Digital Investigation xxx (2018) 1–8

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

Navigating the Windows Mail database

Howard Chivers

Department of Computer Science, The University of York, United Kingdom

ARTICLE INFO

Article history:
Received 10 November 2017
Accepted 4 February 2018
Available online xxx

Keywords:
Windows Mail
Email
Message
Appointment
Calendar
ESE
Database
store.vol
Unistore
ESECarve

ABSTRACT

The Extensible Storage Engine (ESE) database is used to support many forensically important applications in the Windows operating system, and a study of how ESE is used in one application provides wider insights into data storage in other current and future applications. In Windows 10, Windows Mail uses an ESE database to store messages, appointments and related data; however, field (column) names used to identify these records are hexadecimal property tags, many of which are undocumented. To support forensic analysis a series of experiments were carried out to identify the function of these tags, and this work resulted in a body of related information about the Mail application. This paper documents property tags that have been mapped, and presents how Windows Mail artifacts recovered from the ESE *store.vol* database can be interpreted, including how the paths of files recorded by the Mail system are derived from database records. We also present examples that illustrate forensic issues in the interpretation of email messages and appointment records, and show how additional information can be obtained by associating these records with other information in the ESE database.

Introduction

The Microsoft Extensible Storage Engine (ESE[1]) is important to forensic practitioners because of the growing number of applications that use this database. In addition to Windows Search, the first forensically important user application to use ESE (Chivers and Hargreaves, 2011), the database now supports browser history and cache indexing (Internet Explorer and Edge), Cortana search records, System Resource Usage (SRU) records, Windows Mail, and most recently browser page management records such as user favourites.[2]

Detailed study of how a Windows application uses ESE has a wider benefit for forensic practitioners than the application itself, since an understanding of how ESE is used provides valuable insights into other applications that use the database, and continuity between applications often means that study of an application provides a good basis for understanding its successor. For example, the study of how Internet Explorer 10 used ESE (Chivers, 2014) applies equally to the Edge browser, and provides insights into how Windows Apps store Internet data.

Windows Mail was introduced as a *metro app* in Windows 8. It is a consumer product, free to users of the Windows operating system. In contrast with Microsoft Outlook, which is regarded as a professional or business product, it has simplified user facilities and fails to record some forensically important information (see section File content, below). However, an important feature of this application is its ability to integrate access to many different email accounts in a single user interface; if a user adds an account to the Mail application, then emails, contacts and appointments from that account are cached locally on the PC.

In Windows 10 there have been significant changes in how Windows Mail stores its data: the indexing, history and account information is now stored in an ESE database, and emails are stored in auxiliary text or html files, rather than *.eml* files. The relationship between the ESE database and other files is not documented and is one of the contributions of this paper.

The recovery of email evidence from the mail system is of practical forensic importance (Murphy et al., 2015), often prompted by text discovery within the database. Direct recovery of database records has the potential to provide more forensic evidence than can be retrieved via live analysis of a forensic image or by reading a database file via the ESE application programming interface (API). This is because database files may be inaccessible because they are often 'dirty' when a system is closed, and even if the file can be recovered to a 'clean' state via the API, data will be lost in the

E-mail address: hrchivers@iet.org.

[1] We acknowledge Microsoft copyright in terms used in this paper to describe Microsoft products, including: Windows, Windows 10, unistore and ESE.

[2] In recent versions of Windows 10 Favourites are now stored with reading lists in the *spartan.edb* database.

https://doi.org/10.1016/j.diin.2018.02.001
1742-2876/

process. A productive source of forensic evidence has also proved to be deleted database logs; deleted log files can often be recovered using standard forensic file recovery tools, and provide a rich source of historical database information. Such logs are not database files so cannot be read via the API, but contain recoverable database records.

ESECarve[3] is a tool developed by the Author that is capable of reading ESE databases via the standard application interface, or carving for ESE records. It has been in continuous development since 2010 and the work described here informed the next iteration of the tool. The main applications for carving have been the recovery of data from dirty or unrecoverable databases, and from recovered log files. ESECarve can read arbitrary databases and tables, but also has a set of modes that select filenames, prefixes and tables of most interest to forensic examiners. The forensic modes also provide application-specific data interpretation, since different applications use the underlying ESE data types in different ways, for example dates and times may be stored as strings, as binary or as integers.

A problem in interpreting ESE data from Windows Mail is that the field names for the data are hexadecimal property tags rather than meaningful names. We are not aware of a definitive reference to the meaning of these tags, so a mapping project was carried out to support field name translation for ESECarve. The resulting mapping of property tags is presented here, together with associated results of interest to forensic practitioners including how file and database content are related, and how message metadata is supplemented by content in other database tables (Store, Folders, Recipient, Attachment, Appointment).

The future support of Windows Phone and the Windows Mobile operating system is in doubt; however, the Windows Mail PC application uses the same ESE database and similar file structure as the Windows 10 Mobile operating system. Phone analysts will be familiar with the *store.vol* ESE database in Phone 8 and the MAPI hexadecimal property values some of which were present in the Phone 5 *fldr* database (Casey et al., 2010) and which have been further developed in the Windows Mail applications. The results described here have been checked against a sample database from Windows Phone 8 for consistency, but the primary focus of this paper is the PC application.

