what data is stored and the format of that data, that is Microsoft's domain.

Having created some calendaring appointments, let's look in the calendar folder using `omcontain`.

```
Option?s
 3001 (----) User 2223 Folder                    (USER FOLDER)
 3002 (3001) User 2223 Filing Tray               (FILING AREA)
 3003 (3002) Calendar                            (FOLDER)
 Done

 Option?l
 Ctr Handle (3003):
     1) LocalFREEBUSY                            (MESSAGE)
     2) let's get reminded                       (MESSAGE)
     4) Status Review                            (MESSAGE)
     5) Bank Holiday                             (MESSAGE)
 Done

 Option?
```

As you can see each appointment is represented by a separate message. The first message (`LocalFREEBUSY`) contains the local copy of this users Free/Busy time. If you try and open one of these messages

```
Option?o
 Ctr Handle (3003):
 Entry Number:2
 Ctr Handle = 3004
 Done
```

then you can, as it is an OpenMail container....

```
Option?l
 Ctr Handle (3004):
 Done
```

but there is nothing to see. What we have here is an empty container! So where is the information about the appointment?

The answer is this is held as MAPI Properties in the container extension files and the object file (in TNEF format), all we are doing is allowing that information to be stored in the OpenMail message store.

Free/Busy

Free/Busy information provides a "view" of the calendar showing the times when people are available/unavailable for an appointment and, when they are unavailable, what type of appointment they have; tentative, busy, out of office).

In the Calendar folder we have already seen the message "`LocalFREEBUSY`". In this message (again in Microsoft's own format) is the Free/Busy information for this user. When you add a meeting to your calendar your Free/Busy time is updated accordingly.

"`LocalFREEBUSY`" is also used as the trigger for the emboldening the date in the month display that appears when you look at the calendar folder. Where you see dates in bold and no meeting in the calendar, or vice versa, this implies that the calendar and `LocalFREEBUSY` have somehow got out of sync.

In Outlook if you go to

Actions -> Plan a Meeting...

Then you see yourself listed as an attendee and, scrolling across, you will see any appointments represented as time bars and coloured coded depending on the appointment type. Right clicking on a time bar will reveal the meeting details for the appointment. You do not need any special permissions for this, as it is looking at your calendar folder.

Whilst seeing your own Free/Busy information might be interesting, it is actually not much help for scheduling meetings. What is really required is the ability to see other peoples Free/Busy time, so that you can see whether they are available before you send them a meeting request. It is also likely that you will want to see this information for people on other OpenMail servers.

In order to accomplish both of these tasks we need to store the information on the server in such a way that it is accessible to other people. Here we come back to the OpenMail `FREEBUSY` directory which, as we have already seen, we create using the `omaddfb` command in B.05.30 (in B.05.20 it was created by the patch installation process if it didn't already exist).

Publishing Free/Busy

Making your Free/Busy information available on the server is called publishing. Outlook will try and publish Free/Busy information at specified intervals and also when you log off.

If you go to

Tools -> Options -> Calendar Options -> Free/Busy Options

you can see the settings for free/busy information. By default, Outlook will publish 2 months of calendar data every 15 minutes. In the UAL Trace you see this as:

```
COMMAND 326     PUT_FREE_BUSY_TIME      10:18:48
 {
        UARef            = 5273
        SeqNo            = 41
        PutFBTFlags      = 0
        PutFBTFile       =
        PutFBTData       = 2001^]^^2002^].........
 }

 REPLY 326       PUT_FREE_BUSY_TIME
 {
        UARef            = 5273
        SeqNo            = 41
        ReplyFlags       = 0
        ErrorNo          = 0
        Err2             = 0
        Err3             = 0
 }                      10:18:50
```

Outlook only initiates this if there is actually some information that needs publishing.

Where Is The Information Stored, And In What Format?

Let's take a look at an entry in the `FREEBUSY` directory

```
# omsearch -d FREEBUSY -e "s=mapi/g=mr"
 S=Mapi/G=Mr/OU1=canberra/OU2=class/FREEBUSY-0=\001\000\000\000/\
 FREEBUSY-2=\165$u\012\195$u\012\002\000\000\000\220)u\012\250)u\
 \012\002\000\000\000\004>u\012\164Cu\012\002\000\000\0008Fu\012\
 \176Fu\012\002\000\000\000\156Ku\012\186Ku\012\002\000\000\000\
 \000Qu\012ZQu\012\002\000\000\0008su\012Vsu\012\002\000\000\000
```

Free/Busy information is stored in the `FREEBUSY` directory against the directory entry for the particular user. The user's addressing attributes must match the entry in the SYSTEM default directory, even if the attributes are not actually part of the user's actual mailnode. The reason for this is that Free/Busy lookup is driven from the `SYSTEM` default directory and therefore we must ensure the entry in `FREEBUSY` is consistent with this.

The actual Free/Busy data is stored against some new directory attributes, which are newly defined in `~/sys/dir.attribs`

```
# grep -i FREEBUSY ~openmail/sys/dir.attribs
 # Free/busy-time attributes for use in FREEBUSY directory
 2000    MV      OCTET-STRING    0       (FREEBUSY 0)
 2001    MV      OCTET-STRING    0       (FREEBUSY 1)
 2002    MV      OCTET-STRING    0       (FREEBUSY 2)
 2003    MV      OCTET-STRING    0       (FREEBUSY 3)
 2004    MV      OCTET-STRING    0       (FREEBUSY 4)
 2005    MV      OCTET-STRING    0       (FREEBUSY 5)
 2006    MV      OCTET-STRING    0       (FREEBUSY 6)
 2007    MV      OCTET-STRING    0       (FREEBUSY 7)
 2008    MV      OCTET-STRING    0       (FREEBUSY 8)
```

The format of the data is again defined by Microsoft, we simply store this when it is presented from the client. For this reason I would not be tempted into using `ommodent` to try an alter or populate the Free/Busy data, this is Outlook's domain. The only commands you should need are `omsearch` (in order to check an entry is populated) and `omaddent/omdelent` to enable/disable publishing of Free/Busy time.

If we remove a user's entry from the `FREEBUSY` directory, or more likely, if it does not match the `SYSTEM` default entry, that user will not be able to publish their Free/Busy time:

```
# omdelent -d FREEBUSY -e "S=mapi/G=mr"
 [OM 16956] Deleted 1 entry from the Directory
```

The client will display the error

```
"Unable to update public free/busy data"
```

and in the UAL trace you'll see

```
REPLY 326       PUT_FREE_BUSY_TIME
 {
        UARef            = 5452
        SeqNo            = 41
        ReplyFlags       = 0
        ErrorNo          = 6063
        Err2             = 0
        Err3             = 0
 }                      11:12:49

 COMMAND 180     GET_ERRTEXT     11:12:49
 {
        UARef            = 5452
        SeqNo            = 42
        ErrMsgFlags      =
        Errno            = 6063
        Err2             = 0
        Err3             = 0
        Command          =
 }
```

```
REPLY 180       GET_ERRTEXT
{
        UARef           = 5452
        SeqNo           = 42
        ReplyFlags      = 0
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        ErrText1        = No free/busy time entry
        ErrText2        =
        ComDescr        =
}                      11:12:49
```

If you add or change a user entry in FREEBUSY, then we would recommend that you login and create a new appointment and then exit the client to force all the Free/Busy information to be published.

The best way to check whether any Free/Busy information has, at some point, been published is to use

```
omsearch -d FREEBUSY -e "S=.../G.../..."
```

Obviously you canot dig down to individual appointments as Microsoft does not publish the format of this Free/Busy data to enable you to interpret it.

If you want to disable Outlook even attempting to publish Free/Busy data then go to

Tools -> Options --> Calendar Options --> Free/Busy Options

Set the number of months to publish to be 0.

Lookup of Free/Busy Time

Obviously lookup of Free/Busy information for a user queries the FREEBUSY directory containing that user's Free/Busy details. From the client, access to other users' Free/Busy information is done via the Meeting Planner.

The Meeting Planner can be accessed from various places in the client, the simplest being to create a new appointment and then click on the "Meeting Planner" tab.

Alternatively go to the calendar folder and select

Calendar -> Plan a Meeting...

You can enter addresses directly into the attendees column, or you can click on the "Invite Others..." button. Having entered an address a pop-up window should appear stating

```
"Updating Schedule Information"
```

this is attempt to retrieve the Free/Busy information from the server.

If the line returned contains hashed lines then this basically indicates that for some reason Free/Busy information could not be found. On a local system this could be for a number of reasons

- The name does not exist in the system default directory, in which case it does not get underlined).
- The name can not be uniquely resolved, in which case it appears with a red swiggle underline.
- There is no entry for the user in FREEBUSY or there is no data against the entry in FREEBUSY.

For a remote system there is the potential for one further cause and we will look at this in a minute.

If everything is OK, then in the UAL Trace you see

```
COMMAND 325     GET_FREE_BUSY_TIME      16:36:13
```

```
{
        UARef           = 7925
        SeqNo           = 37
        GetFBTFlags     = 0
        UserName        = Mapi^]Mr^]^]^]canberra^]class
        FBTFields       = 2000^^2001^^2002^^2003^^2004^^ \
                          2005^^2006^^2007^^2008
}

REPLY 325       GET_FREE_BUSY_TIME
{
        UARef           = 7925
        SeqNo           = 37
        ReplyFlags      = 0
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        ReplyFlags      = 0
        GetFBTFile      =
        GetFBTData      = 2002^]\245$u^[.....
}                      16:36:13
```

The mnMapFile

Each OpenMail Server with Outlook users has a `FREEBUSY` directory with holds Free/Busy information for the users on mailnodes hosted by that server. We therefore need some way to enable us to be able to ascertain that

Mailnode	**Maps to Host**
canberra,class	kuda.pwd.hp.com

remembering that a single OpenMail server can host a number of local mailnodes. This information is held in the configuration file `~/sys/mnMapFile`, which is created automatically.

Following a local Free/Busy lookup, let's take a look at this file

```
# cat -vt ~openmail/sys/mnMapFile
^]^]^]^]canberra^]class^Ilocalhost
```

The field separator is the <GS> character (0x1d, displayed as ^]), and <Tab> (0x09, displayed as ^I) is used to separate the mailnode from the hostname. (The four <GS> characters at the start of file are to skip over Surname, Given Name, Initial and Generation Qualifier which have no meaning here).

I expect you are now asking yourself how and who configures the information in this file. The answer is that this can be either manual or the system will try and work it out for itself. If an existing entry maps a mailnode to a host, then this is used. If not then the system will try to work it out and then write the entry into the file for future use.

Entries can be altered manually, either directly (e.g. with vi) or using the backend commands; `omaddmnmp, omdelmnmp, ommodmnmp, omshowmnmp, omresetmn.`

You must have system administration (root) capability to use these commands.

For example to view the contents of the file use

```
# omshowmnmp
 canberra,class     localhost
```

Where an entry does not already exist, the system will try and add an entry for a mailnode to the `mnMapFile`. It attempts to do this using the Directory Relay Server and information from the OpenMail (SMINTFC) routes.

You can use the `omresetmn` command to test entries in `mnMapFile` and clear out old, unused entries from `mnMapFile`. The mapping file is gradually repopulated as users attempt to access remote nodes. It is a good idea to run this command whenever a mailnode is added, deleted or moved from one host to another.

See the man page entries for details on how to use these commands and available options.

The Role of Directory Relay Server (DRS)

We introduced the Directory Relay Server (DRS) some time ago as a way to access directories remotely (actually it was originally used to access an X.500 directory on the same server but remote from OpenMail). The same service can be used to query `FREEBUSY` on another server. We will look at cross server lookups in the next section. For now let's just recap on some DRS basics, since it also has a role in updating the `mnMapFile`.

Note: If you manually make changes to the `mnMapFile`, you should restart the DRS (`omoff/omon -s drs`) to ensure the DRS can see the updated host/mailnode information.

