Personal Folder File (PFF) forensics

Analyzing the horrible reference file format

By Joachim Metz <forensics@hoffmannbv.nl>
Hoffmann Investigations

Summary

In forensic investigation of corporate environments one of the more frequent encountered e-mail clients is Microsoft Outlook. It uses the Personal Folder File (PFF) format in PST, OST and PAB files. Because PFF is a propriety file format, little information about it is available in the public domain. As a consequence, it is unclear how well different forensic tools support the PFF format and the different Outlook files.

This document provides an overview of the PFF format and its intricacies regarding forensic analysis.

Regarding the subtitle the horrible reference file format refers to the fact that PFF contains a lot of obscure references. It is also a pun on the horrible property file format which refers to certain aspects of the OLE 2 compound file format, another Microsoft file format mainly used in Microsoft Office 97 – 2003 files.

Document information

Author(s): Joachim Metz <forensics@hoffmannbv.nl>

Abstract: This document contains information about the forensic analysis of the Personal Folder File (PFF) format.

Classification: Public

Keywords: PFF, Personal Folder File, OFF, Offline Folder File, PAB, Personal Address Book, PST, Personal Storage Table, OST, Outlook Storage Table

License

Version

Version	Author	Date	Comments
1.0	Joachim Metz	March 11, 2009	Initial version.
1.1	Joachim Metz	August 17, 2009	Corrections, example did not match text.

Table of Contents

1. Overview of the PFF format

The Personal Folder File (PFF) format is mainly used by Microsoft Outlook in PST, OST and PAB files. Microsoft has kept the specification of PFF closed. The information in this document was obtained by the information available on the Internet and reverse engineering of the file format. The information obtained is documented in the Personal Folder File format specification [PFF08].

Basically a PFF consists of the following elements:
- file header
- file header data
- blocks containing:
  - allocation tables
  - index nodes
  - item data and list structures
  - data

The following paragraphs provide an overview of these elements.

1.1. File header

The PFF starts with a file header which is at least 16 bytes of size.

```
0000000: 21 42 44 4e d3 4b 10 c0 53 4d 17 00 13 00 01 01  !BDN.K..SM......
```

The first 4 bytes of this header contain the unique signature '!BDN' signifying the PFF format. Other significant values in the header are the content and data type.

The 9th and 10th byte contain the content type which is 'SM' in this case. The content type signifies if the file contains Personal Storage Table (PST). Other content types are Offline Storage Table (OST) and Personal Address Book (PAB). The PST and OST data are very similar, the PAB is not.

The 11h and 12th byte contain the data type which is 0x0017 in this case. The data type refers to the type of file, which can be a 32-bit or 64-bit version. In the example above the data type refers to a 64-bit PFF version. The main difference between the 32 and 64-bit PFF versions are the size of file offset values. A 32-bit PFF mainly uses extended ASCII strings with a codepage while a 64-bit PFF uses UTF-16 strings.

The file header is followed by data type specific file header data. The actual structure of this data differs for the 32 and 64-bit PFF versions but consists of similar values.

1.2. Allocation tables

The file header contains master allocation tables of which the exact application has not yet been fully reversed engineered. However the master allocation tables tie closely together with the allocation tables starting at file offset of 17408 (0x04400). The values after the file header data and before offset 17408 are likely to be used for the master allocation tables. This also has not yet been fully reversed engineered.

The block at offset 17408 contains the first data structure allocation table. In this allocation table every bit represents a block of 64 bytes. The allocation table contains 496 bytes of allocation bits.

This means an allocation table represents 496 x 8 x 64 = 253952 bytes of blocks. The allocation table is 512 bytes of size and is repeated every 253952 (0x03e000) bytes.

At file offset 17920 (0x04600) the first index node allocation table is found. In this allocation table every bit represents a block of 512 bytes. The allocation table contains 496 bytes of allocation bits. This means a allocation table represents 496 x 8 x 512 = 2031616 bytes of blocks. The allocation table is 512 bytes of size and is repeated every 2031616 (0x01f0000) bytes.

Starting from file offset 18432 (0x04800) the PFF contains multiple data structure and index node blocks.

1.3. Indexes

The file header data contains two index references: a data structure index and an item descriptor index. These indexes are B-trees.

The data structure index contains information about the data and list structures that make up the items in the PFF. It assigns a unique identifier to:
- a file offset of the data structure;
- a size of the data structure.

Note

The item descriptor index is referred to as index1 by libpst. Scanpst refers to it as the BBT.

The item descriptor index contains information about the items within the PFF. It assigns a unique identifier to:
- a data structure index identifier of the item table;
- a data structure index identifier of the local descriptor list;
- a parent item descriptor identifier.

Note

The item descriptor index is referred to as index2 by libpst. Scanpst refers to it as the NBT.

There are index nodes containing a type indicator of 0x85 0x85. These index nodes seem to be used temporarily but can be found in PFFs. It is possible that these index nodes are used to balance the index B-tree.

1.4. The PFF item

The actual data of an item within a PFF is scattered over different data structures:
1. The item descriptor index node is the main structure to find an item. It contains a reference to the item table and the local descriptor list.
2. The local descriptor list contains a list of descriptor identifiers. The local descriptor list provides data structure index identifiers of the item table and sub lists.
3. The item table contains the item values. These values can contain the actual data or refer to a descriptor in the local descriptor list. I.e. an item table containing e-mail item values contains item value references pointing to the actual 'Mail Internet Header', and 'Plain Text

Body' data. The Outlook Message API (MAPI) refers to the combination of the item type and item value type as the property type i.e. PR_BODY (Plain Text Body).

There are different types of item tables. All the different tables store the item values differently. The table types currently known are:

- the bc table, used by items to store a single set of item values i.e. e-mail, task, appointment, etc.;
- the 7c table, used by items to store a limited amount of multiple sets of item values i.e. attachments;
- the ac table, used by items with larger multiple sets of item values;
- the a5 table, used to store item values for the ac table;
- the 9c table, used to store GUID descriptor relationships.

Note

The terms Global Unique Identifier (GUID), Universal Unique Identifier (UUID) and (OLE) class identifier are used interchangable in this document.

Besides the item tables and descriptor list, the PFF has another data structure, namely the data array. Data arrays are used to store large data structures, for which they combine multiple data structure blocks. The data array actually contains a set of data structure index identifiers referring to data structure blocks it entails.

1.5. The PFF item hierarchy

The PFF items are stored in a hierarchy. There are three types of PFF items:

- items that are parts of the item folder hierarchy, like folders, e-mail messages, appointments, etc.;
- items that are used for special purposes, like the message store, name-to-identifier map, etc. These items are mainly stored outside of the item folder hierarchy;
- items that are not part of the item folder hierarchy but linked to items that are, such as attachments and embedded e-mail items.

The item root folder has a parent item descriptor value which refers to itself. Items part of the item folder hierarchy have parent values. The special purpose items do not have a parent value (more accurately they contain a parent value of 0).

Attachments are not part of the PFF item hierarchy. I.e. an e-mail item contains a reference to the attachments table in its local descriptor list. The attachments table contains references to the individual attached items, which in their turn contain a reference to the actual attached data.

1.6. Encryption

The information in a PFF can be encrypted using multiple encryption types. The encryption type is stored in the file header data. Currently three encryption types are known; none, compressible and high encryption. The compressible and high encryption are actually more of a way to obfuscate the information in the PFF than real means to ensure confidentiality.

When a PFF is encrypted, only the item tables and item data are encrypted; the internal data structures are not.

The bad news for forensic analysis is that PFF obfuscates the information in the data structures which makes a basic text string search impossible. For both compressible and high encryption it uses a substitution cypher. For compressible encryption the algorithm the result is always the same. For the high encryption the data structure index identifier is used as a key.

2. Manual analysis of a PFF item

The best way to understand the intricacies of analyzing a PFF file is by example of a manual analysis. The following chapter describes a manual analysis of a PFF e-mail item and its attachments.