The next section provides a summary of the method used to develop these results. The main forensic contribution of this paper is presented in three sections. Files and database tables describes files recorded by the Mail system, and how file paths are derived from row identifiers within the ESE database. Properties introduces property tags and their interpretation, supported by Appendix A which lists tags which have now been identified. Finally Windows Mail database records provides example emails and appointment records to illustrate forensic issues in the interpretation of Windows Mail records, and shows how to associate message and appointment records with other information in the ESE database. The paper is concluded in Conclusion.

Method

There were few published academic source of information. Mail properties are available via an application programming interface and can be used to extract email related information, provided the target machine is live and it is feasible to run a forensic agent as an application (Jithin et al., 2015); however this project was not concerned with extracting data from a live database.

A search was made for relevant Microsoft documentation. Microsoft publishes core MAPI tags which are a basic set of hexadecimal tags used to label email properties (Microsoft, 2017a). They are distributed in header files for Office 2013 (Microsoft, 2017b) via the Outlook 2010 SDK; data types used within these tags are also documented (Microsoft, 2017c). These basic MAPI properties were found to correspond to fields in the *Message* database table, but never to properties in other tables, which as a consequence had to be determined by experiment. Analysts familiar with Exchange Server will be aware of tags used by this database; unfortunately the Exchange server extended tags are not used in the Windows Mail application.

A preliminary set of experiments were carried out by exchanging emails and appointments between 3 systems: a PC under test (Windows 10, build 15063.674), a gmail account (*user1*) linked to the PC Mail application, and a separate SMTP account (*user2*); some tests required additional accounts. Patterns of actions were contrived to cover typical messaging scenarios while maximising the number of settings (replies, attachments, priority etc) included. These tests provided a good basis for understanding how database records are linked to typical events and how emails, attachments etc in the file system were stored. To finally resolve certain properties and field behaviours it was then necessary to vary settings one at a time and observe the resulting changes in the database.

Property fields whose functions were identified were named in order of priority by: the MAPI tag name, the name in the iCalendar protocol (Daboo, 2009), or the name presented to the user in the Mail applet. Finally a check was carried out to ensure the resulting attributions were valid in a Windows Phone 8 sample which included SMS messaging.

In most experiments it was necessary to ensure that ESE data was written to the database before imaging, so the test PC was restarted before imaging the *store.vol* database. Even after this precaution the database was usually dirty on recovery; data from about 5% of the database samples had to be recovered by carving.

Files and database tables

This section describes files recorded by the Mail system, how the file paths are derived from row identifiers within the ESE database, and their types.

Paths for the ESE database and associated files are well known, they are:

```
%UserProfile%\AppData\Local\Comms\UnistoreDB\
store.vol
%UserProfile%\AppData\Local\Comms\Unistore\data\
```

The equivalent locations in the Windows Phone 8 sample were:

```
\Users\DefApps\AppData\Local\Comms\UnistoreDB\
store.vol
\SharedData\Comms\Unistore\data\
```

Note that the phone paths were found in a single sample of Windows Phone 8 and have not been widely confirmed. Other researchers identify different paths (Epifani et al., 2016) and also different partitions; the data partition on the sample was labelled 'Partition 33', whereas 'Partition 28' is reported elsewhere.

The database tables within *store.vol* of most forensic significance are: *Message, Contact, Appointment, Attachment, Recipient, Folders,* and *Store*. The use of these tables is described at Windows Mail database records below.

[3] ESECarve is a Windows application which is available at no cost to forensic practitioners and researchers, please email the author if you would like to use the software.

Please cite this article in press as: Chivers, H., Navigating the Windows Mail database, Digital Investigation (2018), https://doi.org/10.1016/j.diin.2018.02.001

Relationship of folders to tables

Files within the data folder are organised into numbered then lettered sub-directories, for example:

```
...data\3\n\4000020d000000031013.dat
```

The numbered sub-directories correspond to *store.vol* tables, as shown in Table 1.

The construction of filenames and the allocation of files to lettered sub-directories achieves a pseudo-random distribution of files within the numbered data folders, however for a given file and record type (e.g. appointment) there is a direct correspondence between filename and database record ID, and vice-versa. The relationship is shown in Fig. 1. The hexadecimal filename is divided into three sections. The most significant 8 characters are the hexadecimal value of the row ID, with the second character moved to the most significant position. The next most significant 8 characters carry the same table identification number as the numbered sub-directory (see Table 1) while the least significant 4 characters indicate file content.

The lettered sub-directory in which the file is placed is derived from the least significant hexadecimal digit of the related row ID by substituting the letters a-p for the numbers 0x0-0xF.

File content

The content of files stored by the mail system correspond to the table with which they are associated; for example, files indexed by the *Message* table contain the message part of the email body, files indexed by the *Attachment* table contain attachments, etc. The database tables, described below, record metadata including message header fields such as *From* and abstracts of the email message content.