You start the DRS with

```
# omon -s drs
```

Three processes are started

```
# ps -ef | grep drs | grep -v grep
 openmail  9091  9089  0 19:07:45 ?         0:00 omdrs
 openmail  9089     1  0 19:07:45 ?         0:00 omdrs
 openmail  9090  9089  0 19:07:45 ?         0:00 omdrs
```

One of these (pid 9089 in the above example) is the parent process that multiplexes requests received on the udp port 5757. The other two are child processes, which do the work. These two child processes are permanent, so that initial requests do not incur the overhead of process startup. If more child processes are required, because of the number of search requests, then these are spawned up to the maximum, which is configured by the `general.cfg` tweak `DRS_MAX_CHILDREN`. Once spawned, additional child processes will hang around waiting for another search request and will terminate after 5 minutes of inactivity.

These basic facts can be seen by running the DRS in standalone mode:

```
# omdrs -s
 Directory Relay Server Configuration Report

 Configured maximum number of child processes: 16
 Configured maximum number of reserved child processes: 2
 Configured child process idle timeout: 300 seconds
 Configured client connection timeout: 3000 seconds

 Effective maximum number of child processes: 16
 Effective maximum number of reserved child processes: 2
 Effective child process idle timeout: 300 seconds
 Effective client connection timeout: 3000 seconds
```

There are two tracing options for the DRS which are invoked through the setting and exporting of the environment variables:

```
OM_DIR_TRACE=1
```

```
TRACE_RDA_PACKETS=1
```

and then restarting the DRS. Setting of these cause the following files to be created

```
~/tmp/dir.<pid>.trc
~/tmp/rda.<pid>.trc
```

Let's look now at the simple example of putting the local host entry into the `mnMapFile`. First remove the existing `mnMapFile`:

```
# cd ~openmail/sys
 # rm mnMapFile
```

Turn off the DRS:

```
# omoff -d0 -s drs
```

and restart it with tracing enabled:

```
# TRACE_RDA_PACKETS=1 OM_DIR_TRACE=1 omon -s drs
 Enabling 1 subsystem(s).
```

Note: Here I am just using a function of the Korn Shell to export the two environment variables to the command "`omon -s drs`"

```
# export TRACE_RDA_PACKETS=1
# export OM_DIR_TRACE=1
# omon -s drs
```

would have had exactly the same effect, except the variables are now exported to the current shell.

There is a `~/tmp/dir.<pid>.trc` file created at this point:

```
# cd ~openmail/tmp
 # ll *.trc
 -rw-rw----   1 openmail   hpoffice      3813 May 26 19:07 dir.9087.trc
```

but it doesn't really tell us anything interesting and the `mnMapFile` has not yet been recreated. However, if we now use Outlook to lookup the Free/Busy time of a local user, then we see the following trace files

```
# ll *.trc
 -rw-rw----   1 openmail   hpoffice      3813 May 26 19:07 dir.9087.trc
 -rw-rw----   1 openmail   hpoffice      3518 May 26 19:11 dir.9090.trc
 -rw-rw-rw-   1 openmail   hpoffice       215 May 26 19:11 rda.9089.trc
 -rw-rw-rw-   1 openmail   hpoffice       694 May 26 19:11 rda.9090.trc
```

Looking through these files, the `omdrs` with the pid 9090 has actually done the work.

Without going into detail, in `dir.9090.trc`, you see it looking for local mailnodes; look for a line saying

```
Filter:          430=Mailnode
```

This is looking at the hidden `USERLIST` directory to find the local mailnodes for this system; equivalent to

```
# omsearch -d userlist -t h -u -e 430=Mailnode
```

The more interesting file is

```
# cat rda.9090.trc
 *** omdrs packet trace START - Tue May 26 19:11:58 1998

 <- = OUTGOING
 -> = INCOMING

 <- COMMAND_ACK
      Packet Size  : 1024
 -> DATA
      Seq. No.     : 1
```

```
      Data Length  : 57
      Command Ref  : 0
      Data         : ....
         Field 1  :<STX>
         Field 2  :101
         Field 3  :0
         Field 4  :
         Field 5  :1
         Field 6  :896206318.1988386457.1106
         Field 7  :====canberra=class
         Field 8  :<ETX>
 <- DATA_ACK
      Seq. No.     : 1
 <- DATA
      Seq. No.     : 1
      Data Length  : 26
      Command Ref  : 0
      Data         : ....
         Field 1  :<STX>
         Field 2  :101
         Field 3  :0
         Field 4  :0
         Field 5  :0
         Field 6  :
         Field 7  :
         Field 8  :
         Field 9  :localhost
         Field 10 :<ETX>
 -> DATA_ACK
      Seq. No.     : 1
```

Here you can see the DRS querying the mailnode (canberra,class) and returning the host (localhost) information.This is the information written to the `mnMapFile`

```
# cat -vt ~openmail/sys/mnMapFile
 ^]^]^]^]canberra^]class^Ilocalhost
```

Further local server Free/Busy lookups will not bother the DRS, unless they refer to a different local mailnode and therefore require another entry in the `mnMapFile`. For users on the same server the DRS plays no further part, since the client UAL connection directly accesses the `FREEBUSY` directory on the local server.

Cross Server Lookup

If we were only ever going to look up Free/Busy information from the local server, then there really would be no need for the `mnMapFile`. In reality we will actually want to look up details for users that reside on other OpenMail servers. Here the DRS provides a dual role:

- It creates the entry in the `mnMapFile`, if none exists.
- It provides the lookup mechanism for remotely querying the information from the `FREEBUSY` directory on the other server.

We have seen how the DRS looks for local mailnodes. For a remote mailnode, the DRS attempts to use the SMINTFC routing information to determine which server hosts the mailnode. For example, I have added the following route from my system

```
# omaddrt -m "brisbane,class" -q SMINTFC -i openmail@bean.pwd.hp.com
```

From this I can see that if I were to send mail to "brisbane,class", then it would go to the machine "bean.pwd.hp.com". If I now attempt to lookup Free/Busy information for a user on this mailnode, then the following happens:

- The local DRS works out this is not a local mailnode and queries its routing table for an SMINTFC route to this mailnode.

- When it finds a route, it attempts to contact the DRS on that machine and asks it which server hosts this mailnode.
- If this second machine is the host, it returns its host name to the querying DRS, which then updates the `mnMapFile`.

On the local machine you see this conversation in the DRS (rda) log file.

```
<- DATA    ** Request sent
      Seq. No.     : 1
      Data Length  : 59
      Command Ref  : 0
      Data         : ....
         Field 1  :<STX>
         Field 2  :101
         Field 3  :0
         Field 4  :
         Field 5  :0
         Field 6  :896262061.126715586.1037
         Field 7  :====brisbane=class
         Field 8  :<ETX>
 -> DATA_ACK
      Seq. No.     : 1
 -> DATA   ** Reply from the
      Seq. No.     : 1   remote DRS
      Data Length  : 36
      Command Ref  : 0
      Data         : ....
         Field 1  :<STX>
         Field 2  :101
         Field 3  :0
         Field 4  :0
         Field 5  :0
         Field 6  :
         Field 7  :
         Field 8  :
         Field 9  :bean.pwd.hp.com
         Field 10 :<ETX>
 <- DATA_ACK
      Seq. No.     : 1
```

This information is then used to update the `mnMapFile`

```
# cat -vt ~openmail/sys/mnMapFile
 ^]^]^]^]canberra^]class^Ilocalhost
 ^]^]^]^]brisbane^]class^Ibean.pwd.hp.com
```

This mechanism also works in a hub/spoke environment. For example, on the local machine I have setup

```
# omaddrt -m "tigger,class" -q SMINTFC -i openmail@bean.pwd.hp.com
```

In this example, the local DRS will contact the DRS on `bean.pwd.hp.com` and ask for the host for the mailnode "`tigger,class`".

The machine, bean, does not host this mailnode but it does have

```
# omaddrt -m "tigger,class" -q SMINTFC -i openmail@tigger.pwd.hp.com
```

so the DRS on bean contacts the DRS on `tigger.pwd.hp.com`, which then returns its official hostname, as it is the host for this mailnode. This information is passed back to DRS that initiated the query

```
<- DATA
      Seq. No.     : 1
      Data Length  : 36
      Command Ref  : 0
```

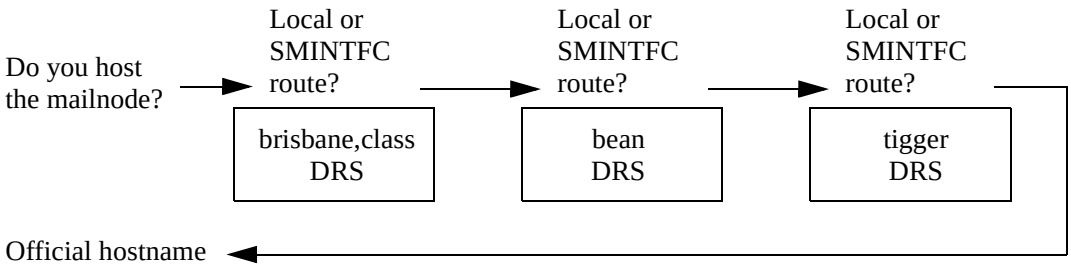
```
      Data         : ....
         Field 1  :<STX>
         Field 2  :101
         Field 3  :0
         Field 4  :0
         Field 5  :0
         Field 6  :
         Field 7  :
         Field 8  :
         Field 9  :hpcca036.pwd.hp.com<-----
         Field 10 :<ETX>
 -> DATA_ACK
      Seq. No.     : 1
```

and the information used to update the `mnMapFile`

```
# cat -vt ~openmail/sys/mnMapFile
 ^]^]^]^]canberra^]class^Ilocalhost
^]^]^]^]brisbane^]class^Ibean.pwd.hp.com
^]^]^]^]tigger^]class^Ihpcca036.pwd.hp.com
```

Note: NOTE: We return the official hostname (`hpcca036.pwd.hp.com`) for the machine, not the alias, `tigger.pwd.hp.com`. Similarly, if the route uses an MX host, we still return the official hostname, because, when querying the DRS for Free/Busy information, we need to make a connection to port 5757 on a physical host.

The following diagram summarises the process:



This process is only used the first time, to populate the `mnMapFile`, and it relies on

- SMINTFC routes between machines.
- the DRS running on the target and intermediate machines in the network.

Where you don't have SMINTFC routes, for example in an X.400 routed network, you will need to manually create the entries in the `mnMapFile`.

Note: For remote Free/Busy lookup to work, the local machine must be able to establish a direct connection of port 5757 on the target machine. If your network does not allow this, then remote Free/Busy lookup will not work.

Let us assume we now have an entry in the `mnMapFile`, which tells us which server hosts a given mailnode. The DRS now performs its second role, which is to query the `FREEBUSY` directory on the remote server hosting the mailnode. The local DRS is not involved in this process.

The user enters the meeting planner and requests the Free/Busy details for a remote user. This causes the following UAL request

```
COMMAND 325     GET_FREE_BUSY_TIME      12:35:38
{
        UARef            = 13565
```

```
        SeqNo           = 38
        GetFBTFlags     = 0
        UserName        = George^]Jonathan^]^]^]brisbane^]class
        FBTFields       = 2000^^2001^^2002^^2003^^2004^^2005^^2006 \
                          ^^2007^^2008
 }
```

As the mailnode is on a remote system, as defined by the `mnMapFile`, then a connection is made to the DRS on the server hosting this mailnode and the `FREEBUSY` directory queried. This can be seen in the DRS trace on the remote machine:

```
*** omdrs packet trace START - Wed May 27 11:50:05 1998

 <- = OUTGOING
 -> = INCOMING

 <- COMMAND_ACK
      Packet Size  : 1024
 -> DATA
      Seq. No.     : 1
      Data Length  : 160
      Command Ref  : 0
      Data         : ....
         Field 1  :<STX>
         Field 2  :100
         Field 3  :0
         Field 4  :
         Field 5  :7
         Field 6  :896266394.527013943.1656
         Field 7  :kuda                    <----Incoming connection
         Field 8  :1                            from this host
         Field 9  :1
         Field 10 :1
         Field 11 :FREEBUSY
         Field 12 :0
         Field 13 :
         Field 14 :
         Field 15 :
         Field 16 :1=George/2=Jonathan/5=brisbane/6=class
         Field 17 :2000/2001/2002/2003/2004/2005/2006/2007/2008
         Field 18 :
         Field 19 :1
         Field 20 :10
         Field 21 :
         Field 22 :
         Field 23 :2
         Field 24 :<ETX>
 <- DATA_ACK
      Seq. No.     : 1
```

reply returned....

```
<- DATA
      Seq. No.     : 1
      Data Length  : 670
      Command Ref  : 0
      Data         : ....
         Field 1  :<STX>
         Field 2  :100
         Field 3  :0
         Field 4  :0
         Field 5  :0
         Field 6  :0
         Field 7  :0
```

```
        Field 8  :
        Field 9  :1
        Field 10 :2000=<ESC>A<ESC>@<ESC>@<ESC>@/2002=@....
        Field 11 :<ETX>
 -> DATA_ACK
      Seq. No.     : 1
```

The FREEBUSY information (Field 10) is then passed back to Outlook via

```
REPLY 325       GET_FREE_BUSY_TIME
 {
        UARef           = 13565
        SeqNo           = 38
        ReplyFlags      = 0
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        ReplyFlags      = 0
        GetFBTFile      =
        GetFBTData      = 2000^]^[A^[@^[@^[@^^2002^]@.....
 }                     12:35:39
```

Note: As this lookup is making use of the DRS on the remote machine, if the DRS is switched off you will not be able to access Free/Busy information for users on this remote server, and hashed lines will appear against the user name to indicate this.