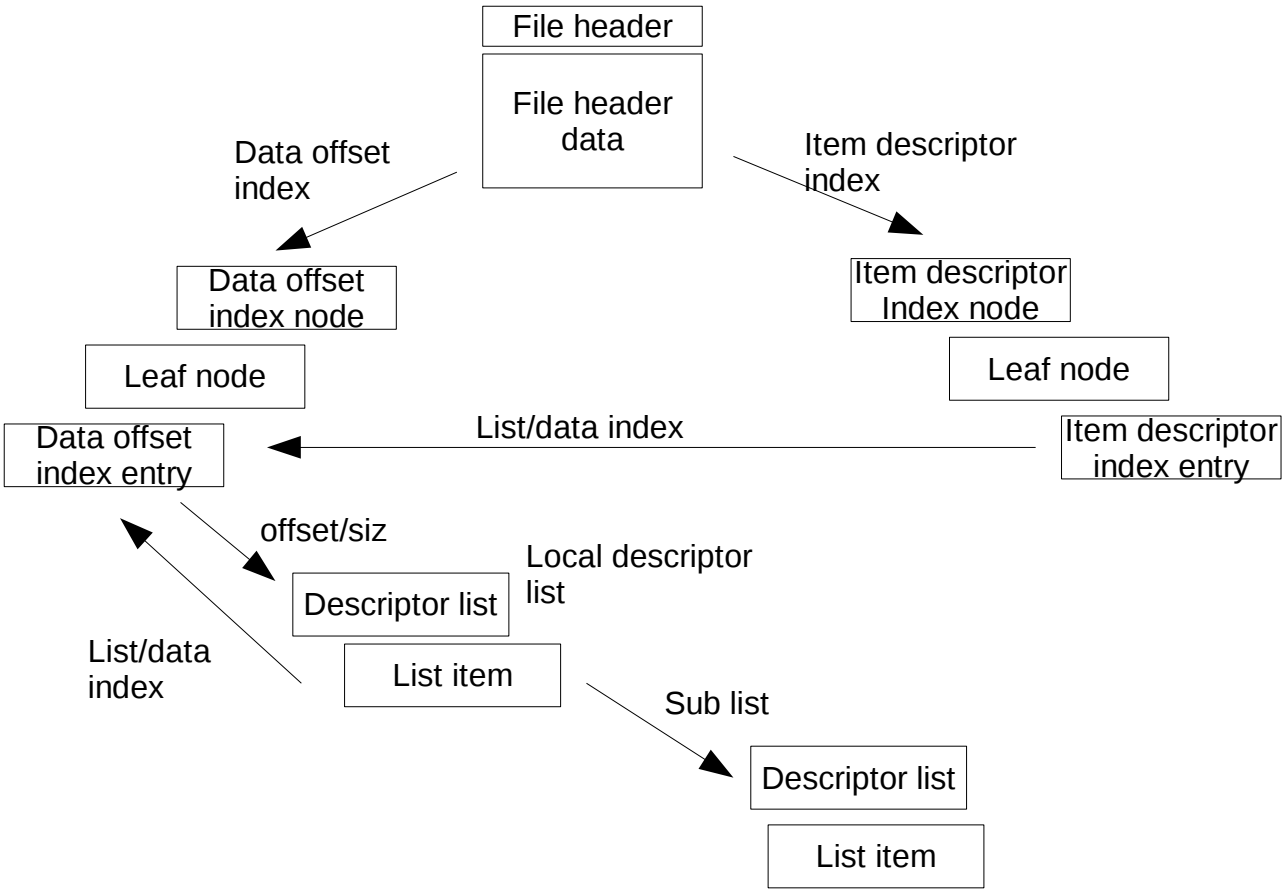
In the example a 64-bit PFF was used.

2.1. The e-mail item descriptor

The first value needed to find a specific PFF item is the item descriptor index identifier. This identifier can be found in the item descriptor index. In this case, the item descriptor index entry has the following values:

- identifier: 2097188
- item table: 1012
- local descriptor list: 982
- parent item: 32898

The corresponding item table and descriptor list data structure index identifiers can be found in the data structure index.



Note

To make matters more confusing. Currently libpff refers to the data structure index as the data offset index, because it contains the offsets of the data structures. This will probably be corrected in the near future.

The data structure index entries for data and lists have the following values:

item table:

- identifier: 1012
- offset: 108800
- size: 1454

local descriptor list

- identifier: 982
- offset: 45824
- size: 368

The unencrypted list data structure consists of the following data:

```
00000000: 02 00 0f 00 00 00 00 00  71 06 00 00 00 00 00 00   ........ q.......
00000010: cc 03 00 00 00 00 00 00  d2 03 00 00 00 00 00 00   ........ ........
00000020: 92 06 00 00 05 37 03 00  ac 03 00 00 00 00 00 00   .....7.. ........
00000030: 00 00 00 00 00 00 00 00  25 80 00 00 05 37 03 00   ........ %....7..
00000040: f8 01 00 00 00 00 00 00  06 02 00 00 00 00 00 00   ........ ........
00000050: 45 80 00 00 34 00 f4 77  20 02 00 00 00 00 00 00   E...4..w  .......
00000060: 2e 02 00 00 00 00 00 00  65 80 00 00 16 00 1c 00   ........ e.......
00000070: 48 02 00 00 00 00 00 00  56 02 00 00 00 00 00 00   H....... V.......
00000080: 85 80 00 00 00 40 dd a3  70 02 00 00 00 00 00 00   .....@.. p.......
00000090: 7e 02 00 00 00 00 00 00  a5 80 00 00 60 00 00 00   ~....... ....`...
000000a0: 98 02 00 00 00 00 00 00  a6 02 00 00 00 00 00 00   ........ ........
000000b0: c5 80 00 00 9d 02 00 00  c0 02 00 00 00 00 00 00   ........ ........
000000c0: ce 02 00 00 00 00 00 00  e5 80 00 00 67 00 37 00   ........ ....g.7.
000000d0: e8 02 00 00 00 00 00 00  f6 02 00 00 00 00 00 00   ........ ........
000000e0: 05 81 00 00 6a 00 70 00  10 03 00 00 00 00 00 00   ....j.p. ........
000000f0: 1e 03 00 00 00 00 00 00  25 81 00 00 67 00 09 00   ........ %...g...
00000100: 38 03 00 00 00 00 00 00  46 03 00 00 00 00 00 00   8....... F.......
00000110: 45 81 00 00 e6 00 00 00  60 03 00 00 00 00 00 00   E....... `.......
00000120: 6e 03 00 00 00 00 00 00  65 81 00 00 00 00 00 00   n....... e.......
00000130: 88 03 00 00 00 00 00 00  9e 03 00 00 00 00 00 00   ........ ........
00000140: 9f 81 00 00 30 00 33 00  b2 03 00 00 00 00 00 00   ....0.3. ........
00000150: 00 00 00 00 00 00 00 00  bf 81 00 00 53 00 4d 00   ........ ....S.M.
00000160: de 03 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

The list starts with 0x02 identifying it as a list and 0x00 meaning that the list contains local descriptor values and does not refer to sublists. The value 0x0f 0x00 after it specifies the amount of elements in the list, which in this case are 16 elements that contain the following data:

```
element: 01 identifier                  : 1649
element: 01 data identifier             : 972
element: 01 list identifier             : 978
element: 02 identifier                  : 1682
element: 02 data identifier             : 940
element: 02 list identifier             : 0
element: 03 identifier                  : 32805
element: 03 data identifier             : 504
element: 03 list identifier             : 518
element: 04 identifier                  : 32837
element: 04 data identifier             : 544
element: 04 list identifier             : 558
element: 05 identifier                  : 32869
element: 05 data identifier             : 584
element: 05 list identifier             : 598
element: 06 identifier                  : 32901
element: 06 data identifier             : 624
element: 06 list identifier             : 638
element: 07 identifier                  : 32933
element: 07 data identifier             : 664
element: 07 list identifier             : 678
element: 08 identifier                  : 32965
element: 08 data identifier             : 704
element: 08 list identifier             : 718
element: 09 identifier                  : 32997
element: 09 data identifier             : 744
element: 09 list identifier             : 758
element: 10 identifier                  : 33029
element: 10 data identifier             : 784
element: 10 list identifier             : 798
```

```
element: 11 identifier                  : 33061
element: 11 data identifier             : 824
element: 11 list identifier             : 838
element: 12 identifier                  : 33093
element: 12 data identifier             : 864
element: 12 list identifier             : 878
element: 13 identifier                  : 33125
element: 13 data identifier             : 904
element: 13 list identifier             : 926
element: 14 identifier                  : 33183
element: 14 data identifier             : 946
element: 14 list identifier             : 0
element: 15 identifier                  : 33215
element: 15 data identifier             : 990
element: 15 list identifier             : 0
```

This list is called the local descriptor list, which is used to find data and list offsets of descriptors in the item table.

2.2. The e-mail table

The actual e-mail content data of a PFF e-mail item is largely contained in an item table data structure. The unencrypted table data structure consists of the following data:

```
00000000: 7e 05 ec bc 20 00 00 00  00 00 00 00 b5 02 06 00   ~... ... ........
00000010: 60 00 00 00 49 00 50 00  4d 00 2e 00 4e 00 6f 00   `...I.P. M...N.o.
00000020: 74 00 65 00 1a 00 1f 00  40 00 00 00 37 00 1f 00   t.e..... @...7...
00000030: 20 02 00 00 39 00 40 00  00 01 00 00 41 00 02 01    ...9.@. ....A...
00000040: 80 01 00 00 42 00 1f 00  40 01 00 00 64 00 1f 00   ....B... @...d...
00000050: 00 02 00 00 65 00 1f 00  c0 01 00 00 70 00 1f 00   ....e... ....p...
00000060: 60 02 00 00 19 0c 02 01  60 01 00 00 1a 0c 1f 00   `....... `.......
00000070: 20 01 00 00 1e 0c 1f 00  e0 01 00 00 1f 0c 1f 00    ....... ........
00000080: a0 01 00 00 04 0e 1f 00  80 02 00 00 06 0e 40 00   ........ ......@.
00000090: e0 00 00 00 07 0e 03 00  10 00 00 00 08 0e 03 00   ........ ........
000000a0: 47 cc 00 00 00 10 1f 00  bf 81 00 00 13 10 02 01   G....... ........
000000b0: 9f 81 00 00 07 30 40 00  a0 00 00 00 08 30 40 00   .....0@. .....0@.
000000c0: c0 00 00 00 0b 30 02 01  80 00 00 00 de 3f 03 00   .....0.. .....?..
000000d0: e4 04 00 00 04 66 03 00  c7 59 00 00 10 66 03 00   .....f.. .Y...f..
000000e0: 4e 01 00 00 19 66 1f 00  a0 02 00 00 05 80 03 00   N....f.. ........
000000f0: 98 c3 01 00 06 80 1f 00  40 02 00 00 14 80 0b 00   ........ @.......
00000100: 01 00 00 00 d2 fb 6e ae  04 72 8f 40 a4 7a 12 0c   ......n. .r.@.z..
00000110: 77 e1 2a 42 00 6b 26 ff  32 76 c9 01 00 6b 26 ff   w.*B.k&. 2v...k&.
00000120: 32 76 c9 01 a0 26 34 ff  32 76 c9 01 a0 26 34 ff   2v...&4. 2v...&4.
00000130: 32 76 c9 01 4f 00 75 00  74 00 6c 00 6f 00 6f 00   2v..O.u. t.l.o.o.
00000140: 6b 00 20 00 32 00 30 00  30 00 33 00 20 00 54 00   k. .2.0. 0.3. .T.
00000150: 65 00 61 00 6d 00 4f 00  75 00 74 00 6c 00 6f 00   e.a.m.O. u.t.l.o.
00000160: 6f 00 6b 00 20 00 32 00  30 00 30 00 33 00 20 00   o.k. .2. 0.0.3. .
00000170: 54 00 65 00 61 00 6d 00  00 00 00 00 81 2b 1f a4   T.e.a.m. .....+..
00000180: be a3 10 19 9d 6e 00 dd  01 0f 54 02 00 00 01 80   .....n.. ..T.....
00000190: 4f 00 75 00 74 00 6c 00  6f 00 6f 00 6b 00 20 00   O.u.t.l. o.o.k. .
000001a0: 32 00 30 00 30 00 33 00  20 00 54 00 65 00 61 00   2.0.0.3.  .T.e.a.
000001b0: 6d 00 00 00 53 00 4d 00  54 00 50 00 00 00 6f 00   m...S.M. T.P...o.
000001c0: 6c 00 74 00 65 00 61 00  6d 00 40 00 6d 00 69 00   l.t.e.a. m.@.m.i.
000001d0: 63 00 72 00 6f 00 73 00  6f 00 66 00 74 00 2e 00   c.r.o.s. o.f.t...
000001e0: 63 00 6f 00 6d 00 00 00  00 00 00 00 81 2b 1f a4   c.o.m... .....+..
000001f0: be a3 10 19 9d 6e 00 dd  01 0f 54 02 00 00 01 80   .....n.. ..T.....
00000200: 4f 00 75 00 74 00 6c 00  6f 00 6f 00 6b 00 20 00   O.u.t.l. o.o.k. .
00000210: 32 00 30 00 30 00 33 00  20 00 54 00 65 00 61 00   2.0.0.3.  .T.e.a.
```