Full email headers are not stored in either the database or in other files. From the forensic perspective this is particularly unfortunate since the chain of servers that delivered the email is not available within this mail system.

The least significant 4 hexadecimal characters in the filename indicate the use of the file, not necessarily its encoding, although that may be implicit. Content codes that have been identified are presented in Table 2. The three types used for contact images are not fully understood, they often appear in groups but in experiments all images associated with a single contact were hash-identical jpeg thumbnail images.

Table 1
Sub-Directories. Numbered sub-directories corresponding to *store.vol* tables.

Directory	Table
2	Contact
3	Message
5	Appointment
7	Attachment

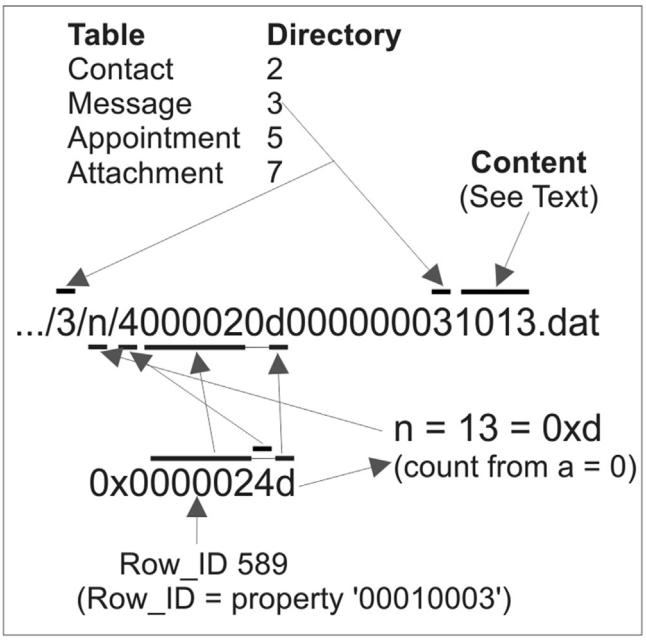

Fig. 1. Mapping Database Row IDs to filenames.

Properties

This section introduces hexadecimal property tags which identify record fields (database columns); tags whose functions have been positively identified are listed in Appendix A.

Database field names in the *store.vol* are 32bit property tags, usually displayed as eight hexadecimal digits and recorded in the ESE database as 8 character strings. These specify both the property represented in the field and its type; for example, the property tag 0x0037001f signifies email-subject, and the least significant part 0x001f specifies that the field is a Unicode string. In general the most significant 4 characters of the property tag specify the property and the least significant four characters its type. However, Microsoft documentation warns that the whole 8 characters are required to unambiguously specify a property - meaning that two property values with identical most significant 4 characters but different types may represent completely different properties; it is therefore unsafe to rely on just the 4 most significant characters to identify a field.

As noted above, properties confirmed by experiment for the *Message* table include those in the documented basic MAPI properties set. In some cases there are minor changes of type between the documented MAPI property tags and the data that are observed, for example 0x001E (ASCII) to 0x001F (Unicode) or 0x0013 (unsigned 32bit) to 0x0003 (signed 32bit). Properties determined by experiment are listed in Appendix A; MAPI properties are listed at Table A6, and previously undocumented properties are listed at Table A7.

Internally, ESE stores data in one of a range of specific data formats known as column types (Microsoft, 2015); these can be used to suggest property types that are not otherwise documented. Table 3 lists the property types that were observed, together with

Table 2
File Use. The least significant 4 characters of the filename is a tag which indicates the use of the file; it does not necessarily indicate the data encoding.

Tag	Type	Comment
001e	Unicode	Appointment details. (null-terminated with leading byte-order mark)
00ff	JPEG	Contact Photos or Icons (See text).
01a8	JPEG	
01b5	JPEG	
1000	Unicode	Message Content
1013	ASCII	
10b0	ASCII	Metadata in Name:Value Pairs.
	e.g. Contact:	REV:2016-10-22T19:05:16Z UID:2d6cd25209713f21
	e.g. Appointment:	CREATED:20170406T141734Z X-MICROSOFT-CDO-IMPORTANCE:1
3701	Any	Attachment. The same tag is used for all file types.

Please cite this article in press as: Chivers, H., Navigating the Windows Mail database, Digital Investigation (2018), https://doi.org/10.1016/j.diin.2018.02.001

Table 3
Property Types. This table lists the predominant property types used in the *store.vol* database; standard MAPI types are given if available, together with the underlying ESE database column types.

Type	MAPI name	MAPI format	ESE Type	ESE format
0x0003	PT_I4	4 byte signed integer	4	4 byte signed integer
0x000b	PT_BOOLEAN	2 byte 16bit boolean	1	1 byte boolean
0x0012	n/a		17	2 byte unsigned integer
0x0013	n/a		14	4 byte unsigned integer
0x0069	n/a		15	8 byte signed integer
0x001f	PT_TSTRING	null terminated Unicode	12	long text
0x0040	PT_SYSTIME	8-byte FileTime date	15	8 byte signed integer
0x0041	n/a		11	long binary
0x0064	n/a	(see text)	11	long binary

their MAPI format if known and the ESE column type used to store the property.