The UAL trace will simply show

```
REPLY 180       GET_ERRTEXT
 {
        UARef           = 13565
        SeqNo           = 48
        ReplyFlags      = 0
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        ErrText1        = No free/busy time entry
        ErrText2        =
        ComDescr        =
 }                     12:49:59
```

Appointment Details

Looking at the Free/Busy map tells you when someone is busy but it doesn't tell you what is actually keeping them busy at that time. Providing you have the appropriate permission on the user's calendar folder, Outlook will display the appointment details if you right click on the time bar.

In order to give another user access to your calendar folder:

- Right click on the folder and select "Properties", then select the "Permissions" tab.
- Click on the "Add" button to select the user from SYSTEM default directory, select the user and press the "To->" button.
- Back on the Permissions tab, highlight the newly added user and assign the required permissions, either using the pre-defined roles or by selecting the combination of check boxes and radio buttons.

For example, setting "Reviewer" permissions causes the following UAL command

```
COMMAND 341     ADD_BBACL_ENTRY 14:06:46
 {
        UARef           = 14088
        SeqNo           = 77
        AddACLFlags     = 32
```

```
ListRef          = 0
ItemRef          = 0
ACLEntry         = Mapi^]Mr^]^]^]canberra^]class
```

user to whom permissions are granted

```
ACLCaps          = 2105344
```

=0x202000

Read permissions *Folder visible*

```
DirectRef        = 00011e403d73694e
```

Direct Ref of the Calendar folder

Having been given the appropriate permission, a right-click on an appointment for this user will pop-up the details of the meeting.

You can easily try out this functionality by using the meeting planner and right clicking on a meeting from your own local Free/Busy map.

Note: If the user has recurring appointments then you may see confusing meeting detail information. This is a known problem.

Viewing of appointment details also works across servers. I have two users:

Mr Mapi/canberra,class on the server kuda.

Jonathan George/brisbane,class on the server bean.

The user "Jonathan George" gives "Mr Mapi" reviewer access to his calendar folder.

"Mr Mapi" wants to schedule a meeting with "Jonathan George" and does a Free/Busy lookup using the meeting planner.

He sees there is an appointment on the 26th May. To view details of the appointment, he right clicks on this appointment.

What is happening here is that the `mnMapFile` is once again used to see which server hosts the mailnode "brisbane,class". A UAL connection is made to this server (bean), attempting the login as

```
COMMAND 102     SIGNON  19:34:32
 {
        UARef            = 14524
        SeqNo            = 2
        SignOnFlags      = 989691904
        UserName         = George^]Jonathan^]^]^]brisbane^]class
        Password         =
        UserID           =
```

```
        NewPassword      =
        Language         =
        ClientType       =
        DesignateName    = Mapi^]Mr^]^]^]canberra^]class
 }
```

The password supplied here is that of "Mr Mapi". In order to determine that we are actually "Mr Mapi", the `mnMapFile` on bean is queried to see which server hosts the mailnode "canberra,class". A callback is then done to `advmail.sckd` on this server (kuda) specifying the username (Mr Mapi) and password that have just been presented. If this checks out, then the UAL connection continues and the calender folder can then be accessed to determine the meeting details. The UAL session to the remote server then remains open in case additional appointment details are requested.

This callback activity can be seen if you enable connection logging for `advmail.sckd` on the server that is requesting the appointment details (kuda, in this example).

To enable connection logging

1. Create the log file

   ```
   # cd ~openmail/tmp
   # touch omsckd.log
   ```

2. Kill the currently running daemon

   ```
   # ps -ef | grep advmail.sckd
   <get the process id of /opt/openmail/bin/advmail.sckd>
   # kill -15 <pid>
   ```

3. Restart the daemon

   ```
   # /opt/openmail/bin/advmail.sckd
   ```

When the appointment details are requested you'll see the following logged

```
# tail -f ~openmail/tmp/omsckd.log
   Sockets client connecting - Wed May 27 19:37:41 1998
   Password check for: Mr Mapi / canberra, class
   Result          : Password matched OK
```

indicating that the callback verification worked OK.

Delegates

With the default store functionality at version B.05.20, it was possible to grant others access to the calendar folder (in an OpenMail default store) and for those users to view details of the appointment through the meeting planner. OpenMail Release 5.30 extends this functionality, providing full delegate access to another user's (the principal's) message store on the OpenMail server.

Delegate access allows a principal to assign permissions to a delegate to access their Outlook folders, enabling a colleague or administrative assistant to help manage their e-mail and schedule.

A delegate has the capability to send messages on behalf of the principal. They may also be given the following capabilities:

- send meeting requests on behalf of the principal
- send task requests on behalf of the principal
- respond to meeting requests on behalf of the principal
- respond to task requests on behalf of the principal
- respond to messages on behalf of the principal
- access the principal's private folders and perform actions on the private folders

A principal has the following capabilities:

- add someone as a delegate
- remove someone as a delegate
- control which private folders a delegate can access and what actions the delegate is allowed to perform on the private folders
- choose to have a delegate receive copies of meeting-related messages sent to them
- choose to have meeting requests and responses sent only to their delegate(s).

To use delegate functionality, both the delegate and principal must be using an OpenMail default store profile.

How to set up a delegate is described in Chapter 7 of the *OpenMail MAPI Service Providers Technical Guide.*

Difference between Designates and Delegates

An Outlook delegate is not the same as an OpenMail designate. The table below lists the fundamental differences between the two.

OpenMail Designates	**Outlook Delegates**
Designates logon as principal, ie using the principal's name and a password setup by the principal.	Delegates logon as themselves using their own profile.
If a designate sends a message, it appears to the recipient that it has been sent by the principal, i.e .not on behalf of the principal.	If the delegate sends a message and puts the principal's name in the From: field, the recipient sees `<Delegate's name> on behalf of <Principal's name>` Similarly, if the delegate replies to a message in the principal's Mailbox, the recipient will see "on behalf of".[1]
Designate and principal must be on same server,	Delegate and principal can be on different servers.
Designate cannot add principal's Mailbox to his own folder hierarchy.	A delegate can add the principal's special folders to his own folder view. The principal can also mail a special folder shortcut to a delegate.

1. This behaviour varies slightly when interoperating with Exchange. See "Delegates In A Mixed Exchange/OpenMail Environment" on page 65.

However, on the OpenMail server, the Outlook delegate feature makes use of some of the designate functionality. For instance, when a delegate is added, a designate file (`000002k`) is created in the principal's 'g' directory.

I have set up two delegates for Mr Mapi; Mr Delegate and Mr Normal. If I now use `tfbrowse` to view the file `~openmail/user/g000038/000002k`, I can see the following entries:

```
root@kuda[g000038] #tfbrowse -i 000002k
HEADER            (DN)  1  0  2  1004  0x0  0x0  0
  ** Warning - Insufficient fields present  (Record No. 1)
DESIGNATE         (DN)  0x2  103  70/1/1 00:00.00  70/1/1 00:00.00  0x0
0x0 0x0  0x0  0x0  0x0  0x1  "Normal/Mr///canberra/class"  ""
  ** Warning - Insufficient fields present  (Record No. 2)
DESIGNATE        (DN)  0x2  1237807525  70/1/1 00:00.00  70/1/1 00:00.00
0x0 0x0  0x0  0x0  0x0  0x0  0x1  "Delegate/Mr///brisbane/class"  ""
  ** Warning - Insufficient fields present  (Record No. 3)
```

In a UAL trace, COMMAND 200 (LIST_DELEGATES) and COMMAND 340 (LIST_BBACL) are used to obtain delegate information and permissions. The UAL trace excerpt below shows these commands being used to find out about our delegates, Mr Delegate and Mr Normal:

```
COMMAND 200     LIST_DESIGNATES 17:04:38
{
        UARef            = 3527
        SeqNo            = 45
        Flags            = 6
}

REPLY 200       LIST_DESIGNATES
{
```

```
        UARef           = 3527
        SeqNo           = 45
        ReplyFlags      = 1
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        DesTotal        = 2
        DesFlags        = 2
        DesName         = Normal^]Mr^]^]^]canberra^]class
        StartTime       = 0
        ExpiryTime      = 0
        IntrayCaps      = 0
        OutTrayCaps     = 0
        TrackingCaps    = 0
        FileCabCaps     = 0
        ListAreaCaps    = 0
        OtherCaps       = 0
        DelegateFlags   = 1
+
        UARef           = 3527
        SeqNo           = 45
        ReplyFlags      = 0
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        DesTotal        = 2
        DesFlags        = 2
        DesName         = Delegate^]Mr^]^]^]brisbane^]class
        StartTime       = 0
        ExpiryTime      = 0
        IntrayCaps      = 0
        OutTrayCaps     = 0
        TrackingCaps    = 0
        FileCabCaps     = 0
        ListAreaCaps    = 0
        OtherCaps       = 0
        DelegateFlags   = 1
}                      17:04:38

COMMAND 340     LIST_BBACL      17:04:38
{
        UARef           = 3527
        SeqNo           = 46
        ListACLFlags    = 12
        ListRef         = 0
        ItemRef         = 0
        DirectRef       = 00010004767fde45
}

REPLY 340       LIST_BBACL
{
        UARef           = 3527
        SeqNo           = 46
        ReplyFlags      = 1
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        ListSize        = 4
        EntryName       = Mapi^]Mr^]^]^]canberra^]class
        EntryCaps       = 16773120
        EntryType       = 100
        ACLFlags        = 5
+...
```

Accessing the Principal's Message Store

When the delegate attempts to open the principal's message store a new UAL session is established. The delegate issues the following UAL signon command

```
COMMAND 102     SIGNON  13:42:51
{
        UARef           = 3470
        SeqNo           = 2
        SignOnFlags     = 989691904
        UserName        = Mapi^]Mr^]^]^]canberra^]class
        Password        =
        UserID          =
        NewPassword     =
        Language        =
        ClientType      =
        DesignateName   = Delegate^]Mr^]^]^]brisbane^]class
}
```

It is the SignOnFlags which indicate this is a delegate signon (0x10000000).

So what is effectively given to the UAL is 2 user names and one password. The principal's name is given as the `UserName` (i.e. the person we want to signon as). The delegate's name is passed in the `DesignateName` field and the password is that of the delegate.

If both users exist on same server, then the UAL simply checks the password against USERLIST. If they exist on different machines, then initially the `~/sys/mnMapFile` is used to identify which server hosts the mailnode of the principal. A UAL connection is then attempted to that machine.

The `ual.remote` process on the target machine looks up the mailnode of the delegate in its own `mnMapFile` and makes a connection back to `advmail.sckd` on the calling machine asking it to verify the name and password of the delegate (without actually signing on).

If this check is successful then the UAL session to the principal's store is established.