```
00000220: 6d 00 00 00 53 00 4d 00  54 00 50 00 00 00 6f 00   m...S.M. T.P...o.
00000230: 6c 00 74 00 65 00 61 00  6d 00 40 00 6d 00 69 00   l.t.e.a. m.@.m.i.
00000240: 63 00 72 00 6f 00 73 00  6f 00 66 00 74 00 2e 00   c.r.o.s. o.f.t...
00000250: 63 00 6f 00 6d 00 00 00  6f 00 6c 00 74 00 65 00   c.o.m... o.l.t.e.
00000260: 61 00 6d 00 40 00 6d 00  69 00 63 00 72 00 6f 00   a.m.@.m. i.c.r.o.
00000270: 73 00 6f 00 66 00 74 00  2e 00 63 00 6f 00 6d 00   s.o.f.t. ..c.o.m.
00000280: 6f 00 6c 00 74 00 65 00  61 00 6d 00 40 00 6d 00   o.l.t.e. a.m.@.m.
00000290: 69 00 63 00 72 00 6f 00  73 00 6f 00 66 00 74 00   i.c.r.o. s.o.f.t.
000002a0: 2e 00 63 00 6f 00 6d 00  53 00 4d 00 54 00 50 00   ..c.o.m. S.M.T.P.
000002b0: 53 00 4d 00 54 00 50 00  57 00 65 00 6c 00 63 00   S.M.T.P. W.e.l.c.
000002c0: 6f 00 6d 00 65 00 20 00  74 00 6f 00 20 00 4d 00   o.m.e. . t.o. .M.
000002d0: 69 00 63 00 72 00 6f 00  73 00 6f 00 66 00 74 00   i.c.r.o. s.o.f.t.
000002e0: 20 00 4f 00 66 00 66 00  69 00 63 00 65 00 20 00    .O.f.f. i.c.e. .
000002f0: 4f 00 75 00 74 00 6c 00  6f 00 6f 00 6b 00 20 00   O.u.t.l. o.o.k. .
00000300: 32 00 30 00 30 00 33 00  31 00 31 00 2e 00 30 00   2.0.0.3. 1.1...0.
00000310: 57 00 65 00 6c 00 63 00  6f 00 6d 00 65 00 20 00   W.e.l.c. o.m.e. .
00000320: 74 00 6f 00 20 00 4d 00  69 00 63 00 72 00 6f 00   t.o. .M. i.c.r.o.
00000330: 73 00 6f 00 66 00 74 00  20 00 4f 00 66 00 66 00   s.o.f.t.  .O.f.f.
00000340: 69 00 63 00 65 00 20 00  4f 00 75 00 74 00 6c 00   i.c.e. . O.u.t.l.
00000350: 6f 00 6f 00 6b 00 20 00  32 00 30 00 30 00 33 00   o.o.k. . 2.0.0.3.
00000360: 4e 00 65 00 77 00 20 00  4f 00 75 00 74 00 6c 00   N.e.w. . O.u.t.l.
00000370: 6f 00 6f 00 6b 00 20 00  55 00 73 00 65 00 72 00   o.o.k. . U.s.e.r.
00000380: 20 00 09 00 54 00 68 00  61 00 6e 00 6b 00 20 00    ...T.h. a.n.k. .
00000390: 79 00 6f 00 75 00 20 00  66 00 6f 00 72 00 20 00   y.o.u. . f.o.r. .
000003a0: 75 00 73 00 69 00 6e 00  67 00 20 00 4d 00 69 00   u.s.i.n. g. .M.i.
000003b0: 63 00 72 00 6f 00 73 00  6f 00 66 00 74 00 ae 00   c.r.o.s. o.f.t...
000003c0: 20 00 4f 00 66 00 66 00  69 00 63 00 65 00 20 00    .O.f.f. i.c.e. .
000003d0: 4f 00 75 00 74 00 6c 00  6f 00 6f 00 6b 00 ae 00   O.u.t.l. o.o.k...
000003e0: 20 00 32 00 30 00 30 00  33 00 21 00 20 00 54 00    .2.0.0. 3.!. .T.
000003f0: 68 00 69 00 73 00 20 00  76 00 65 00 72 00 73 00   h.i.s. . v.e.r.s.
00000400: 69 00 6f 00 6e 00 20 00  6f 00 66 00 20 00 4f 00   i.o.n. . o.f. .O.
00000410: 75 00 74 00 6c 00 6f 00  6f 00 6b 00 20 00 69 00   u.t.l.o. o.k. .i.
00000420: 6e 00 63 00 6c 00 75 00  64 00 65 00 73 00 20 00   n.c.l.u. d.e.s. .
00000430: 6e 00 65 00 77 00 20 00  63 00 61 00 70 00 61 00   n.e.w. . c.a.p.a.
00000440: 62 00 69 00 6c 00 69 00  74 00 69 00 65 00 73 00   b.i.l.i. t.i.e.s.
00000450: 20 00 64 00 65 00 73 00  69 00 67 00 6e 00 65 00    .d.e.s. i.g.n.e.
00000460: 64 00 20 00 74 00 6f 00  20 00 68 00 65 00 6c 00   d. .t.o.  .h.e.l.
00000470: 70 00 20 00 79 00 6f 00  75 00 20 00 61 00 63 00   p. .y.o. u. .a.c.
00000480: 63 00 65 00 73 00 73 00  2c 00 20 00 70 00 72 00   c.e.s.s. ,. .p.r.
00000490: 69 00 6f 00 72 00 69 00  74 00 69 00 7a 00 65 00   i.o.r.i. t.i.z.e.
000004a0: 2c 00 20 00 61 00 6e 00  64 00 20 00 61 00 63 00   ,. .a.n. d. .a.c.
000004b0: 74 00 20 00 6f 00 6e 00  20 00 63 00 6f 00 6d 00   t. .o.n.  .c.o.m.
000004c0: 6d 00 75 00 6e 00 69 00  63 00 61 00 74 00 69 00   m.u.n.i. c.a.t.i.
000004d0: 6f 00 6e 00 73 00 20 00  61 00 6e 00 64 00 20 00   o.n.s. . a.n.d. .
000004e0: 69 00 6e 00 66 00 6f 00  72 00 6d 00 61 00 74 00   i.n.f.o. r.m.a.t.
000004f0: 69 00 6f 00 6e 00 20 00  73 00 6f 00 20 00 74 00   i.o.n. . s.o. .t.
00000500: 68 00 61 00 74 00 20 00  79 00 6f 00 75 00 20 00   h.a.t. . y.o.u. .
00000510: 6d 00 61 00 79 00 20 00  75 00 73 00 65 00 20 00   m.a.y. . u.s.e. .
00000520: 79 00 6f 00 75 00 72 00  20 00 74 00 69 00 6d 00   y.o.u.r.  .t.i.m.
00000530: 65 00 20 00 6d 00 6f 00  72 00 65 00 20 00 65 00   e. .m.o. r.e. .e.
00000540: 66 00 66 00 69 00 63 00  69 00 65 00 6e 00 74 00   f.f.i.c. i.e.n.t.
00000550: 6c 00 79 00 20 00 61 00  6e 00 64 00 20 00 6d 00   l.y. .a. n.d. .m.
00000560: 6f 00 72 00 65 00 20 00  65 00 61 00 73 00 69 00   o.r.e. . e.a.s.i.
00000570: 6c 00 79 00 20 00 6d 00  61 00 6e 00 61 00 15 00   l.y. .m. a.n.a...
00000580: 00 00 0c 00 14 00 24 00  04 01 14 01 1c 01 24 01   ......$. ......$.
00000590: 2c 01 34 01 56 01 78 01  e8 01 58 02 80 02 a8 02   ,.4.V.x. ..X.....
000005a0: b0 02 b8 02 08 03 10 03  60 03 80 03 7e 05         ........ `...~.
```

As you can see there is some readable text in the item table.

The 3rd and 4th byte of item table contain the value 0xec 0xbc. This is the table type definition as mentioned during the overview of the PFF format.

The 1st and 2nd byte (0x7e 0x05) contain the offset of the table index, in this case 1406.

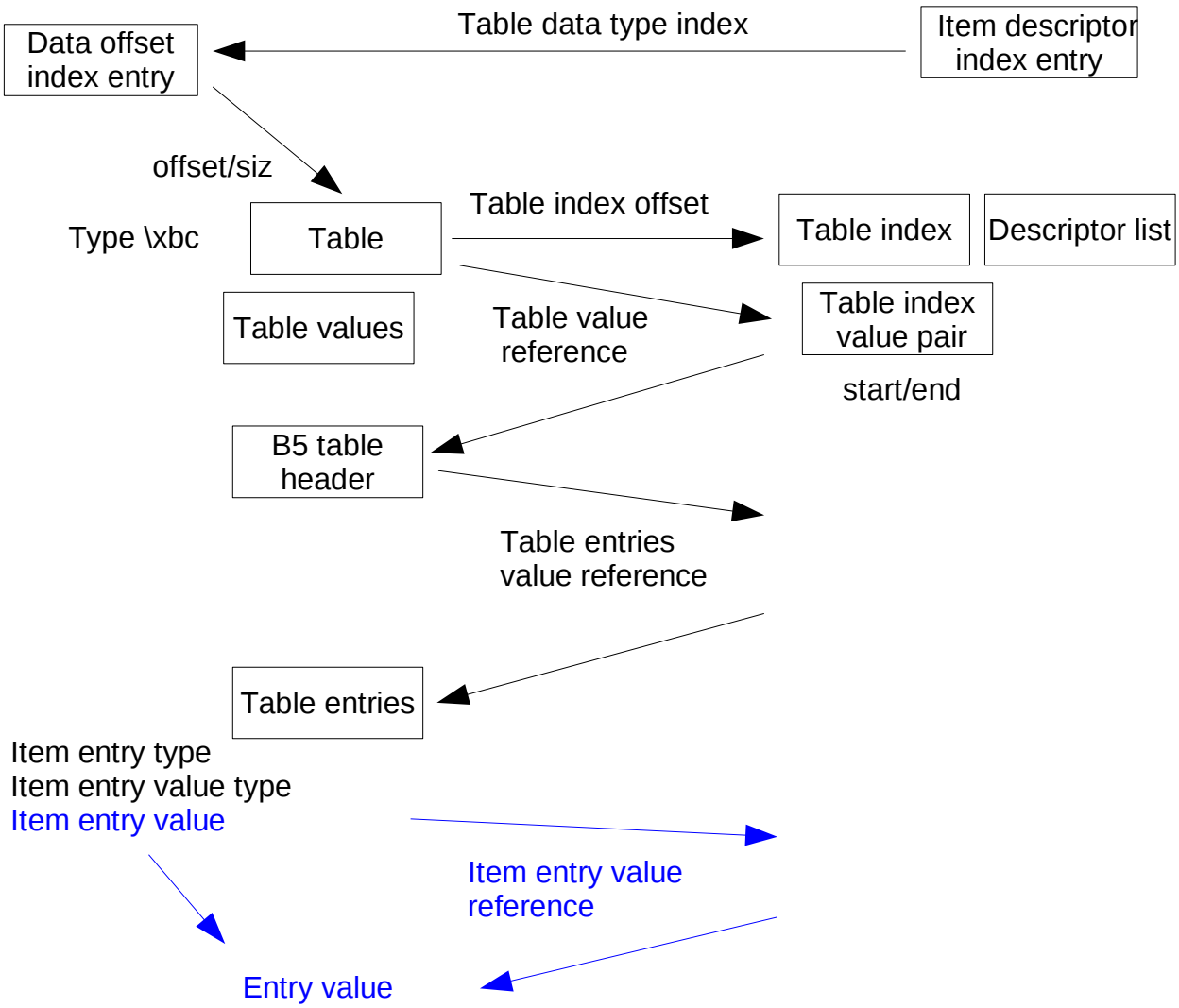
The table index consists of the following data:

```
00000570:                                                 15 00                 ..
00000580: 00 00 0c 00 14 00 24 00  04 01 14 01 1c 01 24 01   ......$. ......$.
00000590: 2c 01 34 01 56 01 78 01  e8 01 58 02 80 02 a8 02   ,.4.V.x. ..X.....
000005a0: b0 02 b8 02 08 03 10 03  60 03 80 03 7e 05         ........ `...~.
```

The index starts with the amount of items value 0x15 0x00 in this case 21. An index value consists of 2 values: the start and end offset. The end offset of a value is used as start offset of the next. In this case the first index value has a start offset 0x00 0x00 and end offset 0x0c 0x00. The second a start offset of 0x0c 0x00 and an end offset of 0x14 0x00, etc.