Most of the types are straightforward, however 0x0064 is a special binary format which encodes a list of email name pairs, for example (as usually presented) 'john smith [john.smith@nowhere.com]'. The names are unicode formatted, 4 byte aligned, and each pair is preceded by two 4 byte values the first of which is undiagnosed and the second provides the response code for that user (see Table 4). The whole list is preceded by a 4 byte entry count.

Windows Mail database records

The Message, Contacts and Appointments tables are self-contained and it will often be sufficient to review these tables in isolation to determine if there are records of interest. However, if more detail is required there is much to be gained by correlating message or appointment records with information from other database tables.

The example emails and appointment records in this section illustrate forensic issues in the interpretation of certain records, and show how additional information can be obtained by associating these records with information in other database tables. First, Message records are discussed, followed by their relation to Store, Folders, Recipient and Attachment records; Appointment records are then reviewed and related to information that may be found in messages.

The examples presented below were exchanged between two users. *user1* is the user of the account *user1@gmail.com* which is linked to Windows Mail in the user's computer. The second user has an account *user2@xxx.co.uk* on a different network and uses Microsoft Outlook as an SMTP email client.

Message records

The email exchange shown at Fig. 2 was recovered from *user1's* computer. In this exchange *user1* emails *user2* (9918) who replies with an email that includes two attachments (9905), *user1* then replies to the second email (9919). Windows Mail was powered off during this exchange between the gmail and the remote account; the computer was subsequently powered on allowing the Windows Mail cache to be updated. The fields in the figure are those needed for discussion; for a list of fields with mapped property names see Appendix A.

Table 4
Response codes.

Value	*Meaning*
1	Accepted
2	Declined
3	Tentative
4	Awaiting

RowID is an arbitrary number given to records to ensure that they are unique; most tables use such IDs to index records and they are distinguished by the same property tag (0x00010003). The *Note-Location* value is calculated by ESECarve from the *RowID* as shown above and provides the location of the email in the file system, it is not a field within the database table.

RowIDs may sometimes be used to infer the order of events; this is not the case in this database as the ID sequence does not follow the order of the emails. This is because the IDs are assigned as the emails are cached, which may not be in the same order that they actually arrived: Windows Mail has downloaded and cached a folder at a time, not emails in time order.

ConversationIDs reference RowIDs in the *Conversation* database table. Emails with the same number are part of the same conversation (also known as email thread). The *Conversation* table itself provides little additional forensic information.[4]

Dates and Times are UT (GMT), but note that some are locally added by Windows Mail (e.g. *RecordCachedTime*) while others are present in data provided by the linked email account (e.g.*MessageDeliveryTime*). The table also records a *LastModifiedTime* which is less useful forensically for linked accounts since it follows after the cached time; however, for email folders within the Windows Mail computer, such as draft emails, *LastModifiedTime* does record the last user update to the email. The *RecordCachedTime* records the time a record is cached, but not necessarily the first time it was cached, since it may be updated by subsequent synchronisation events.

Abstract provides a summary of the message content. Although the ESE type used to store this field allows very long text records, in practice the ASCII record is limited to the first 256 characters of any email, which provides a useful content preview. On the Windows Phone sample the Mail application was also used to store SMS messages; however, for SMS messages the *Abstract* field is empty and the whole message is found in the *Subject* field.

Subject Prefix is given a separate database field, the value of which is derived from the *Subject* header field in the email. If the subject starts with between one and three characters followed by a colon (e.g. 'X:', 'Re:', 'Fwd:') this is removed from the subject and stored separately as the prefix. The original subject including the prefix is displayed to the user as the email subject. If the user replies to the email a new prefix is added to the subject and the old prefix discarded. For example an email received with the subject 'Fwd: Topic' is stored in the database as a subject prefix of 'Fwd:' and a subject of 'Topic'; if the user replies to this email the reply will be sent with the subject of 'RE: Topic'. This avoids adding a sequence of 'Re:'s to emails that are part of a continuing conversation. This is not

[4] The *Conversation* table includes the number of emails in the conversation (property tag 0x36020013) and the RowID of the most recent message in the Conversation (property tag 0x82070013).

Please cite this article in press as: Chivers, H., Navigating the Windows Mail database, Digital Investigation (2018), https://doi.org/10.1016/j.diin.2018.02.001

Message

RowID (00010003)	Conversation ID(30150013)	MessageDeliveryTime (0e060040)	RecordCachedTime (82a50040)	Abstract (3fda001f)	SenderEmail (8279001f)	Subject (0037001f)	SubjectPrefix (003d001f)	<= Note-Location =>
9918	5024	08/07/2017 13:07:21	08/07/2017 13:36:19	Hullo	user1@gmail.com	test_mail		\3\o\b000260e00000003*
9905	5024	08/07/2017 13:25:20	08/07/2017 13:36:16	Hi, data attache	user2@xxx.co.uk	test_mail	RE:	\3\b\b000260100000003*
9919	5024	08/07/2017 13:32:04	08/07/2017 13:36:19	Thanks for data	user1@gmail.com	test_mail	Re:	\3\p\b000260f00000003*

Fig. 2. Selected fields from the message table.