This callback activity can be seen if you enable connection logging for `advmail.sckd` on the server on which the delegate resides:

1. Create the log file

   ```
   # cd ~openmail/tmp
   # touch omsckd.log
   ```

2. Kill the currently running daemon

   ```
   # ps -ef | grep advmail.sckd
   ```

   get the process id of `/opt/openmail/bin/advmail.sckd`

   ```
   # kill -15 <pid>
   ```

3. Restart the daemon

   ```
   # /opt/openmail/bin/advmail.sckd
   ```

If you set tail -f on the log file:

```
# tail -f ~openmail/tmp/omsckd.log
```

When the delegate attempts to open the principal's store, you will see:

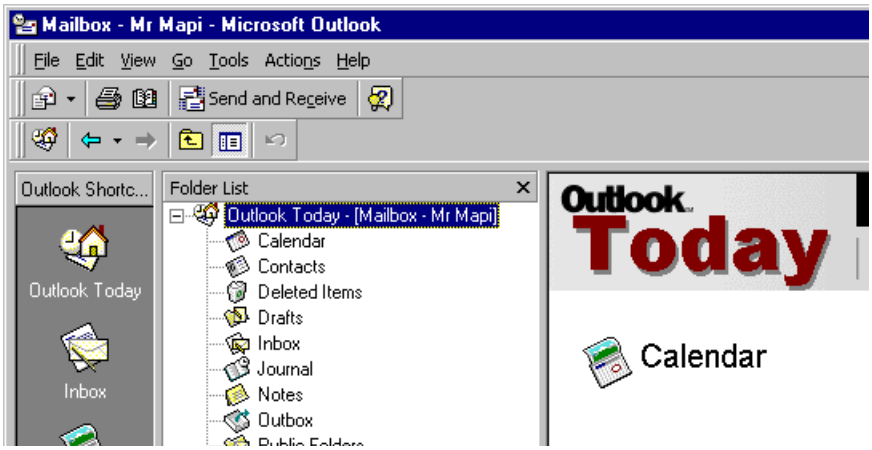
```
Sockets client connecting - Thu Jun 24 11:58:25 1999
Password chk for: Mr Delegate / brisbane, class
Result          : Password matched OK
```

indicating that the callback verification worked OK.

Any server UAL or client side logging settings for the delegate (as defined on the Support tab of the OpenMail Property sheets), will take effect on the session established for the principal. The client side logging will be written to the same `openmail.log` file. If the server logging level is set, additional log files with the OpenMail ID of the principal will be created in `~openmail/tmp`.

In order to add the mailbox of a principal to the delegate profile, the principal must have given the delegate at least reviewer access to their mailbox (i.e. the top folder in the folder list, as shown in Figure 3), in addition to setting delegate permissions from the Delegate tab on the Options window.

Figure 3

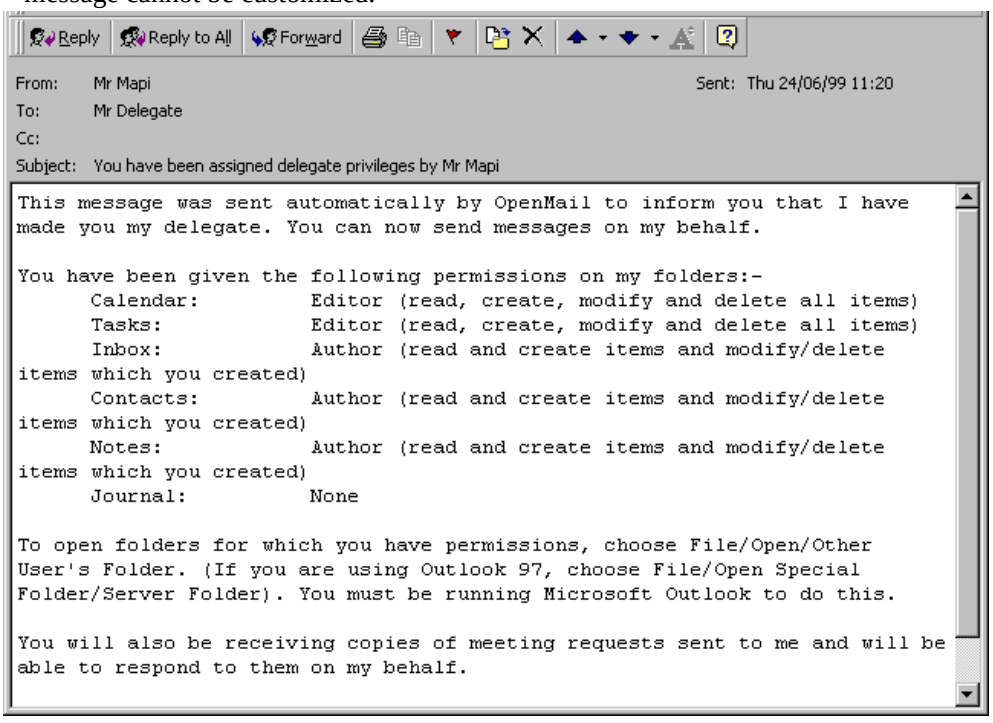


To do this the principal needs to:

- Right click on the mailbox and select Properties.
- Go to the Permissions tab and grant the delegate the appropriate permissions.

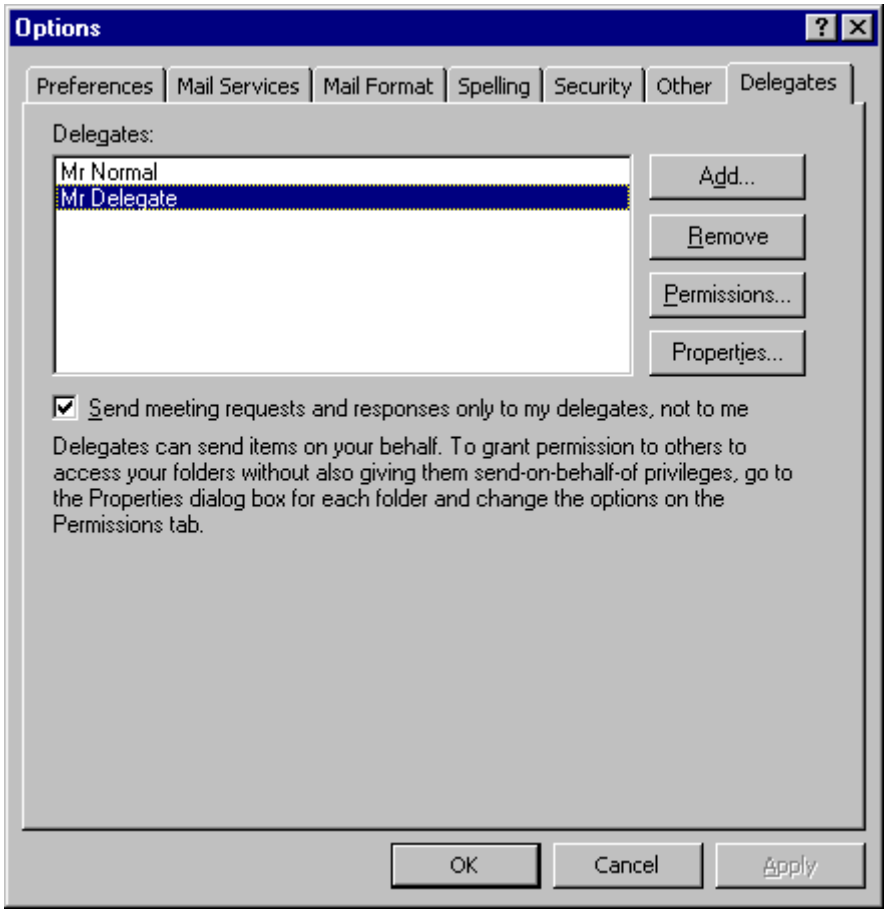
If this has not been done, the delegate will be able to add the store to their profile but not expand the folders.

If the principal checks the box for a message to be sent to the delegate, summarizing the permissions they have been given, a message similar to the one shown below will be sent. This message cannot be customized.



Auto Actions Set Up When Forwarding Meeting Requests

As a principal, you can elect to have a delegate receive copies of meeting requests sent to you. Alternatively, if you do not want to receive meeting requests at all, you can elect for all meeting requests and responses to be auto-forwarded to a delegate. This is done by checking the box shown in the screen below.



Setting up the delegate to receive copies of meeting requests sets up the following in the principal's auto action file:

```
root@kuda[g000038] #tfbrowse -i 000003d
HEADER            (DN)  1  0  2  1002  0x0  0x0  0  0x0  0x0
  ** Warning - Insufficient fields present  (Record No. 1)
AA_NO            (DN)  0x0  451  0  0  0  0  "OpenMail redirected message"
""  "ISO8859_1"  ""  ""
FILTER_START      (DN)  0x0  0x0  0
FILTER_STRING     (DN)  0x0  0x0  14  1  0  0  "IPM.Schedule.Meeting"  ""
"ISO8859_1"
FILTER_END        (DN)
AA_REDIRECT       (DN)  0x0  0x2  0  0  "S=Delegate/G=Mr/OU1=brisbane/
OU2=class"
```

Attribute 14, which is what the `FILTER_STRING` record is looking for identifies the Item Class. If the principal chooses to send meeting requests to the delegate only, then the `AA_REDIRECT` record is set to:

```
AA_REDIRECT       (DN)  0x0  0x2  0  0 \
                          "S=Delegate/G=Mr/OU1=brisbane/OU2=class"
```

The second set of flags indicate:

0x1 - Retain Intray Copy

0x2 - Specifically identifies this is re-direction as a result of Outlook delegation

Some Limitations And Unusual Behavior

The way some aspects of the delegate functionality works may not be as expected. To help identify the areas in question, I have taken the following subsections from the Release Notes for B.05.30.

Viewing Changes Made To Subfolders By Delegates

If another user has logged on to your mailbox, you can see the changes the user is making in real time, because the OpenMail server sends notifications to your copy of Outlook.

However, you will only see changes occurring to messages inside a folder if you have opened the folder. (You open a folder by a single click in the hierarchy pane.)

You will only see changes occurring to subfolders inside a folder if you have expanded the folder. (You expand a folder by a double click in the hierarchy pane, or by a single click on the + symbol to the left of the folder name.)

If you can see a folder, F, in your hierarchy pane, and another user creates a subfolder, S, inside F, you will only see S appear in real time if you have already expanded F. Also, if S is the first subfolder of F, a + symbol (indicating that F has sub-folders) will not appear next to F until F is opened.

Delegates In A Mixed Exchange/OpenMail Environment

There are two scenarios relating to the use of delegates and interoperability with Microsoft Exchange through the OpenMail TNEF internet gateway which cause unexpected behavior.

First, when sending from OpenMail as a delegate on behalf of a principal, the sender name of the message received by the Exchange-based user will exhibit the following behavior:

- There will be no indication that the message was originated ‘on behalf of’ the sender
- The sender name will be in OpenMail format
- The main consequence of this second issue is that such messages will be unreplyable from Exchange. This will be the case for any message (including replies and meeting responses) sent from an OpenMail delegate on behalf of an OpenMail principal user via the OpenMail TNEF internet gateway to an Exchange user.

Second, when sending from an Exchange-based delegate on behalf of an Exchange-based principal to an OpenMail user via the internet, the sender name in the message received by the OpenMail user will not contain any indication that the message was sent by a delegate `on behalf of' the sender. However, the message will be replyable, as the sender name is mapped correctly to SMTP form.

Obtaining Details Of A User’s Private Appointments/Meetings

By default, a delegate will not be able to obtain details (i.e. start-time, end-time and type) of a principal's private Calendar items (i.e. appointments and meetings) by right-clicking on the item in the Meeting Planner.

A delegate will only be able to obtain details of a principal's private Calendar items if the delegate has (at least) Reviewer access to the principal's Calendar folder and if the delegate has been granted access to the principal's sensitive (i.e. private, personal or company confidential) items via the `UAL_SENSITIVE_ITEM_DELEG_ACCESS=TRUE` tweak.

"Folder Visible" Permission On Private Folders

The "Folder Visible" permission on private folders (i.e. any folders in the Mailbox which are not direct/indirect descendants of the Public Folders folder), set via the Permissions property page, is effectively meaningless at this release.