```
table index value: 000 offset            : 0 - 12
table index value: 001 offset            : 12 - 20
table index value: 002 offset            : 20 - 36
table index value: 003 offset            : 36 - 260
table index value: 004 offset            : 260 - 276
table index value: 005 offset            : 276 - 284
table index value: 006 offset            : 284 - 292
table index value: 007 offset            : 292 - 300
table index value: 008 offset            : 300 - 308
table index value: 009 offset            : 308 - 342
table index value: 010 offset            : 342 - 376
table index value: 011 offset            : 376 - 488
table index value: 012 offset            : 488 - 600
table index value: 013 offset            : 600 - 640
table index value: 014 offset            : 640 - 680
table index value: 015 offset            : 680 - 688
table index value: 016 offset            : 688 - 696
table index value: 017 offset            : 696 - 776
table index value: 018 offset            : 776 - 784
table index value: 019 offset            : 784 - 864
table index value: 020 offset            : 864 - 896
table index value: 021 offset            : 896 - 1406
```

Sometimes the end offset of the last table index value does not match the offset of the index values. It is unknown if the remaining byte has a function. In most cases it contains the value of 0x00.



The third value in the table at offset 4 (0x20 0x00 0x00 0x00) contains the table value reference, in this case 0x00000020. This value needs some elaboration. Reference values can be:

- within the table (internal table reference);
- outside the table (external table reference);
- within a certain table (data) array block (data array table reference). If the table is stored in a data array, every data array block has a table index with offsets relative to the start offset of the block.

You probably understand the nickname horrible references file format by now. For more details on how to determine a reference value, refer to [PFF08] or [LIBPST02].

In this case the table value reference refers to the second value in the table addressed by the index, namely from offset 12 (0x0c) to 20 (0x014).

```
00000000:                                       b5 02 06 00                ....
00000010: 60 00 00 00                                                      `...
```

For a table of type bc the table value reference refers to a b5 table header. The b5 table header is marked by the 0xb5 at its start. It is followed by the item entry record size 0x02 and the item entry value record size 0x06. The last 4 bytes contain the table entries reference (0x60 x00 0x00 0x00),

in this case 0x00000060. This refers to the fourth value in the table addressed by the index, namely from offset 36 (0x024) to 260 (0x0104).

```
00000020:             1a 00 1f 00  40 00 00 00 37 00 1f 00       .... @...7...
00000030: 20 02 00 00 39 00 40 00  00 01 00 00 41 00 02 01    ...9.@. ....A...
00000040: 80 01 00 00 42 00 1f 00  40 01 00 00 64 00 1f 00   ....B... @...d...
00000050: 00 02 00 00 65 00 1f 00  c0 01 00 00 70 00 1f 00   ....e... ....p...
00000060: 60 02 00 00 19 0c 02 01  60 01 00 00 1a 0c 1f 00   `....... `.......
00000070: 20 01 00 00 1e 0c 1f 00  e0 01 00 00 1f 0c 1f 00    ....... ........
00000080: a0 01 00 00 04 0e 1f 00  80 02 00 00 06 0e 40 00   ........ ......@.
00000090: e0 00 00 00 07 0e 03 00  10 00 00 00 08 0e 03 00   ........ ........
000000a0: 47 cc 00 00 00 10 1f 00  bf 81 00 00 13 10 02 01   G....... ........
000000b0: 9f 81 00 00 07 30 40 00  a0 00 00 00 08 30 40 00   .....0@. .....0@.
000000c0: c0 00 00 00 0b 30 02 01  80 00 00 00 de 3f 03 00   .....0.. .....?..
000000d0: e4 04 00 00 04 66 03 00  c7 59 00 00 10 66 03 00   .....f.. .Y...f..
000000e0: 4e 01 00 00 19 66 1f 00  a0 02 00 00 05 80 03 00   N....f.. ........
000000f0: 98 c3 01 00 06 80 1f 00  40 02 00 00 14 80 0b 00   ........ @.......
00000100: 01 00 00 00                                       ....
```

The table entry values consist of multiple table item entries. The format of the entries is determined by the item entry record size and the item entry value record size in the b5 table header. In this case the format of the table item entry is:

- 2 bytes item entry type
- 2 bytes item entry value type
- 4 bytes item entry value

In the following example the value is directly stored in the entry.

```
entry: 015 item entry type                : 0x0e08 (PR_MESSAGE_SIZE)
015 item entry value type                 : 0x0003 (PT_LONG)
015 item entry value                      : 0x0000cc47
```

In the following example the value is stored in another table entry.

```
entry: 013 item entry type                : 0x0e06 (PR_MESSAGE_DELIVERY_TIME)
entry: 013 item entry value type          : 0x0040 (PT_SYSTIME)
entry: 013 item entry value               : at offset: 292 of size: 8
Filetime  : 10:29:35 14/01/2009 UTC
```

Another example where the value is stored outside the table.

```
entry: 016 item entry type                : 0x1000 (PR_BODY)
entry: 016 item entry value type          : 0x001f (PT_UNICODE)
entry: 016 item entry value reference     : 33215
```

In this case the item entry value contains an external reference. This reference refers to an entry in the local descriptor list.

```
element: 15 identifier                    : 33215
element: 15 data identifier               : 990
element: 15 list identifier               : 0
```

The corresponding data can be found by looking up the data identifier in the data structure index.

The corresponding data structure contains the following data:

```
00000000: 01 01 02 00 ac 3f 00 00  d8 03 00 00 00 00 00 00   .....?.. ........
00000010: e0 03 00 00 00 00 00 00                           ........
```

The data array structure always begins with the values 0x01 0x01, followed by the amount of array entries (0x02 0x00), which is 2 in this case. After the amount of entries there is a 4 byte (0xd8 0x03 0x00 0x00) total size value of 16300. This value is followed by the array entries after a 4 byte padding.

In this case the data array contains the Microsoft Outlook 2003 welcome message.

```
Thank you for using Microsoft® Office Outlook® 2003! ...
```

All the individual array data structure index identifiers refer to the blocks that make up the entire message.

In general the following PFF e-mail item entry values are not stored within the table:
- the transport message headers (SMTP headers);
- the plain text body;
- the RTF body.

The MAPI defines entry types with values of 0x8000 to 0xfffe as named properties. A PFF contains the special item 'name-to-identifier map' to find the name of these properties. This name-to-identifier map will be explained after the attachments.

2.3. The attachments

The attachments are stored in a different manner. First of all the local descriptor list of the e-mail item contains a fixed value identifier for the attachments table, which is 1649.