Store

RowID (00010003)	DisplayName (3001001f)	Address (3003001f)	Protocol (8117001f)	DownloadEmailFrom (days) (81230013)
11	Gmail	user1" <user1@gmail.com>"	IMAP4	90

Message

RowID (00010003)	StoreID (00020003)	ConversationID (30150013)	NoOfAttachments (0e1b0012)	ParentFolderID (0e090013)	SenderEmail (8279001f)
9918	11	5024	0	45	user1@gmail.com
9905	11	5024	2	43	user2@xxx.co.uk

Folder

RowID (00010003)	StoreID (00020003)	ParentFolderID (0e090013)	DisplayName (3001001f)	CreateTime (30070040)
42	11	41	IPM.Root	07/11/2016 14:32
43	11	42	Inbox	07/11/2016 14:32
45	11	42	Sent Mail	07/11/2016 14:32

Fig. 3. Mapping message to store and folders.

uncommon behaviour in email systems, for example Outlook has similar processing, while others simply add new prefixes to the incoming message subject. It should be noted that prefixes may be removed as well as added and do not necessarily provide evidence of the history of the email conversation.

The *SenderEmail* field is included in the figure to distinguish the different emails in this example.

The table also includes an *Importance* field (not shown in the figure), reflecting metadata transmitted with the email. It was noted in testing that default values for this field vary between email clients. Examiners should be wary of concluding that an importance level was manually set by a user without knowing the behaviour of the specific user client.

Message stores and folders

Fig. 3 shows how emails in Fig. 2 can be linked to accounts and folders within those accounts via the *Store* and *Folders* tables.

StoreID in a message record identifies the *Store* table record of the linked account from which a message was obtained. The *Store* table contains a range of useful information not shown here, including connection information and how often the cache is updated.

DownloadEmailFrom specifies the number of previous days email that will be downloaded and kept in the Windows Mail cache. This field, and *DownloadNewEmail* which specifies how often the cache is updated, use apparently anomalous values to signal non-numeric user choices.[5]

Folder records are linked from **ParentFolderID** in the message record, and the folder tree can similarly be navigated within the *Folders* table; hence email 9905 is in the *IPM.Root|Inbox* gmail folder.

[5] Special values in these fields are not fully mapped, but 0xFFFFFFFF in *DownloadEmailFrom* signifies 'anytime' and the most significant bit set of *DownloadNewEmail* appears to signify 'based on usage'.

Please cite this article in press as: Chivers, H., Navigating the Windows Mail database, Digital Investigation (2018), https://doi.org/10.1016/j.diin.2018.02.001