For a private folder to be visible to a delegate in the folder pane (assuming the delegate has added the principal's Mailbox to their user profile), the delegate must have been granted at least "Read items" permission for the parent folder and at least one of the following permissions - "Create items", "Read items", "Create subfolders" and "Folder owner" - for the private folder itself.

Principal's private Calendar items are displayed to a delegate as being non-private: If a delegate has been granted access to a principal's sensitive (i.e. private, personal or company confidential) Calendar items, when the delegate views the principal's Calendar items, any private items will be displayed as being non-private. If the sensitivity of private items were not misreported in this way, Outlook would refuse to display them to the delegate. (Note that, if the delegate opens one of the principal's private Calendar items, the sensitivity is correctly displayed; it is only in the view of the principal's Calendar that the sensitivity is incorrectly displayed.)

The Mailcfg Applet Has Gone

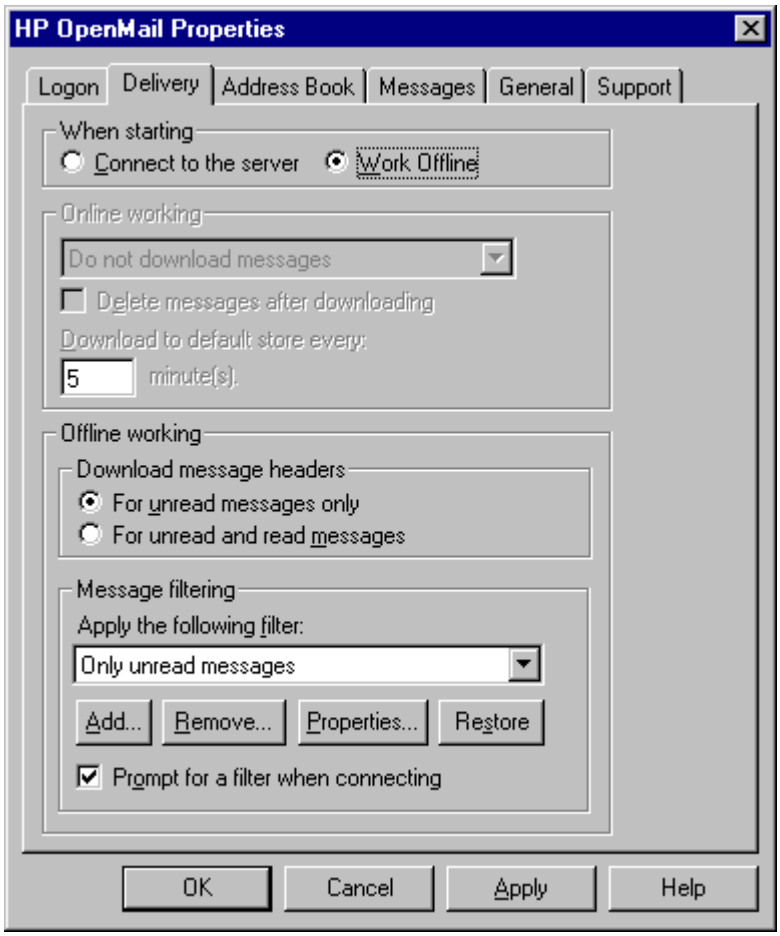
The 16-bit mail configuration applet (mailcfg) was originally released to enable clients to plug in to the designate and auto-action features of OpenMail. It is no longer applicable to the Outlook (MAPI) service providers for OpenMail, because the functionality that it provided can now be accessed from within the Outlook client.

Remote Mail

Remote mail incorporates features to help offsite users manage connections to their OpenMail server and, in B.05.30, customize filters for downloading messages from the server. To use the remote mail features, you must have an “OpenMail without server store” profile. Filters are configured in OpenMail Properties -> Delivery page, which is only visible when OpenMail is not the default store.

Between B.05.20 and B.05.30 there is a change in behaviour when you select a remote mail profile. In B.05.30 you do not see the OpenMail logon dialogue; Outlook will start without you having to enter your username and password. This is because the password on the logon dialogue is used to control access to the server, not to the MAPI Service Providers. The OpenMail logon dialogue is only displayed when you subsequently attempt to connect to the server.

Figure 4 B.05.30 Delivery page



The Delivery page is split into two main sections: Online working and Offline working. Only one of these sections will be enabled, depending on the ‘When starting’ setting. The other group will be greyed out.

The default settings on the Delivery page of a newly created profile depend on the profile type. Below is a summary of the default settings on the Delivery page for each profile type.

Profile Type	**Default Settings**
OpenMail Default Store:	Delivery page not visible.
PST Default Store, with OpenMail store:	When starting, connect to server (Work offline is disabled)
	Do not download messages
PST Default Store, Without OpenMail store:	When starting, connect to server
	Download new messages
	“Delete after download” is checked
	Download to default store every 5 minutes
	Offline working section is disabled

The drop down menu in the Online working section configures how messages are to be downloaded when you are working connected to the server. The options relate to whole messages, not just headers. Remote mail filters are not used at all for online working.

When “Work offline” is selected, there are two ways to connect to the server and download messages:

1. Tools -> Remote Mail -> Connect-> Next (on Remote Connection Wizard) -> “Do only the following:” (invokes the Remote Mail Wizard)
2. Tools -> Send/Send and Receive (Shortcut key: F5). The Outlook 98 Send command causes all messages in your Outbox to be sent, but no messages are retrieved.

Within the Offline working section of the Delivery page, the radio button options in the “Download message headers” section relate to the first method of connecting to the server. This configures which message headers are downloaded by the Remote Mail Wizard. (This setting is completely ignored if you start the session by pressing F5, because F5 does not download headers).

The Message filtering section relates to the second method of connecting to the server and lets you decide which messages are downloaded by a session initiated using Send/Send and Receive or F5. (These settings are completely ignored if the user starts the session using the Remote Mail Wizard, because the Remote Mail Wizard downloads messages whose headers have been marked by the user).

To summarize, then, the filter displayed in the Message filtering field is only effective in a “Work offline” session, when the server connection is initiated using Send/Send and Receive or F5 key. It has no effect on a “Work offline” session when the server connection is initiated using the Remote Mail Wizard; if a user has marked a message header for retrieval, he will receive that message, even if the message does not conform to the filter that is selected in the Message filtering field when the server connection is initiated.

Controlling Which Headers Are Downloaded

The Remote Mail Wizard allows you to download message headers. By default, only headers of messages that you have not yet read are downloaded. However, if you are on the road and want to download a copy of a message you have already read, there are two ways you can do this:

1. In the ‘Download message headers’ section, check ‘for unread and read messages’. Connecting to the server using the Remote Mail Wizard (Tools -> Remote Mail -> Connect-> Next (on Remote Connection Wizard) -> “Do only the following:”) then retrieves all headers.

   Locate the header of the message you want. Right click and choose ‘Mark to retrieve a Copy’.

   Invoke another session by clicking Tools -> Remote Mail -> Connect to retrieve the message.

2. If you know enough about the subject/sender/date-received of the message, you can set up a precise filter in the Message filtering field and retrieve the message in a single session by pressing F5. We will look at creating filters in the following sections.

Headers for which the Server copy no longer exists

It is possible for you to have a message header, but the underlying server message has been deleted (possibly by another user). If you mark to retrieve a copy of such a header, and the underlying message has been ‘soft-deleted’ (so it still exists in your waste basket), then that message will still be downloaded.

If the message cannot be found even in the waste basket, then no message is downloaded (and no error is reported).

Message Filtering

The fields and checkboxes in the “Message filtering” section of the Delivery page allow you to:

- Select a filter from the list of filters installed on your PC (Apply the following filter field)
- Add a new filter (Add button). This displays the filter properties dialogue. The effect of the fields in this dialogue are discussed below.
- Delete a filter (Delete button)
- Modify a filter (Properties button). To modify an existing filter, select the filter and click the Properties button. The filter dialogue opens fully; you do not have to click the Advanced button to see the entire dialogue.
- Restore the pre-defined filters that were shipped with the Service Providers (Restore button)
- Specify whether the system should prompt for a filter at the start of a session. (Prompt for a filter when connecting checkbox). Checking this box causes the system to display the Delivery page at the start of an F5 initiated remote mail session. The dialogue is displayed before connecting to the server. You are only prompted for a filter if you are doing a send and receive (If you select Tools -> Send in Outlook 98, you will not be prompted for a filter).

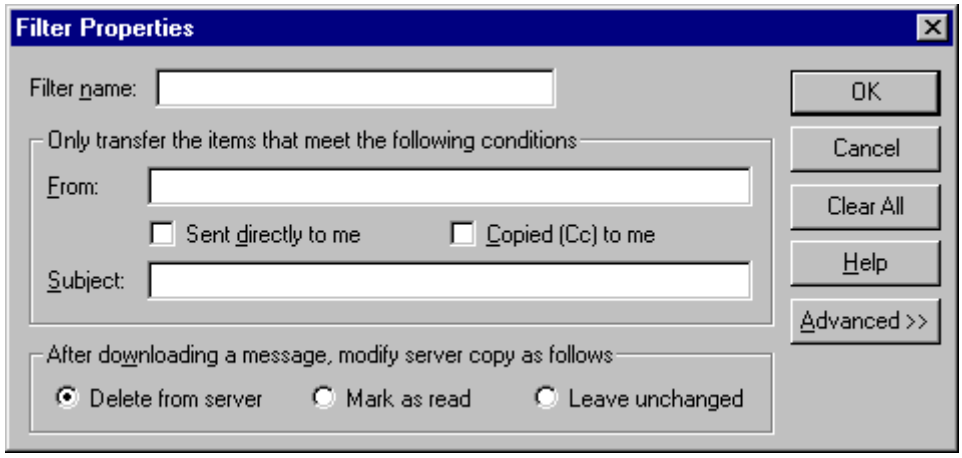
The dropdown list under the “Apply the following filter” field is populated, when the Delivery page is created, with the `*.OMF` (OpenMail Filter) files from the local OpenMail folder (`C:\WINNT\OpenMail` on NT). When the list is populated, no check is made that the `.OMF` files are valid, the list is simply populated with the filenames of any `.OMF` files in the OpenMail folder.

Any `.OMF` files installed on a PC will be visible to all OpenMail MAPI profiles on that machine, and any changes made to a filter using a particular profile will be visible to all profiles.

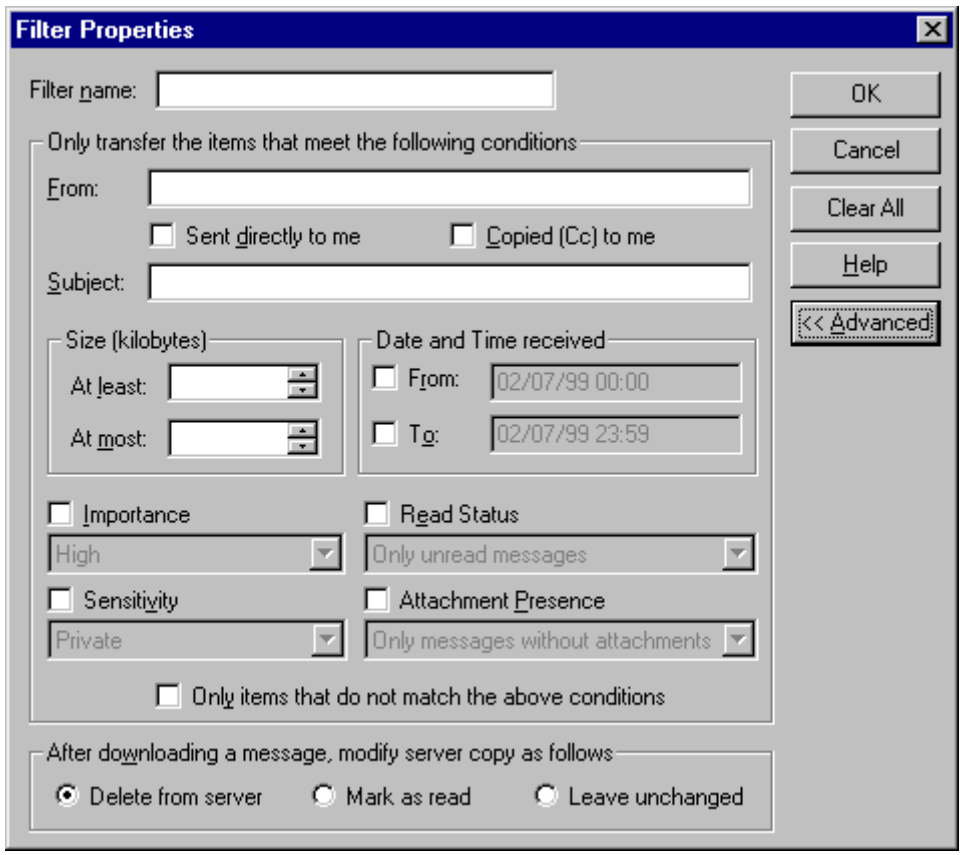
Adding a new Filter

To add a new filter:

- Click the Add button. The filter dialog shown below appears.