```
element: 01 identifier                  : 1649
element: 01 data identifier             : 972
element: 01 list identifier             : 978
```

The data structure of the attachments table can be found using the data (structure index) identifier.

The local descriptor list:

```
00000000: 02 00 01 00 00 00 00 00  3f 00 00 00 00 00 00 00   ........ ?.......
00000010: c8 03 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

```
element: 01 identifier                  : 63
element: 01 data identifier             : 968
element: 01 list identifier             : 0
```

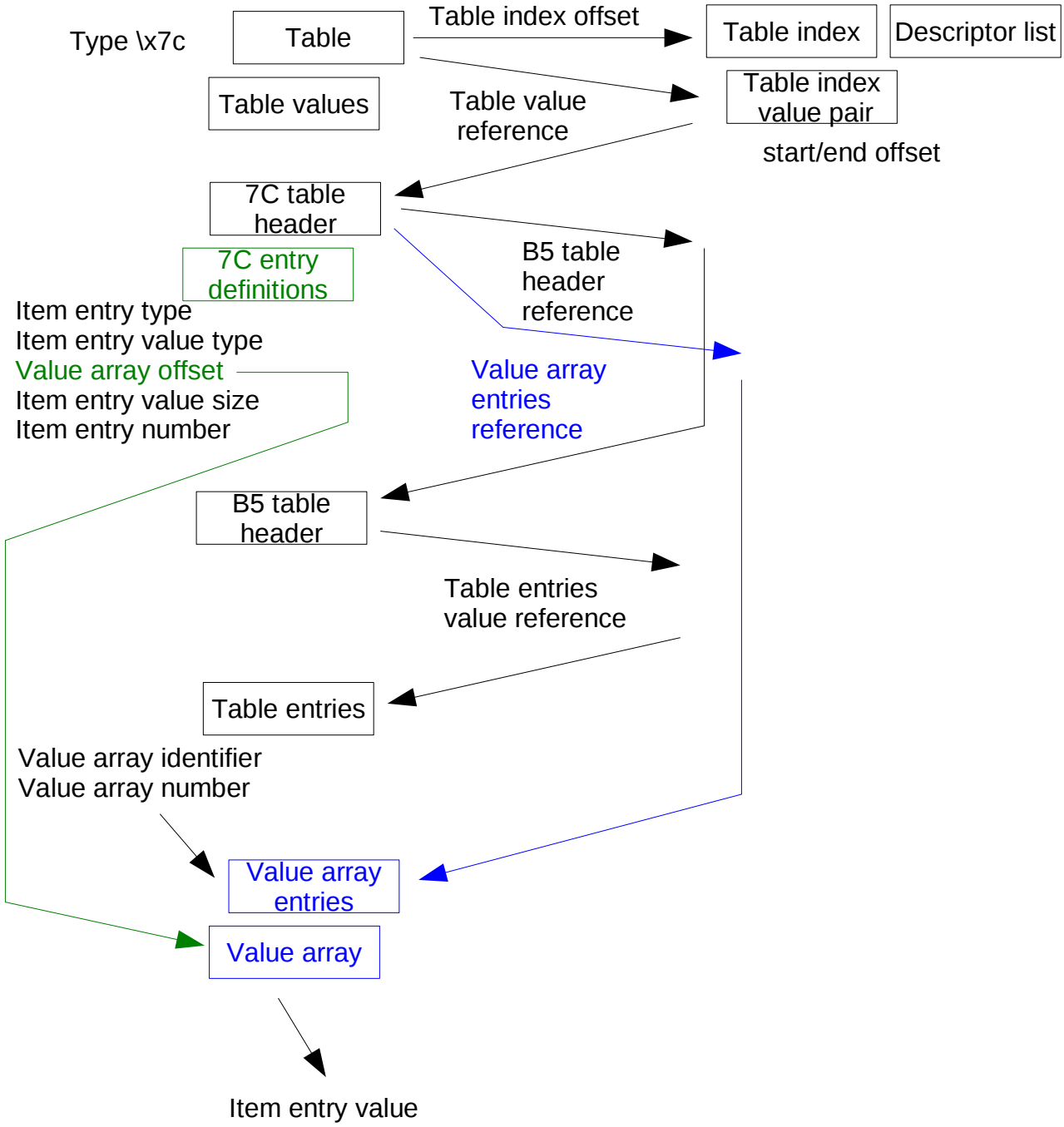
The attachments table is a 7c table which contains multiple item sets, one for every attachment.

```
00000000: d4 03 ec 7c 40 00 00 00  00 00 00 00 b5 04 04 00   ...|@... ........
00000010: 60 00 00 00 7c 19 64 00  64 00 66 00 6a 00 20 00   `...|.d. d.f.j. .
00000020: 00 00 3f 00 00 00 00 00  00 00 03 00 20 0e 0c 00   ..?..... .... ...
00000030: 04 03 1f 00 01 30 2c 00  04 0b 1f 00 03 37 28 00   .....0,. .....7(.
```

```
00000040: 04 0a 1f 00 04 37 14 00  04 05 03 00 05 37 10 00   .....7.. .....7..
00000050: 04 04 1f 00 07 37 24 00  04 09 1f 00 08 37 20 00   .....7$. .....7 .
00000060: 04 08 02 01 0a 37 18 00  04 06 03 00 0b 37 08 00   .....7.. .....7..
00000070: 04 02 1f 00 0d 37 1c 00  04 07 1f 00 0e 37 40 00   .....7.. .....7@.
00000080: 04 10 02 01 0f 37 30 00  04 0c 1f 00 11 37 34 00   .....70. .....74.
00000090: 04 0d 1f 00 12 37 3c 00  04 0f 1f 00 13 37 38 00   .....7<. .....78.
000000a0: 04 0e 03 00 14 37 60 00  04 18 03 00 f2 67 00 00   .....7`. .....g..
000000b0: 04 00 03 00 f3 67 04 00  04 01 03 00 09 69 44 00   .....g.. .....iD.
000000c0: 04 11 03 00 fa 7f 5c 00  04 15 40 00 fb 7f 4c 00   ......\. ..@...L.
000000d0: 08 13 40 00 fc 7f 54 00  08 14 03 00 fd 7f 48 00   ..@...T. ......H.
000000e0: 04 12 0b 00 fe 7f 64 00  01 16 0b 00 ff 7f 65 00   ......d. ......e.
000000f0: 01 17 25 80 00 00 00 00  00 00 45 80 00 00 01 00   ..%..... ..E.....
00000100: 00 00 65 80 00 00 02 00  00 00 85 80 00 00 03 00   ..e..... ........
00000110: 00 00 a5 80 00 00 04 00  00 00 c5 80 00 00 05 00   ........ ........
00000120: 00 00 e5 80 00 00 06 00  00 00 05 81 00 00 07 00   ........ ........
00000130: 00 00 25 81 00 00 08 00  00 00 45 81 00 00 09 00   ..%..... ..E.....
00000140: 00 00 65 81 00 00 0a 00  00 00 31 00 32 00 2e 00   ..e..... ..1.2...
00000150: 6a 00 70 00 67 00 31 00  2e 00 6a 00 70 00 67 00   j.p.g.1. ..j.p.g.
00000160: 2e 00 6a 00 70 00 67 00  31 00 2e 00 6a 00 70 00   ..j.p.g. 1...j.p.
00000170: 67 00 69 00 6d 00 61 00  67 00 65 00 2f 00 6a 00   g.i.m.a. g.e./.j.
00000180: 70 00 65 00 67 00 31 00  2e 00 6a 00 70 00 67 00   p.e.g.1. ..j.p.g.
00000190: 32 00 2e 00 6a 00 70 00  67 00 2e 00 6a 00 70 00   2...j.p. g...j.p.
000001a0: 67 00 32 00 2e 00 6a 00  70 00 67 00 69 00 6d 00   g.2...j. p.g.i.m.
000001b0: 61 00 67 00 65 00 2f 00  6a 00 70 00 65 00 67 00   a.g.e./. j.p.e.g.
000001c0: 32 00 2e 00 6a 00 70 00  67 00 33 00 2e 00 6a 00   2...j.p. g.3...j.
000001d0: 70 00 67 00 2e 00 6a 00  70 00 67 00 33 00 2e 00   p.g...j. p.g.3...
000001e0: 6a 00 70 00 67 00 69 00  6d 00 61 00 67 00 65 00   j.p.g.i. m.a.g.e.
000001f0: 2f 00 6a 00 70 00 65 00  67 00 33 00 2e 00 6a 00   /.j.p.e. g.3...j.
00000200: 70 00 67 00 34 00 2e 00  6a 00 70 00 67 00 2e 00   p.g.4... j.p.g...
00000210: 6a 00 70 00 67 00 34 00  2e 00 6a 00 70 00 67 00   j.p.g.4. ..j.p.g.
00000220: 69 00 6d 00 61 00 67 00  65 00 2f 00 6a 00 70 00   i.m.a.g. e./.j.p.
00000230: 65 00 67 00 34 00 2e 00  6a 00 70 00 67 00 35 00   e.g.4... j.p.g.5.
00000240: 2e 00 6a 00 70 00 67 00  2e 00 6a 00 70 00 67 00   ..j.p.g. ..j.p.g.
00000250: 35 00 2e 00 6a 00 70 00  67 00 69 00 6d 00 61 00   5...j.p. g.i.m.a.
00000260: 67 00 65 00 2f 00 6a 00  70 00 65 00 67 00 35 00   g.e./.j. p.e.g.5.
00000270: 2e 00 6a 00 70 00 67 00  36 00 2e 00 6a 00 70 00   ..j.p.g. 6...j.p.
00000280: 67 00 2e 00 6a 00 70 00  67 00 36 00 2e 00 6a 00   g...j.p. g.6...j.
00000290: 70 00 67 00 69 00 6d 00  61 00 67 00 65 00 2f 00   p.g.i.m. a.g.e./.
000002a0: 6a 00 70 00 65 00 67 00  36 00 2e 00 6a 00 70 00   j.p.e.g. 6...j.p.
000002b0: 67 00 37 00 2e 00 6a 00  70 00 67 00 2e 00 6a 00   g.7...j. p.g...j.
000002c0: 70 00 67 00 37 00 2e 00  6a 00 70 00 67 00 69 00   p.g.7... j.p.g.i.
000002d0: 6d 00 61 00 67 00 65 00  2f 00 6a 00 70 00 65 00   m.a.g.e. /.j.p.e.
000002e0: 67 00 37 00 2e 00 6a 00  70 00 67 00 38 00 2e 00   g.7...j. p.g.8...
000002f0: 6a 00 70 00 67 00 2e 00  6a 00 70 00 67 00 38 00   j.p.g... j.p.g.8.
00000300: 2e 00 6a 00 70 00 67 00  69 00 6d 00 61 00 67 00   ..j.p.g. i.m.a.g.
00000310: 65 00 2f 00 6a 00 70 00  65 00 67 00 38 00 2e 00   e./.j.p. e.g.8...
00000320: 6a 00 70 00 67 00 39 00  2e 00 6a 00 70 00 67 00   j.p.g.9. ..j.p.g.
00000330: 2e 00 6a 00 70 00 67 00  39 00 2e 00 6a 00 70 00   ..j.p.g. 9...j.p.
00000340: 67 00 69 00 6d 00 61 00  67 00 65 00 2f 00 6a 00   g.i.m.a. g.e./.j.
00000350: 70 00 65 00 67 00 39 00  2e 00 6a 00 70 00 67 00   p.e.g.9. ..j.p.g.
00000360: 31 00 30 00 2e 00 6a 00  70 00 67 00 2e 00 6a 00   1.0...j. p.g...j.
00000370: 70 00 67 00 31 00 30 00  2e 00 6a 00 70 00 67 00   p.g.1.0. ..j.p.g.
00000380: 69 00 6d 00 61 00 67 00  65 00 2f 00 6a 00 70 00   i.m.a.g. e./.j.p.
00000390: 65 00 67 00 31 00 30 00  2e 00 6a 00 70 00 67 00   e.g.1.0. ..j.p.g.
000003a0: 2e 00 6a 00 70 00 67 00  31 00 32 00 2e 00 6a 00   ..j.p.g. 1.2...j.
000003b0: 70 00 67 00 69 00 6d 00  61 00 67 00 65 00 2f 00   p.g.i.m. a.g.e./.
000003c0: 6a 00 70 00 65 00 67 00  31 00 32 00 2e 00 6a 00   j.p.e.g. 1.2...j.
000003d0: 70 00 67 00 3a 00 00 00  0c 00 14 00 f2 00 4a 01   p.g.:... ......J.
000003e0: 56 01 60 01 68 01 72 01  86 01 90 01 9a 01 a2 01   V.`.h.r. ........
000003f0: ac 01 c0 01 ca 01 d4 01  dc 01 e6 01 fa 01 04 02   ........ ........
```

```
00000400: 0e 02 16 02 20 02 34 02  3e 02 48 02 50 02 5a 02   .... .4. >.H.P.Z.
00000410: 6e 02 78 02 82 02 8a 02  94 02 a8 02 b2 02 bc 02   n.x..... ........
00000420: c4 02 ce 02 e2 02 ec 02  f6 02 fe 02 08 03 1c 03   ........ ........
00000430: 26 03 30 03 38 03 42 03  56 03 60 03 6c 03 74 03   &.0.8.B. V.`.l.t.
00000440: 80 03 94 03 a0 03 a8 03  b4 03 c8 03 d4 03         ........ ......
```