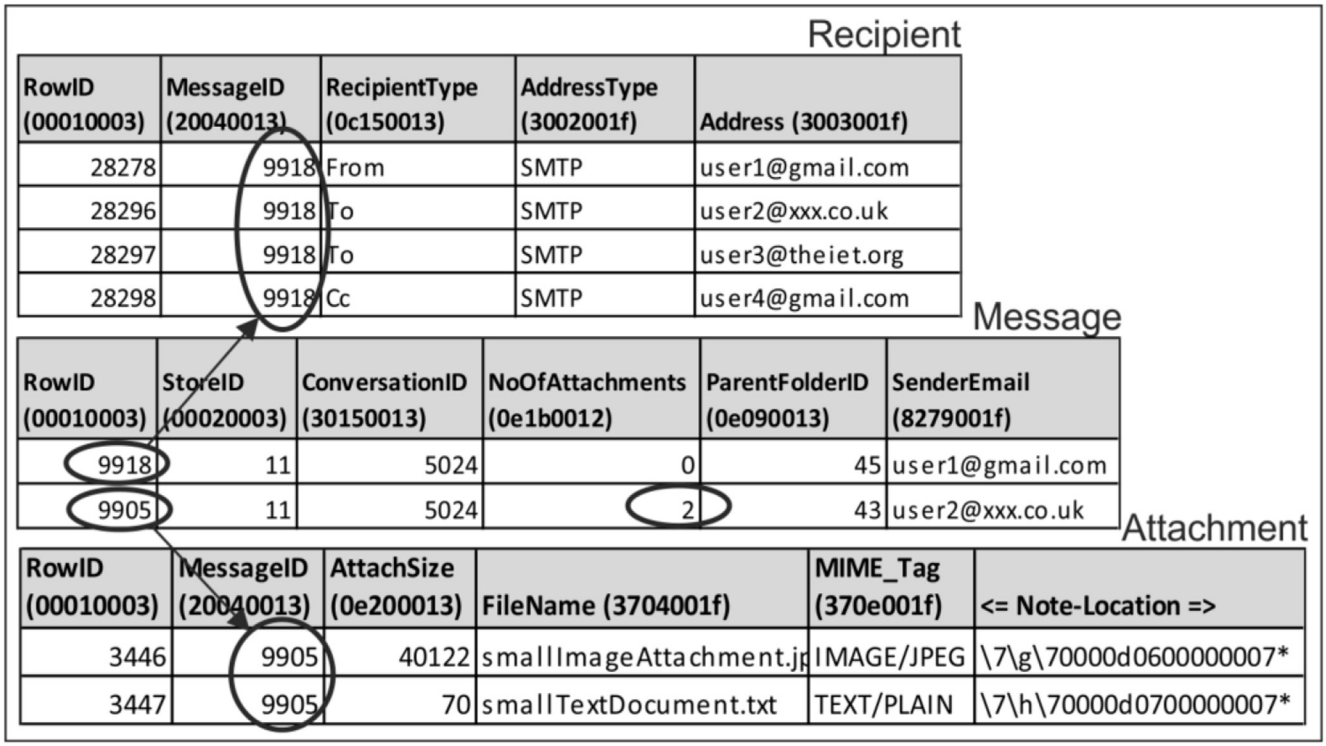

Fig. 4. Mapping message to recipient and attachment.

There were some anomalies in folder handling between Windows Mail and gmail. In the case of emails composed and sent directly from Windows Mail, draft emails have a class of IPM.MESSAGE, as opposed to sent emails which are classed as IPM.NOTE.[6] It was not possible to test if these same classes were used for emails in linked accounts since gmail drafts were never downloaded by Windows Mail. Also, in addition to the usual email folder (inbox, sent, etc) gmail has an *All Mail* folder which 'contains' every email in the user's account. Windows Mail obtained emails from this folder separately, so every email was found to have a duplicate record which referenced the *All Mail* folder. Emails from the *All Mail* folder do not share conversation identifiers with those from the normal folders.

Message recipients and attachments

Because there may be many recipients and attachments associated with a single email, there are no links from message records to these tables, instead recipients and attachments both have *MessageID* fields that indicate the ID of the message to which they are attached, as shown in Fig. 4.

The **Recipient** table lists all the recipients of each message, *RecipientType* specifying the association as shown in Table 5. The *Recipient* database table also contains further user metadata, if it is known. Accounts shown in *Recipient* are evidence of actual email connections; these email addresses are not necessarily in the user's *Contacts* table.

It would be logical to expect the *Sender* and *Reply-to* email header fields to also be found in this table, but the evidence suggests otherwise. Despite specific testing with unique *Sender* values this address appears not to be stored by the Mail system; property tags in the *Message* table which include the text 'Sender' refer to the *From* protocol element. The *Reply-to* header field is stored in the *Message* table under the 0x824401f property tag.

The presence of records in the **Attachment** table are indicated in a message record by the *NoOfAttachments* count. The attachment record includes the original filename and extension and the encoded size. The *RowID* can be used to locate the actual file, as described in Relationship of folders to tables, above.

Appointments

Fig. 5 provides three examples of appointments. Record 602 is a diary date set within Windows Mail to which no other users are invited. Record 605 is an appointment invitation sent from a gmail account linked to Windows Mail; the invitee responds by proposing a different time then subsequently accepts an updated appointment. Record 608 is an appointment invitation sent from a remote account and declined by the Windows Mail user. The two messages in the figure are associated with appointment 605 and are discussed in the sequel.

Appointments created within Windows Mail but never sent as invitations are simply diary entries. Such entries are distinguished by the lack of *Account* or *AdditionalPeople* information, as in record 602.

Table 5
Recipient Type Codes.

Value	Meaning
0	From
1	To
2	Cc
3	Bcc

[6] *MessageClass* is available as a field within the message records, although not shown in this figure for space reasons, IPM (Interpersonal Message Code) is a Microsoft code which specifies the function of a message, see list at (Microsoft, 2017d).

Appointment

RowID (00010003)	Update Count (381a0013)	Length (Mins) (10400013)	StartTime (10420040)	Response (10450013)	Event (0020001f)	Location (0041001f)	Account (0055001f)	AdditionalPeople (01c20064)	<= Note-Location =>
602	0	30	17/06/2017 16:00	Awaiting	test 1	office		-	\5\k\5000020a00000005*
605	1	30	17/06/2017 18:00		test 8	home	user1@gmail.com	user2 [user2@xxx.co.uk] (Accepted)	\5\n\5000020d00000005*
608	0	30	17/06/2017 19:30	Declined	test 16	cafe	user2@xxx.co.uk	user1 [user1@gmail.com] (Declined)	\5\a\6000020000000005*

Messages related to appointment ID 605

Message

RowID (00010003)	Conversation (30150013)	StartDate (00600040)	EndDate (00610040)	MessageClass (001a001f)	SenderEmail (82790001f)	Subject (0037001f)	Place (852a001f)
9529	4782	17/06/2017 18:30	17/06/2017 19:00	IPM.Schedule.Meeting.Request	user2@xxx.co.uk	New Time Proposed: test 8	home
9537	4785	17/06/2017 18:00	17/06/2017 18:30	IPM.Schedule.Meeting.Resp.Pos	user2@xxx.co.uk	Accepted: test 8	home

Fig. 5. Selected fields from the appointment table with related messages.