This allows you to enter a simple filter based on the fields shown. For more complex filters, you can click the Advanced button to expand the dialog:



(The above dialog shows the default values for a new filter).

Note: All of the fields that you are able to filter on, with the exception of the Read Status, require a B.05.30 compatible server.

Generally, if a field is left blank, then any value for that field is considered to match. Note that if all fields are left blank, no warning is given that no filter has been specified

If multiple fields are specified, a message will only be retrieved if it meets all of the criteria.

The Filter name field

The Filter name must contain a valid Win32 filename when you clicks the OK button. The length of the name is limited to 30 characters.

If an attempt is made to name a filter using a name that has already been used, you are given an opportunity to overwrite it:

The From field

The From field allows you to specify that you only want to receive messages from one or more particular sender(s).

- This field is not case sensitive and there is no limit to the number of characters that can be entered.
- Multiple names can be specified provided each is separated by a semicolon.
- Names typed into this field must conform to the current ‘forename-surname’ setting on the Address Book page. If you OK the dialog, then change the forename-surname ordering, the From field is automatically reformatted to reflect the new forename-surname order. This reformatting takes place even if you re-open the filter without clicking the property sheet’s Apply button after changing the forename-surname order.
- An asterisk can be used as a wildcard character, either as a substitute for a complete attribute (as in WILLIAM *) or as part of an attribute (e.g. TONY BL*).

  A wildcarded attribute will not match an address for which that attribute is empty.
- A single word filter (SMITH) will be interpreted as a surname rather than as a forename.
- Fully-qualified internet (SMTP) names are permitted, but wildcards are not permitted in Internet names. The wildcarded name should instead be written as part of an EFA address string.
- Tag=Value pairs (such as **/O=HP**) can be used to specify a filter on attributes other than those in the personal name or mailnode. However, the explicit tags /G, /O2, /O3 and /O4 cannot currently be used reliably.

The Service Providers auto-format the field when you tab to the next field. For example, excess semicolons are removed.

Currently, no name-check is done on names typed into the from field, because it is quite likely that you will receive email from a sender who is not in any of your address books.

Here are some examples of how names entered will be interpreted:

When Forename-surname ordering is set to forename first:

- DAVE * would match ‘Dave Bound’, ‘Dave Pitt’ etc (regardless of the mailnode’s org units)
- SMITH would match any name with ‘Smith’ as the surname.
- *ON* * would match ‘Jonathan George’.
- RICHARD H* would match ‘Richard Hancock’, ‘Richard Harris’
- * SMITH would match any name with ‘Smith’ as the surname and a non-empty forename.

These are the equivalent filters when ordering is set for SURNAME first:

- *, DAVE
- SMITH (i.e. identical to FORENAME first version)
- *, *ON*
- H*, RICHARD
- SMITH, *

Don’t use the form ‘, DAVE’ since this will be misinterpreted if you subsequently switch to FORENAME first display.

The action of the following filters is unaffected by Forename-surname ordering:

- /PINEWOOD would match any address where ‘pinewood’ was the first org unit.
- /*, PINEWOOD would match any address where ‘pinewood’ was the second org unit.
- /LOCAL; /*,LOCAL; /*,*,LOCAL; /*,*,*,LOCAL would match any name where ‘local’ was any of the org units.
- /O=GOLDSTONE would match any name with this Organisation attribute.
- a.user@here.now would match a user with this internet address as would the following wildcard EFA form (A.USER*)

The Sent directly to me and Copied (Cc) to me fields

You can check either of these fields to retrieve messages that were sent directly or CC’d to you. The system will not allow you to check both.

The Subject field

The Subject field allows you to specify that you only want to receive messages with particular characters in the subject:

- An asterisk can be used as a wildcard character.
- Multiple sub-strings can be specified provided each is separated by a semicolon.
- There is no limit to the number of characters that can be typed into this field.

This field is not case sensitive and is autoformatted when you tab to the next field.

When this field is sent to the server, the Service Providers advise the server of the current client character set.

The Size fields

Limits messages retrieved to those between an upper and lower size limit:

- If At least is left blank, it is assumed to be zero.
- If At most is left blank, it is assumed to be infinite.

During a remote mail session, these sizes are compared against the size of the messages as stored on the server. When a message is downloaded to the PST it usually reduces in size slightly. This may give the appearance that this field is not working correctly, but it is important to remember that the size values are only approximate.

The Date and time received fields

Limits messages retrieved to those received between two dates/times. If the From and To fields have the same value, then any messages received from 30 seconds before this time to 30 seconds after this time will be retrieved:

- When a new filter is created, these values default to today
- The date/time are displayed in a format consistent with the settings in your control panel (Regional Settings). If these settings are changed while the filter dialogue is visible, the formats should change automatically in real time.
- A Date/Time entered is parsed when you tab away from the field and automatically completed if you have not specified enough information. For example, if you enter

  ```
  22/10/98
  ```

  This will complete to

  ```
  22/10/98 12.00AM
  ```

- The Date/Time entered into this field are in local time. They are automatically converted to GMT before being passed to the server in the filter specification.

The Only items that do not match... field

Checking this field causes any messages that do not match the filter to be retrieved.

The After downloading... fields

These fields allow you to specify what should happen to the server's copy of a message after that message has been retrieved.

Filtering And Ack Messages

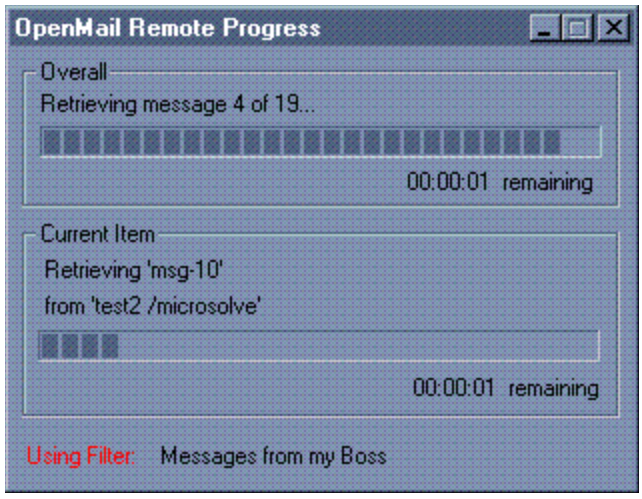
Filters behave slightly differently with acknowledgements (ACKs) than normal messages. Although ACKs appear in your Inbox like a normal message, on the server, they have a different representation; fields such as the Subject are not present. As the server is doing the filtering, this results in the different behaviour. Basically, if the filter includes a field which is not present in the ACK, then no ACKs will be retrieved. The following is a list of fields not present in Acks:

- From
- CC'd to me
- Subject
- Date/Time
- Sensitivity

Importance can be included, provided it is set to Normal. Attachment Presence can be included, provided it is set to Without attachments.

Progress Indicator

At B.05.30, progress indicators let you see what is being downloaded and how long it is liable to take.



There are two gauges in the progress indicator:

- The top gauge (Overall) shows the percent completion of the entire remote mail session and should increment fairly linearly from 0% to 100% during the session.
- The bottom gauge (Current Item) shows the progress for the current item.

A remote mail session consists of a number of phases:

- Connecting to server.
- Examining Inbox.
- Sending.
- Retrieving.
- Deleting.
- Refreshing headers.
- Disconnecting from server.

The information displayed in the progress indicator depends on the current phase. Not all sessions will require each phase.

How the Gauge Works, and Gauge Accuracy

When a remote mail session is started, the remote mail transport provider decides exactly which phases it will have to complete, and which messages/headers it will have to deal with within each phase. With this information, it can estimate how long the entire session will take, which allows it to display an approximate overall-percent complete figure in the Overall gauge. It also allows it to show the correct number of messages in the message which appears above the Overall gauge (e.g.'Sending message x of y')

If the session was started using F5, you are able to create and send new messages after the outgoing flush has begun. In this case, the gauge progress statistics must be (and are) recalculated. You can test this behaviour by:

- Sending a large message over a dial-up line.
- When the message is being sent, create and send another message. The '1 of 1' should change immediately to '1 of 2', and the Overall gauge position should change (as should the estimated time remaining). You can cause another message to be sent by reading one of your inbox messages on which read-acks had been requested.

During the session, several measurements are taken and stored in the profile. An average value is then calculated from these measurements. This means that the system 'adapts' to the speed of a particular setup, but it can take up to 5 sessions to attain full accuracy. Even then, fluctuations in the performance of the server and modem can make the gauge seem inaccurate.

The important point to remember is the remote progress indicator is only designed to give an *estimate* of the progress.

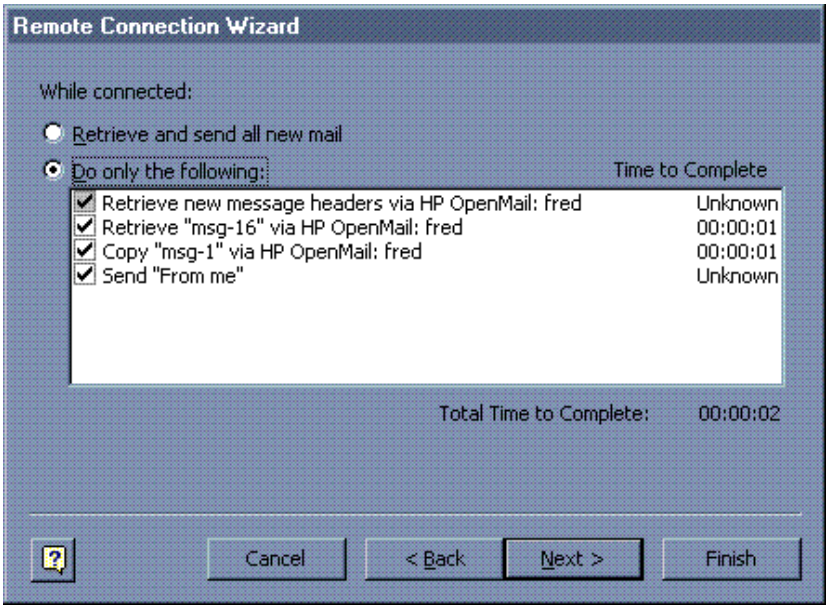
The Time Remaining Indicators

The time-remaining indicators display a '?' symbol if the time remaining cannot be computed with sufficient accuracy. In general, this symbol will be displayed for approximately the first 20% of gauge movement.

Estimated Retrieval Time Display

This display relates to the estimated download times for messages being retrieved using the Remote Mail Wizard.

Tools->Remote Mail->Connect then displays the Remote Mail Wizard, and page 2 of the wizard shows the estimated time to retrieve the marked messages:



How the Estimated Retrieval Time is Calculated

Whenever the transport provider sends/retrieves a message, it measures the time taken and divides it into the message size to obtain a figure for throughput in bytes per second. It stores the last 5 values for throughput in the profile. (Because an average is taken of the last 5 throughput figures, it can take up to 5 sessions to attain full accuracy).

When a message header is downloaded, it uses the average of the last 5 throughput values to compute the estimated retrieval time for the message represented by that header and writes this information into the message header's PR_MESSAGE_DOWNLOAD_TIME property.

The message header is then stored as a row in the PST, and used by the Inbox view and the Remote Mail Wizard when necessary.