To contain multiple sets the 7c table uses entry definitions and a values array. The diagram below outlines their relationship.



The attachments table defines the file names of the attachments and their local descriptor identifier.

```
table set: 000 entry: 000 item entry type                    : 0x67f2
(Descriptor identifier)
table set: 000 entry: 000 item entry value type              : 0x0003
(PT_LONG)
table set: 000 entry: 000 item entry value                   : 0x00008025

table set: 001 entry: 000 item entry type                    : 0x67f2
(Descriptor identifier)
table set: 001 entry: 000 item entry value type              : 0x0003
(PT_LONG)
table set: 001 entry: 000 item entry value                   : 0x00008045
```

This attachments table has a local descriptor list. However often the attachments table does not have a local descriptor list. In either case the attachment descriptor identifiers actual refer to entries within the local descriptor list of its parent e-mail item. For the first set this is the descriptor identifier 32805 (0x08025).

```
element: 03 identifier                  : 32805
element: 03 data identifier             : 504
element: 03 list identifier             : 518
```

In this case the descriptor list entry refers to an attachment item consisting of both a table (data) and list identifier.

The attachment local descriptor list contains the following data:

```
00000000: 02 00 01 00 00 00 00 00  3f 80 00 00 00 00 00 00   ........ ?.......
00000010: f0 01 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

```
element: 01 identifier                  : 32831
element: 01 data identifier             : 496
element: 01 list identifier             : 0
```

And the attachment table is composed of the following data:

```
00000000: ee 00 ec bc 20 00 00 00  00 00 00 00 b5 02 06 00   .... ... ........
00000010: 40 00 00 00 20 0e 03 00  15 08 00 00 01 30 1f 00   @... ... .....0..
00000020: a0 00 00 00 01 37 02 01  3f 80 00 00 02 37 02 01   .....7.. ?....7..
00000030: 00 00 00 00 03 37 1f 00  20 01 00 00 04 37 1f 00   .....7..  ....7..
00000040: 00 00 00 00 05 37 03 00  01 00 00 00 07 37 1f 00   .....7.. .....7..
00000050: 00 01 00 00 0b 37 03 00  ff ff ff ff 0e 37 1f 00   .....7.. .....7..
00000060: e0 00 00 00 13 37 1f 00  c0 00 00 00 14 37 03 00   .....7.. .....7..
00000070: 04 00 00 00 fa 7f 03 00  00 00 00 00 fb 7f 40 00   ........ ......@.
00000080: 60 00 00 00 fc 7f 40 00  80 00 00 00 fd 7f 03 00   `.....@. ........
00000090: 00 00 00 00 fe 7f 0b 00  00 00 00 00 ff 7f 0b 00   ........ ........
000000a0: 00 00 00 00 00 40 dd a3  57 45 b3 0c 00 40 dd a3   .....@.. WE...@..
000000b0: 57 45 b3 0c 31 00 2e 00  6a 00 70 00 67 00 31 00   WE..1... j.p.g.1.
000000c0: 2e 00 6a 00 70 00 67 00  69 00 6d 00 61 00 67 00   ..j.p.g. i.m.a.g.
000000d0: 65 00 2f 00 6a 00 70 00  65 00 67 00 31 00 2e 00   e./.j.p. e.g.1...
000000e0: 6a 00 70 00 67 00 2e 00  6a 00 70 00 67 00 09 00   j.p.g... j.p.g...
000000f0: 00 00 0c 00 14 00 a4 00  ac 00 b4 00 be 00 c8 00   ........ ........
00000100: dc 00 e6 00 ee 00                                 ......
```

Some of the item entry values in the table above are provided below in a more readable way.