Account is the address of the meeting organiser whose name is also given in an *Organiser* field which is not shown in the figure. The *Account* values in appointments 605 and 608 indicate the originator of the appointment, not the account being viewed.

UpdateCount is incremented if an appointment is modified. Experiments confirm that the update count is incremented only following a change in the appointment (e.g. change of time or venue) and not as a result of a change in the appointment database record (e.g. if *response* was changed from awaiting to declined). The Update count of record 605 was incremented to 1 when the appointment time changed.

Response is encoded as shown in Table 4. This is the response sent from Windows Mail, or the account to which it is linked, to an appointment invitation from another user. It does not record the response from other meeting participants to an invitation.

AdditionalPeople lists all the user names and accounts associated with an appointment together with their responses. This is obviously an important field from the forensic perspective, both for documenting meetings and also for determining a user's associates, since the users recorded here do not necessarily appear in the Contacts table. The underlying data are binary with property tag 0x01c20064; the encoding is described in section Properties, above.

The file associated with an appointment record is the body of the invitation email, and provides user-readable information such as time, location and appointment details. File locations are determined from RowIDs as described above.

Appointment requests are sent between systems using emails and the emails recorded in the message table provide some additional information, particularly relating to the history of updated appointments. The message function is given by the message class field,[7] which indicates that email 9529 is a meeting proposal (IPM.Schedule.Meeting.Request), followed by 9537 which accepts an updated appointment (IPM.Scedule.Meeting.Resp.Pos). The *subject* field is helpful, indicating that the request in message 9529 is a proposed new time.

The complete email chain used to agree appointments is not found in the message table, only incoming appointment-related messages are recorded. The iCal (Calendar) MIME data sent with the messages was also not found in attachments or other files linked to Windows Mail. As a consequence it is necessary to match fields such as time, location and event to associate messages with appointments. Note that IPM.Schedule.Meeting.Request messages are not necessarily original appointment invitations, they are also used to propose changes in response to requests, as in email 9529.

Conclusion

The work described in this paper began as a project to map undocumented property tags in Windows Mail for forensic analysis; in the process much was learned about how records and fields within Windows Mail are used, how information between tables can be navigated and how database records are linked to files.

The results presented here are conservative in the sense that the behaviours described were consistently observed in a range of experiments. Nevertheless the observations may be limited by the use of gmail as the primary system linked to Windows Mail. Using a non-Microsoft email source ensured that standard protocols were used by Windows Mail for caching, and gmail was chosen because of its widespread use. However, the message table in particular is large and sparsely used; there is more to be discovered here, and it may be that if the user relied on direct connections to Microsoft-based email accounts, a richer set of information would be available.

The paper has presented a range of information concerning how Windows Mail artifacts recovered from the ESE *store.vol* database can be interpreted, including how the paths of files recorded by the Mail system are derived from row identifiers within the ESE database, the meaning of property tags, and how the fields they label are interpreted. We also present emails and appointments that illustrate forensic issues in the interpretation of certain record fields, and show how additional information can be obtained by associating these records with other information in the ESE database. A detailed list of property tags that have been mapped is presented at A.

Appendix A. Mapped *store.vol* Properties

[7] Messages classes for IPM.Schedule.Meeting are .Request, .Canceled, .Resp.Pos (Accept), .Resp.Neg (Decline), .Resp.Tent (Tentative).

Please cite this article in press as: Chivers, H., Navigating the Windows Mail database, Digital Investigation (2018), https://doi.org/10.1016/j.diin.2018.02.001

Table A.6
*MAPI Properties.*This table lists *unistore* properties that either correspond exactly with those in the documented MAPI property list, or whose types are a close match and whose names consistently match table content. The *Tables* column indicates the tables in which the properties are usually found.

Property	Name	Type	Tables
00170013	Importance	uInt32	Message
001a001f	MessageClass	Unicode	Message
0037001f	Subject	Unicode	Message
003d001f	SubjectPrefix	Unicode	Message
00600040	StartDate	FileTime	Message
00610040	EndDate	FileTime	Message
0c150013	RecipientType	uint32	Recipient
0c1a001f	SenderName	Unicode	Message
0c1f001f	SenderAddress	Unicode	Message
0e060040	MessageDeliveryTime	FileTime	Message
0e080013	MessagerSize	uInt32	Message
0e090013	ParentFolderID	uInt32	Message, Contact, Folders
0e200013	AttachSize	uInt32	Attachment
3001001f	DisplayName	Unicode	Message
3002001f	AddressType	Unicode	Message
3003001f	Address	Unicode	Message
30070040	CreateTime	FileTime	Folders
30080040	LastModifiedTime	FileTime	Message
3704001f	FileName	Unicode	Attachment
370e001f	MIME_Tag	Unicode	Attachment
3a0b001f	Keyword	Unicode	Message

Table A.7
Undocumented Properties. This table lists properties that are used in *unistore* and not found in the standard MAPI property list. The *Tables* column indicates the tables in which the properties are usually found.