It is important to realise that the value stored is the value that was calculated at the time the header was downloaded, and is based on the average throughput values that were available at that time. If you upgrade to a faster modem, the estimated retrieval times are not updated.

The original goal for the accuracy of the Estimated Retrieval Time was that it should be able to distinguish messages that would take seconds to download from those that would take minutes from those that would take hours. It is very difficult to compute accurate estimates for the download of small messages because the server 'lies' about the true size of small messages (The size reported by the server includes server overhead data that is not downloaded with the message). Therefore, if you has 50 small messages to download, the total time computed by Outlook could be inaccurate.

Some points to remember:

- Several days could elapse before you actually decide to download the message, by which time factors could have changed. For example, the server might be under particularly heavy load, or you may have upgraded your modem, or you may be downloading a message and using the dial-up line for something else at the same time.
- Newly installed Service Providers make assumptions about the performance of the server and the speed of the dialup line (a 28K modem is assumed initially). If you have a 14k4 modem and are running against a much slower server, it is possible that the times could be out by a factor of 2. The system does learn the correct figure for bytes/sec throughput, but it can take up to 5 message downloads to achieve full accuracy.

- Most modern modems support data compression. So, if the server says that a message is 100Kb, a modem may compress this into 25Kb, so the message will download in a quarter of the expected time. Eventually, this data compression factor will be absorbed into the estimation and accuracy will be restored. However, not all messages are equally compressible. If a user subsequently downloads an uncompressible 100Kb message after the system has adapted to compressible messages, then it could potentially take 4 times longer than expected to download the message.

To summarize, then, the time to download a particular set of messages will depend on:

- How many times you has used remote mail before.
- The types and compressibilities of messages that you has downloaded before.
- The performance of the server at the time of the download.
- The performance of the modem/dial-up line at the time of download.
- The compressibility of the individual messages.

When Can You Cancel A Remote Mail Session?

There are several ways in which you can cancel a remote mail session:

- If ‘Prompt for a filter when connecting’ is checked, the Delivery page is displayed after you has pressed F5. You can click Cancel at this point to cancel the entire session.
- If you have specified a filter to use, then pressed F5, the Delivery page is displayed if the transport detects a problem with the filter (if, for example, it is damaged, or has been deleted from disk). You can click Cancel at this point to cancel the entire session.
- The Delivery page is also displayed if an incorrect password has been entered, or ‘Use stored password’ is not checked. You can click Cancel at this point to cancel the entire session
- You can click the Cancel button in the ‘Checking for New Mail’ dialogue at any time during the remote mail session, but if you click it while a large message is being sent or retrieved, the Service Providers will finish uploading/downloading the message before the cancel takes effect. (and a message is displayed in the progress indicator saying that a cancel is in progress). Note that if the Cancel button is clicked during the session, some operations will not be done. For example, if you used ‘Mark to retrieve’ on a message header, and click cancel while the message is being downloaded, the server copy of the message will not be deleted.

Using The Internet Mail Gateway With Outlook

The supported method for communicating between OpenMail and Exchange is using SMTP, i.e. the Internet Mail (Unix) gateway. At B.05.20 we introduced the TNEF "flavour" of the Unix gateway. At B.05.30 this has been extended to enable message and meeting request exchange between the two environments.

Transport Neutral Encapsulation Format (TNEF)

If I say "winmail.dat" or TNEF or filetype 1734, what does that mean to you? To most people it's a real pain, particularly if they regularly exchange mail with people using non-Microsoft clients.

You even see people including in their e-mail signature, text to the effect

```
------------------------------------------------------------
I am currently testing an e-mail client called Outlook 98.
If present, please disregard the "winmail.dat" attachment.
------------------------------------------------------------
```

as they got fed up with people asking them what "winmail.dat" is and how they should read it!

To a Microsoft (MAPI) client this contains additional message properties, which don't really effect the content, but could effect the display and message presentation.

TNEF stands for Transport Neutral Encapsulation Format, it was designed to promote interoperability between messaging systems that support different sets of MAPI features. The format encapsulates MAPI properties into a binary stream that can be transported by a transport provider. On receipt of such a message the receiving transport provider can decode the binary stream so making all the properties of the original message available to the client.

Note: Some messages from Exchange embed any message attachments in the TNEF. At B.05.30, we have enhanced the Internet Gateway, so that it can extract such attachments from the TNEF and put them as attachments to the OpenMail message.

From B.05.20 we carry the MAPI properties in "blobs" (container extension files, object files and transaction file records), hence we no longer need a "winmail.dat" attachment. However, older MAPI clients (for example someone using the B.05.10 Service Providers) would still like to see this attachment.

This is why we have the `mapi.cfg` option

```
[Mail]
 Compatible Messages=1
```

With this set, the B.05.20 Service Providers create both "blobs" and the "winmail.dat" attachment. By default, no "winmail.dat" is produced, you can see this in the audit logs (level 11) when the message hits the Service Router.

```
part-size 131
part-type 1166 DISTRIBUTION LIST
part-size 182
part-type 2130 Microsoft RTF
```

with "Compatible Messages=1" then you'd see

```
part-size 131
part-type 1166 DISTRIBUTION LIST
part-size 182
part-type 2130 Microsoft RTF
part-size 1284
part-type 1734 Microsoft Mail Message Data        <----- WINMAIL.DAT!
```

While we generally want to remove "winmail.dat", there maybe times when we want to retain these MAPI properties.

For messages going between OpenMail systems using the `SMINTFC` or `OMX400` queues, this is not an issue; the MAPI properties are held in the transaction file records and object file, which become part of the serialiased/encoded message that is tunnelled through these transports. However, what about where we interoperate with another Microsoft client environment (for example, Exchange) via the Internet Mail Gateway. In this case, we need to take the MAPI property information from the blob and re-build a "winmail.dat" TNEF file.

MIME/TNEF Routes

Most Internet clients don't understand and don't want to see "winmail.dat", but if we are sending to Exchange via the Internet Mail Gateway, we do want to create a "winmail.dat". We have therefore created a new type of Internet Mail Gateway route .... the TNEF route!

Messages arriving in OpenMail which have a "winmail.dat" attachment will be stamped with the mailnodes specified by

```
# omconfux -t <mailnode>
```

Outgoing messages routed to the TNEF route, as specified by

```
# omaddrt -m "..." -q unix -i tnef
```

will have a "winmail.dat" created from the blob information.

Note: `omconfux -t` automatically adds the TNEF route.

For example

```
# omconfux -t "unix,tnef"
 omconfux : OpenMail config file was updated successfully.

#omshowux
 MIME Mailnode     : unix,mime
 UUENCODE Mailnode : unix,uu
 SHAR Mailnode     : unix,shar

 TNEF Mailnode     : unix,tnef                  <-----

Dist List Abbreviation Limit : 0
X.400 Address Format Option : 1
X.400 Address Character Encoding Option : 1
Primary Address Delimiter : /

# omshowrt -m "unix,tnef"
 Queue: UNIX

Route: unix,tnef
Routing Information: TNEF
```

For backward compatibility reasons the default MIME gateway will still generate (from the blob), or retain the "winmail.dat" attachment.

Once specific TNEF routes have been setup, the default MIME gateway should be configured not to produce or retain "winmail.dat", since most Internet users don't want it. This is done using a tweak in `~openmail/sys/general.cfg`:

```
UX_PRE_5_20_COMPATIBILITY_MODE=FALSE
```

Once set, a message sent via the MIME gateway will not have a "winmail.dat" attachment. Incoming messages will have the "winmail.dat" information encoded into the blob and the "winmail.dat" attachment removed.

Further Exchange Interoperability Enhancements In B.05.30

With B.05.20, there is no effective way to differentiate between an SMTP address of an Internet user and the SMTP address of an Exchange User. From B.05.30, when using an SMTP address, if you look at the properties of this address you will see a check box labelled:

`Always send to this recipient in Microsoft Outlook rich text format`

If this box is checked then the message will be routed via the TNEF variant of the Internet gateway. If there is no TNEF gateway configured, or the checkbox is left clear, then the MIME gateway will be used to route the message.

To achieve the best message fidelity between OpenMail and Exchange, this checkbox should be checked.

Note: If the MIME gateway is used meeting requests/responses appear to the recipient as plain messages and therefore cannot be actioned properly.

As part of the improvements made to better enable interworking with Exchange, we have introduced some new `general.cfg` tweaks:

```
UAL_DEF_TNEF_MN_OVERRIDE
UAL_DEF_TNEF_MN_OVERRIDE_CS
UAL_DEF_MIME_MN_OVERRIDE
UAL_DEF_MIME_MN_OVERRIDE_CS
```

These tweaks are intended to allow replies to messages from the Internet to be routed via the correct type of gateway. The UAL command CHECK_NAME will use the mailnode specified in the relevant tweak to resolve any SMTP address given to it.

The _CS tweaks are used to specify the charset of the mailnode in the other tweaks. The combined effect of these tweaks is to allow replying to internet messages from servers which do not have a local internet gateway route configured.

```
UXI_PASSIVE_RECIPS_MAPI_ENABLED
```

Exchange gives us no information about the RTF-awareness of passive message recipients. Consequently, we need to make assumptions for the purposes of “reply to all” from OpenMail. If true, this tweak causes all passive recipients to be treated as RTF-aware in a reply to all from OpenMail via the Internet gateway.

```
UXI_PRESERVE_MAPI_MSG_CLASS
UXO_PRESERVE_MAPI_MSG_CLASS
```

If true, these preserve the MAPI message class of any messages being mapped via the TNEF gateway. The Exchange IMC maps the MAPI message class for backwards compatability with such products as Schedule+ and MSMail. Unfortunately, this mapping confuses the OpenMail MAPI Service Provider, so the mapping of message classes back to their Exchange originals is needed to allow the Service Providers to recognise messages correctly. The mappings are configured in the `~openmail/sys/mapiclass.map` file.

We have also added a new `mapi.cfg` setting

```
[Addressing]
 InternetToOM=1
```

Which causes internet addresses to be displayed as OpenMail addresses (with the relevant RFC-822 type DDAs). This allows for deterministic routing of replies (ie. not all via the default TNEF/MIME route or override).

Steering File

The TNEF route has its own steering file, which is

`~/sys/tnefout.str`

this is based on `~/sys/mimeout.str` (the MIME gateway steering file), which is used if `tnefout.str` is not present.

The reason for having a separate steering file is that you might want to handle certain conversions differently depending on whether you are going to another MAPI type environment or to the native Internet. An example of where you might require different behaviour would be in the handling of RTF. See the section entitled “Supporting Communications to Non-MAPI Clients” in the *OpenMail MAPI Service Providers Technical Guide*.

MIME Content-Types

One other file, worth a brief mention is

`~/sys/mime.types`

this is the file which maps OpenMail filecodes to MIME Content-Types.

The two entries that are worth noting for MAPI are

```
2130    application/rtf
1734    application/ms-tnef
```

The second of these is important as Exchange is looking for this content type to identify a "winmail.dat" (TNEF) attachment. Without this entry "winmail.dat" would be given the following content type

`Content-Type:application/x-openmail-1734`

which Exchange wouldn’t know what to do with!

Recommended Settings To Include/Exclude Winmail.dat

With a setting in mapi.cfg to determine whether the client creates "winmail.dat"

```
[Mail]
 Compatible Messages=
```

and another to determine the behaviour of the MIME gateway in general.cfg:

`UX_PRE_5_20_COMPATIBILITY_MODE=`

and a special TNEF route available....I can see a number of people thinking long and hard about how to use these options in conjunction with each other.

By default, when you install B.05.20 or later (server and service provider):

- clients no longer generate "winmail.dat".
- MAPI properties are stored in the blob.
- the MIME gateway will generate "winmail.dat" outgoing.
- the MIME gateway will leave "winmail.dat" on an incoming message.

Settings for mixed B.05.10/B.05.20/Exchange environment

- `Compatible Messages=1`
- No general.cfg setting

This maintains the status quo with "winmail.dat" still created by the client and maintained at the MIME gateway.

Once B.05.20 (or later) is fully implemented (both client and server):

- Remove Compatible Messages setting.
- UX_PRE_5_20_COMPATIBILITY_MODE=FALSE
- Use the TNEF rout to interoperate with Exchange.