```
entry: 000 item entry type                : 0x0e20 (PR_ATTACH_SIZE)
entry: 000 item entry value type          : 0x0003 (PT_LONG)
entry: 000 item entry value               : 0x00000815

entry: 001 item entry type                : 0x3001 (PR_DISPLAY_NAME)
entry: 001 item entry value type          : 0x001f (PT_UNICODE)
entry: 001 item entry value               : at offset: 180 of size: 10
Unicode string       : 1.jpg

entry: 002 item entry type                : 0x3701 (PR_ATTACH_DATA_BIN)
entry: 002 item entry value type          : 0x0102 (PT_BINARY)
entry: 002 item entry value reference     : 32831
```

In this case the attachment contains a reference to binary data, meaning that the actual attachment data is stored in the PFF. Some attachments can contain embedded objects which can refer to internal items, i.e. in case of bounced e-mails.

You might have guessed it by now: the attachment binary data reference is a data structure index identifier referring to attachment data. In this case the first part of a JPEG.

```
00000000: ff d8 ff e0 00 10 4a 46  49 46 00 01 01 01 00 48   ......JF IF.....H
00000010: 00 48 00 00 ff db 00 43  00 01 01 01 01 01 01 01   .H.....C ........
00000020: 01 01 01 01 01 01 01 01  01 01 01 01 01 01 01 01   ........ ........

...
```

2.4. The name-to-identifier map

The MAPI defines entry types with values of 0x8000 to 0xfffe as named properties. In the PFF some of these named properties are:

- mapped to other properties
- mapped to names

The following property 0x8014 is mapped to another property 0x8514.

```
entry type: 0x8014 maps to: 0x8514
entry: 027 item entry type                : 0x8014 (Unknown)
entry: 027 item entry value type          : 0x000b (PT_BOOLEAN)
entry: 027 item entry value               : 0x00000001
```

However the contents of property 0x8514 is unknown.

The following property 0x8021 is mapped to the name “x-originating-ip”

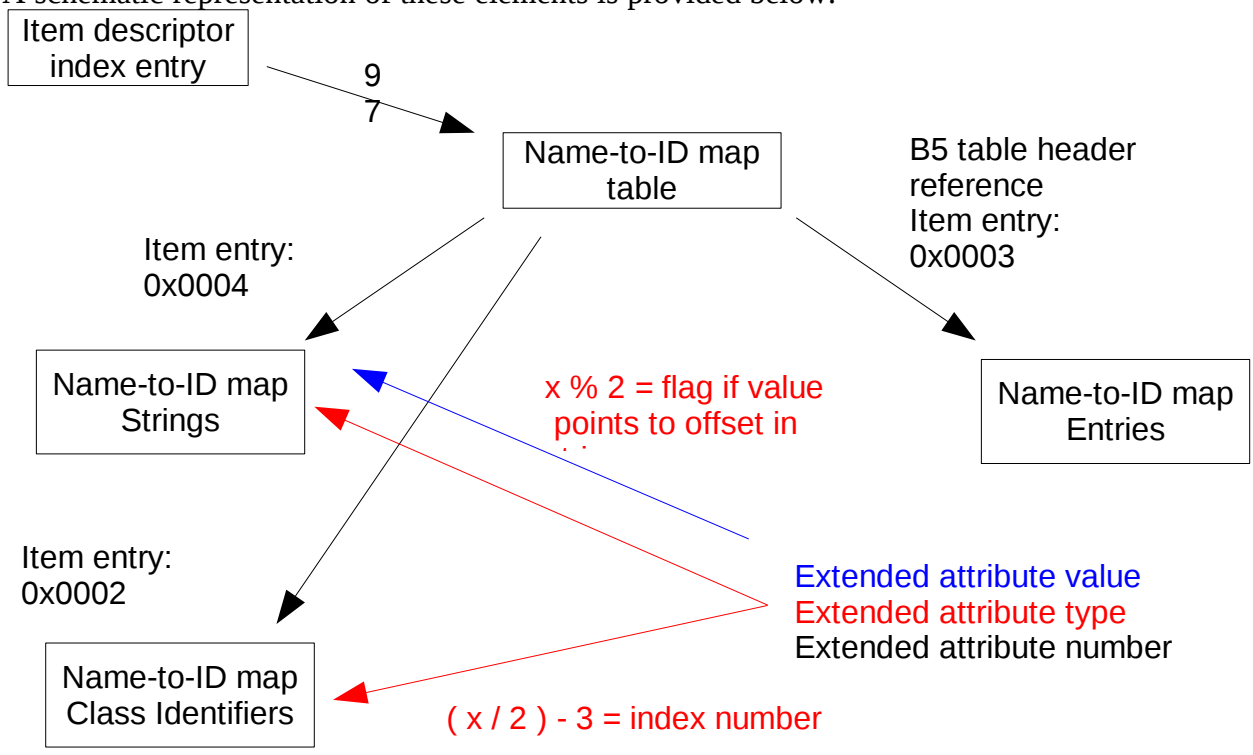
```
entry: 028 item entry type                : 0x8021 (x-originating-ip)
entry: 028 item entry value type          : 0x001f (PT_UNICODE)
entry: 028 item entry value               : at offset: 180 of size: 20
Unicode string       : 127.0.0.1
```

A PFF contains the special item 'name-to-identifier map' to find the corresponding number or name of these mapped properties. The name-to-identifier map item is identified by descriptor identifier 97. The name-to-identifier map item contains several elements that make up the name-to-identifier map, these are:

- an array of COM interface class identifiers (GUID)

- an array of the name-to-id map entries
- an array of the name-to-id map strings
- multiple validation entries

A schematic representation of these elements is provided below.



Some of the extended attribute types seem to refer to default classes like 0x05 to the Public Strings Class.

3. Recovering PFF items

Recovering PFF items is a fairly straight-forward process.

1. Determine which blocks contain unallocated index nodes.
2. Determine if an index entry is recoverable.
    1. Check if the index entry does not already exists.
    2. For a data structure index entry, determine if the file offset range it refers to is unallocated.
    3. Check if the index entry was already recovered.
    4. For an item descriptor index entry that was already recovered check if the parent item identifier exists. If the parent item identifier exists this means that the index entry was deleted more recent that an item descriptor without an existing parent identifier.
3. For a recovered item descriptor index entry check if the data structures identifiers were also recovered.

I.e. pffrecover uses this approach to recover deleted items.

```
$ pffrecover deleted.pst
pffrecover 20081217 (libpff 20081217, libuna 20081011)

Opening file.
```

```
Recovering items.
10 items recovered.
Exporting recovered items.
...
Exporting e-mail item 8 out of 10.
Exporting attachment 1 out of 1.
...
Recovery completed.
```

For item descriptor index entries that contain an existing parent identifier, the item can be put in the item folder hierarchy.

4. Password protection

Microsoft Outlook allows users to set a password on their PST files. This password is stored in the 'Message Store' PFF item as a weak 32-bit Cyclic Redundancy Check (CRC32).

I.e. a PFF without a password.

```
item entry type          : 0x67ff (PR_PST_PASSWORD)
item entry value type    : 0x0003 (PT_LONG)
item entry value         : 0x00000000
```

I.e. a PFF with a password.

```
item entry type          : 0x67ff (PR_PST_PASSWORD)
item entry value type    : 0x0003 (PT_LONG)
item entry value         : 0x50e099bc
```

The weak CRC32 is not suited to store a password hash, because it is to easy to generate a collision. This means that the password can be easily cracked.

But it gets worse; PFF does nothing with the password other than store its weak CRC32. So none of the data in a PFF is actually protected by the password. Good news for forensic analysis.

I.e. pffexport exports items in password protected PST file without supplying any password.

```
$ pffinfo password.pst
pffinfo 20081217 (libpff 20081217, libuna 20081011)

Personal Folder File information:
        File size:              1795072 bytes
        File type:              64-bit
        Encryption type:        compressible

...

$ pffexport password.pst
pffexport 20081217 (libpff 20081217, libuna 20081011)

Opening file.
Exporting items.
Exporting folder item 1 out of 32.
...
Exporting e-mail item 83 out of 115.
```

```
Exporting attachment 1 out of 2.
Exporting attachment 2 out of 2.
...
Export completed.
```

If the password weak CRC32 is set to a 0 value for a PFF containing a password, the password protection is removed for this application conforming to this protection scheme.

Appendix A. References

[PFF08]
Title: Personal Folder File (PFF) format specification
Author(s): Joachim Metz
URL: https://libpff.sourceforge.net/

[LIBPST02]
Title: libpst Utilities
Author(s): David Smith, Joe Nahmias, Brad Hards, Carl Byington
URL: http://www.five-ten-sg.com/libpst/

Appendix B. GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy

that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections

2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only

one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include

a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.