Property	Name	Type	Tables
00010003	RowID	Int32	Message, Folders
00020003	StoreID	int32	Message, Folders
0020001f	Event	Unicode	Appointment
0022000b	Repeat	Boolean	Appointment
0041001f	Location	Unicode	Appointment
0044000b	AllDay	Boolean	Appointment
00450013	Status	uInt32	Appointment
00460013	ReminderTime (Mins)	uInt32	Appointment
0051001f	Organiser	Unicode	Appointment
0055001f	Account	Unicode	Appointment
0070001f	ConversationTopic	Unicode	Message
0080001f	DisplayName	Unicode	Contact, Folders
0082001f	FirstName	Unicode	Contact
0084001f	LastName	Unicode	Contact
0090001f	Email	Unicode	Contact
0091001f	EmailWork	Unicode	Contact
0092001f	EmailOther	Unicode	Contact
0099001f	WorkPhone	Unicode	Contact
00d1001f	Address	Unicode	Contact
010a0041	UserID	Binary	Message,Appointment
01bf000b	HasName	Boolean	Contact
01c20064	AdditionalPeople	namelist	Appointment
0e1b0012	NoOfAttachments	uint16	Message
10400013	Duration (Mins)	uInt32	Appointment
10420040	StartTime	FileTime	Appointment
10450013	Response	uInt32	Appointment
10900013	FlagStatus	uInt32	Message
20040013	MessageID	uInt32	Recipient
30150013	ConversationID	uInt32	Message
3701000b	Received	Boolean	Attachment
37150013	SynchOptionFlags	uInt32	Store
37740069	Flag	uint64	Message
381a0013	Update Count	uInt32	Message,Appointment
3fda001f	Abstract	Unicode	Message
8117001f	Protocol	Unicode	Store
81200013	DownloadNewEmail (Mins)	uInt32	Store
81230013	DownloadEmailFrom (Days)	uInt32	Store
81240012	OutgoingRequresAuthentication	uint16	Store
8125001f	IncomingEmailServer	Unicode	Store
8126001f	OutgoingEmailServer	Unicode	Store
8129001f	OutgoingEmailServerUsername	Unicode	Store
813b001f	ContactsServer	Unicode	Store
813c001f	CalandarServer	Unicode	Store
8244001f	SendRepliesTo	Unicode	Message
82620013	Read	uInt32	Message
8279001f	SenderEmail	Unicode	Message
82a50040	RecordCachedTime	FileTime	Message
851d0041	Encoding	Binary	Appointment
85290040	ResponseTime	FileTime	Message
852a001f	Place	Unicode	Message
853d0040	TimeofRequest	FileTime	Message
85450013	ReminderTime (Secs)	uInt32	Message

References

Casey, E., Bann, M., Doyle, J., 2010. Introduction to Windows Mobile Forensics.
Chivers, H., 2014. Private browsing: a window of forensic opportunity. Digit. Invest. 11 (1), 20–29.
Chivers, H., Hargreaves, C., 2011. Forensic data recovery from the windows search database. Digit. Invest. 7 (3), 114–126.
Daboo, C., December 2009. icalendar Transport-independent Interoperability Protocol (iTIP). RFC 5546, RFC Editor.
Epifani, M., Picasso, F., Scarito, M., 2016. Discovering Windows Phone 8 Artifacts and Secrets. In: Proceedings of the 2016 Digital Forensic Research Workshop (DFRWS). https://www.dfrws.org/conferences/dfrws-eu- 2016/sessions/discovering-windows-phone-8-artifacts-and-secrets (Accessed May 2018).
Jithin, S., Satheesh Kumar, S., Jinu Kumar, S., 2015. A novel method for windows phone forensics. Int. J. Sci. Eng. Res. 6 (1), 1044–1047.
Microsoft, 2015. Ese Column Types. https://msdn.microsoft.com/en-us/library/gg269213(v=exchg.10).aspx (Accessed March 2018).
Microsoft, 2017. Mapi Property Tags. https://msdn.microsoft.com/en-us/library/ms526356(v=exchg.10).aspx (Accessed March 2018).
Microsoft, 2017. Mapi Header Files. https://msdn.microsoft.com/en-us/library/office/cc842403.aspx (Accessed March 2018).
Microsoft, 2017. Mapi Property Types. https://msdn.microsoft.com/en-us/library/office/cc839705.aspx (Accessed March 2018).
Microsoft, 2017. Item Types and Message Classes. https://msdn.microsoft.com/en-us/vba/outlook-vba/articles/item-types-and-message-classes (Accessed March 2018).
Murphy, C., Leong, A., Gaffney, M., Punja, S.G., Gibb, J., McGarry, B., 2015. Windows Phone 8 Forensic Artifacts. https://uk.sans.org/reading-room/whitepapers/forensics/windows-phone-8-forensic-artifacts-35787 (Accessed March 2018).

Please cite this article in press as: Chivers, H., Navigating the Windows Mail database, Digital Investigation (2018), https://doi.org/10.1016/j.diin.2018.02.001