- Client stops creating "winmail.dat". MIME gateway removes "winmail.dat" both incoming and outgoing.
- Routes specifically requiring "winmail.dat" use TNEF routes.

These recommendations are to try and accommodate the conflicting requirements to have/or not have a "winmail.dat" attachment depending on whether you’re a Microsoft or non-Microsoft client.

If you really don’t want "winmail.dat" and are prepared to lose the information it provides then from day one I would suggest

- No Compatible Message setting
- UX_PRE_5_20_COMPATIBILITY_MODE=FALSE

In addition, as from Periodic Patch 2 released in March 98 there is the sledgehammer tweak in general.cfg

```
SR_FILTER_TYPES_OF_ATT=TRUE
```

which causes the Service Router to remove any file of type 1734 (i.e. "winmail.dat") that it sees.

mapi.cfg

The file mapi.cfg actually resides on the server, but is effective only when downloaded to the PC.

The first non-commented line in the file is the version number, if the number of the server version is greater than that of the version on the PC then the file is downloaded to the PC.

You do not need to have a mapi.cfg for Outlook to work with an OpenMail server. Its purpose is to provide a customization mechanism for altering some client behaviour.

We do not ship a default mapi.cfg file, if you want this file then you must create it. Here is an example mapi.cfg file

```
#
 # MAPI Client configuration file:
 #
 # Use this file to set options for all OpenMail MAPI Client users
 #
 # The first non-commented line should be a version number
 # Increment this number to force changes to be downloaded
 # to users' workstations.
 #
 1
 # The [Directories] section
 #
 # Use this section if you want to include additional directories from the
 server
 # The server default directory is always opened by the MAPI address book
 # provider.
 # The entries have the form
 #       N = directory name
 # where N is a consecutive sequence number (1-20)
 #
 # Directory names are case-sensitive.
 #
 # Example showing 2 additional directories:
 #
 #   [Directories]
 #   1 = SALES
 #   2=OVERSEAS
 #
 #
 # The [Name Attributes] section
 #
 # Use this section if you want to include additional attributes from
 # the OpenMail directory as a name/address "properties" page and
 # the "search" page of the MAPI client.
 #
 # The settings are:
 #
 #     heading = text
 #
 # where "text" is used as the page Tab heading (maximum 16 characters)
 # followed by up to 8 customized attributes of the form:
 #
 #     N = label, tag
 #
 # where N is the sequence number (1-8), "label" is the label text
```

```
# displayed
# for the attribute on the page (maximum 24 characters), and tag is the
# numeric tag for the corresponding OpenMail directory attribute.
# Run "omshowatt -u" to see the list of available tags.
#
# Example showing 3 attributes:
#
# [Name Attributes]
# heading = Custom
# 1 = Job Title:, 111
# 2 = Department:, 115
# 3 = Phone:, 116
```

For a full list of possible settings you should refer to the *OpenMail MAPI Service Providers Technical Guide*.

It is possible to have one mapi.cfg file per OpenMail language installed on the system, the client downloads the file from

`~openmail/nls/$LANG`

where $LANG is the language specific directory.

I'm using the default language (C) so I create

`~openmail/nls/C/mapi.cfg`

When I logon to the server, I see in the UAL Trace

```
COMMAND 171     GET_FILE        23:34:59
 {
        UARef           = 5319
        SeqNo           = 4
        GetFlags        = 7
        FromFileId      =
        FromFileNo      =
        ToFileName      =
        ToFileId        =
        ToFileExt       =
        CharSet         =
        AALogStart      =
        DirHandle       =
        FileNameBase    = mapi.cfg
 }
 REPLY 171       GET_FILE
 {
        UARef           = 5319
        SeqNo           = 4
        ReplyFlags      = 0
        ErrorNo         = 0
        Err2            = 0
        Err3            = 0
        Filename        = /var/opt/openmail/nls/C/mapi.cfg
        VersionNo       = 0
        ModDate         = 0
        FileSize        = 1545
 }                      23:34:59
```

On the PC, I can then see the file under

C:\WINNT\openmail\mapi.cfg

It is possible to locally change mapi.cfg on the PC, however if the server version number changes then this will be downloaded and overwrite the local copy.

Some Common Settings

We've already seen the use of one mapi.cfg option

```
[Mail]
 Compatible Messages=1
```

Let's look at some others

```
[Display]
 ShowMailnodes=1
```

This displays OpenMail addresses with their full mailnode information. It is particularly useful when selecting similar entries from the directory as without this setting you tend to have to scroll across to see the mailnode. The setting also applies to the display of addresses when either composing or reading a message.

A useful tip on a an unresolved address is to right-click, this shows possible alternatives, which with setting include the mailnode.

A similar setting in the [Display] section is

```
ShowCustomAttributes=1
```

this should allow custom attributes from the directory to be displayed with addresses.

This setting works in conjunction with

```
UserDefinedAttributes=%(attr)
```

There is currently a problem with using attibutes outside of the standard addressing attributes, for example

```
UserDefinedAttributes=%(1)%(5)
```

will, correctly, only show the Surname and OU1 fields.

However,

```
UserDefinedAttributes=%(1)%(116)
```

which is trying to display the phone number from PHONE-1 will only display the Surname.

If you are going to use this feature then I would recommend you specify enough attributes to make the addresses look meaningful. For example, just using

```
UserDefinedAttributes=%(16)%(17)
```

will only display DDT1/DDV1 values, which some addresses in the directory may not have.

The setting

```
LineLength=xx
```

is only effective where the client is configured to send plain text messages rather than RTF.

The RTF/Plain Text setting can be found on the "Messages" tab of the OpenMail Service properties.

If you receive a lot of Internet mail, then the setting

```
ShowCompleteInternetAddress=1
```

tidies up the display of Internet type addresses.

In the section

```
[Directories]
```

You can list additional OpenMail directories that you wish to be able to view from the client. Make sure these entries are case sensitive, for example if you have the directory

```
MYOWNONE         Shared       LOCAL DB     config update read modifyself
```

(as shown by omlistdirs)

Then the entry in this section needs to read

```
1=MYOWNONE
```

You also need to add these directories to the CDA server (omaddcda) and then run "omexeccda -d <directoryname>" to enable typedown searching on this directory.

Tracing and Logging

We have already mentioned a number of different types of tracing

- DRS Traces (dir.*.trc, rda.*.trc)
- advmail.sckd (omsckd.log)

Also, throughout the OTN I've shown the output of the UAL Full/Command Reply traces.

UAL Tracing can be initiated on the server by setting

```
UAL_TRACE_LEVEL=
```

in the file ~openmail/sys/user.cfg/<OM userid>

Where <OM userid> is the numeric OpenMail user ID you wish to trace.

The output from a UAL trace can be found in

```
~openmail/tmp/<OM userid>.*
```

the exact filename depends on the logging level specified.

Full/Command Reply trace is logging level 8 and this writes a file called

```
OM<OM userid>U.log
```

it also creates files (again in tmp)

```
<OM userid>U.fnnnn
```

for any files that are exchanged between client and server, these files should be viewed as a set.

Support Tab Settings

The server UAL Trace level can also be set on the "Support" tab of the Properties of the OpenMail service. The simplest way to get to this from within the client is

- Tools -> Services
- Highlight the OpenMail Service
- Select Properties
- Go to the Support tab

(The Support tab would also be accessible when logging on).

On the support tab you'll see a number of other options for logging client side information. Firstly you can specify the location of the logfile, by default

```
C:\openmail.log
```

Then for each of the providers

- Transport
- Address Book
- Message Store

you have one of three options

- Errors only (the default)
- MAPI entry and exit points
- MAPI entry and exit points + commentary

There are then two further check boxes to log internal functions and also for setting the UAL trace level.

People often ask for the recommended logging levels. The answer to this is the recommended logging level is the one which shows what the problem is. For normal operation "Errors only" should be used.

In the event of a potential problem, the key thing is to try and reproduce the problem, i.e. define the steps that reliably cause the problem. Having done this the logging levels can then be increased as appropriate to help diagnose the cause.

I know that some problems are, by their nature, intermittent, but that does not mean we should run constantly with high logging levels. "MAPI entry and exit points + commentary" will log a large amounts of data, which all look very interesting but don't really make sense unless you have access to the service provider source.

Here's a taster, from a simple logon with commentary on the message store provider

```
HP OpenMail MAPI 1.0 Service Providers B.05.20.00 [Performance Build] \
                                                  - 05/28/98 : 01:03:29

bcaef fffa8ef3 fff8bca7 01:03:34 A>CMSProvider::Logon
 fffa8ef3 fff8bca7 01:03:34   Logging on to principal message store for \
ef                        principal Mr Mapi /canberra,class on server class
 fffa8ef3 fff8bca7 01:03:34   Created MSLogon Object
 fffa8ef3 fff8bca7 01:03:34   Intray access caps are 0x0000002F
 fffa8ef3 fff8bca7 01:03:34   OutTray access caps are 0x0000002F
 fffa8ef3 fff8bca7 01:03:34   PendingTray access caps are 0x0000002F
 fffa8ef3 fff8bca7 01:03:34   FilingArea access caps are 0x0000002F
 fffa8ef3 fff8bca7 01:03:34   BBArea access caps are 0x00000001
 fffa8ef3 fff8bca7 01:03:34    Profile was created by version B.05.20.00
 fffa8ef3 fff8bca7 01:03:34    A>CMsgStore::OpenEntry
 fffa8ef3 fff8bca7 01:03:34       Opening object <root folder> in
principal\
         message store for principal Mr Mapi /canberra,class on server
class
 fffa8ef3 fff8bca7 01:03:34         A>CMsgStore::AddRef
 fffa8ef3 fff8bca7 01:03:34         A<CMsgStore::AddRef(2)
 fffa8ef3 fff8bca7 01:03:34         A>CProperties::SetProps
 fffa8ef3 fff8bca7 01:03:34           Prop [0x00000000] ID: 0x00003414 \
                                                  Type: 0x00000102
 fffa8ef3 fff8bca7 01:03:34           CPropTagSet added ulPropTag :
0x34140102
 fffa8ef3 fff8bca7 01:03:34         A<CProperties::SetProps(0)
 fffa8ef3 fff8bca7 01:03:34         CPropTagSet emptied
 fffa8ef3 fff8bca7 01:03:34         Created CMAPIFolder object
 fffa8ef3 fff8bca7 01:03:34          Generating wrapped store ENTYRID
 fffa8ef3 fff8bca7 01:03:34         A>CMsgStore::GetProps
 fffa83 fff87 01:03:34           Prop [0] ID: 340d Type: 3
 fffa8ef3 fff8bca7 01:03:34         A<CMsgStore::GetProps(0)
 fffa83 fff8bca7 01:03:34          A>CProperties::SetProps
```

Additional Log Files

- ommapi.log

  The file ommapi.log is a permanent diagnostic log file which is appended to with each session and truncated when it reaches a certain size. The default location is

  C:\WINNT\ommapi.log

  It contains diagnostic information similar to that seen in the openmail.log file, for example

  HP OpenMail MAPI 1.0 Service Providers B.05.20.00 [Performance Build] \

  - 05/28/98 : 01:25:41

fff96a5f fffac457 01:25:41 Caught exception:

Exception Type: UAL Command

Function: OMSession::CheckReply

Source Line No: 2747

Error: 6063

Error1: 0

Error2: 0

fff96a5f fffac457 01:25:41

No free/busy time entry

However, only error information is logged whereas the information in openmail.log is dependent on the settings on the Support tab and is overwritten by each user session.

- omname.log

  All OpenMail name parsing is done via omname32.dll

  If there are specific name parsing problems then this dll has it'sown tracing capability.

  The file

  C:\WINDOWS\ual.ini

  controls the destination and level of logging through the entries

  [Name Parsing]

  Logfile=\OMNAME.LOG

  LogFlags=1

  The file does not appear to be created unless LogFlags is present and a positive value.

  I would suggest you note that this tracing is possible and wait for someone to ask you to set it and what level should be specified!

Message Store Viewer (MDBVU.EXE)

**********This section will give a brief overview of the message store viewer tool available from Microsoft.**********