Personal Folder File (PFF) file format specification

Analysis of the PFF format

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

Summary

PFF is short for Personal Folder File and is mainly used by Microsoft Outlook to store e-mails, appointments, contacts, tasks, etc. This specification is based on the work by libpst [SMITH02] and was complimented by reverse engineering of the file format.

This document is intended as a working document for the PFF specification. Which should allow existing Open Source forensic tooling to be able to process this file type.

Document information

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

Abstract: This document contains information about the Personal Folder File 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
0.0.1	J.B. Metz	Jun 1, 2008	Initial version based on earlier notes.
0.0.2	J.B. Metz	Jun 7, 2008	Added information about LZFu compression and arrays.
0.0.3	J.B. Metz	Jul 7, 2008	Added information about allocation tables.
0.0.4	J.B. Metz	Oct 18, 2008	Updated for initial release.
0.0.5	J.B. Metz	Oct 19, 2008	Added reference about RTF compression.
0.0.6	J.B. Metz	Oct 21, 2008	Addition about attachments of type embedded object.
0.0.7	J.B. Metz	Oct 24, 2008	Addition about 0x85 0x85 index node type.
0.0.8	J.B. Metz	Oct 28, 2008	Added information about descriptor list type.
0.0.9	J.B. Metz	Nov 27, 2008	Added information about name-to-id map.
0.0.10	J.B. Metz	Dec 6, 2008	Additional information about the header.
0.0.11	J.B. Metz	Dec 7, 2008	Additional information about the header and item values types (property types).
0.0.12	J.B. Metz	Dec 8, 2008	Additional information about item types (property names/identifiers) and the name-to-id map.
0.0.13	J.B. Metz	Dec 9, 2008	Additional information about the file header and the item types.
0.0.14	J.B. Metz	Dec 10, 2008 Dec 11, 2008 Dec 12, 2008 Dec 13, 2008	Additional information about the item types.
0.0.15	J.B. Metz	Jan 15, 2009	Additional information about the allocation full maps.
0.0.16	J.B. Metz	Jan 18, 2009	Additional information about HTML e-mail body type.
0.0.17	J.B. Metz	Jan 25, 2009	Additional information about name-to-id map.
0.0.18	J.B. Metz	Jan 29, 2009	Move MAPI definitions to separate document.

Table of Contents

1. Overview

The PFF (Personal Folder File) and OFF (Offline Folder File) format is used to store Microsoft Outlook e-mails, appointments and contacts. The OST (Offline Storage Table), PAB (Personal Address Book) and PST (Personal Storage Table) file format consist of the PFF format.

A PFF consist of the following distinguishable elements:
- file header
- file header data
- index branch node
- index leaf node
- (file) offset index
- (item) descriptor index
- list type
- table type

The PFF format uses little endian.

Certain elements of the PFF format are related to the Microsoft (Office) Outlook Messaging API (MAPI).

1.1. Test version

The following version of programs were used to test the information within this document:
- Microsoft Outlook 2000, 2003, 2007
- Exmerge
- Scanpst

2. File header

The file header common to both the 32-bit and 64-bit pst file format consists of 24 bytes.

offset	**size**	**value**	**description**
0	4	\x21\x42\x44\x4e	The signature (magic identifier)
4	4		A weak CRC32 of the following 471 bytes In 64-bit PST files this CRC seems to be ignored because of the CRC at the end of the file header data at offset 524.
8	2	\x41\x42 (AB) \x53\x4d (SM) \x53\x4f (SO)	The content type (client signature) AB is used for PAB files SM is used for PST files SO is used for OST files
10	2	\x0e\x00 \x17\x00	The data version (NDB version) \x0e\x00 for a 32-bit pst file \x15\x00 for a 64-bit pst file \x17\x00 for a 64-bit pst file

offset	size	value	description
			Does NDB refer to node/network (hierarchical) database? \x15\x00 was found in a 64-bit pst file created by Visual Recovery
12	2		Content version (Client version) Unknown use
14	1	\x01	Creation Platform Unknown use \x02 found in scanpst recovered pst
15	1	\x01	Access Platform Unknown use \x02 found in scanpst recovered pst
16	4		Unknown value (mostly empty)
20	4		Unknown value (mostly empty)

2.1. The 32-bit header data

The 32-bit header data is 455 bytes of size.

offset	size	value	description
24	4		Unknown value (bidNextB) Changes consecutive created pst files
28	4		Unknown value (bidNextP) (value looks similar to back pointer always seems to be 1 increment of the largest index back pointer)
32	4		Unknown value Changes consecutive created pst files
36	128 (32 x 4)		Unknown value (NID high-water marks) NID => Node ID ? Seems to be the equivalent of the items identifiers type=n is the number in the array
164	4		Empty values
168	4		Total file size
172	4		Unknown value (libAMapLast) Points to the last data allocation table
176	4		Unknown value Does not change in consecutive created pst files
180	4		Unknown value Does not change in consecutive created

offset	size	value	description
			pst files
184	4		The descriptor index back pointer the value that should appear in the parent offset of the root node of the descriptor index b-tree
188	4		The descriptor index file offset File offset of the the of the descriptor index b-tree
192	4		The (file) offset index back pointer the value that should appear in the parent offset of the root node of the (file) offset index b-tree
196	4		The (file) offset index file offset File offset of the the of the (file) offset index b-tree
200	4		Unknown value In older formats (fAMapValid) for which values of 00 and 01 are allowed In newer formats (cARVec) for which signed values less equal to 2045 are allowed some kind of vector? following values be related with this value?
204	128		Unknown values (AMap) byte array pointing to the data allocation table starting at 0x4400 every byte represents 0x3e000 bytes scanpst checks if this map contains a minimum of csFree. Probably the minimum amount of free blocks in an data block allocation table.
332	128		Index node allocation table full map (rgbFP) bit array containing an overview of which index node allocation table are full or not. The first bit refers to the index node allocation table at offset 0x4600. Every bit represents 0x1f0000 bytes, which is the amount of bytes defined by the index node allocation table. For values smaller than the file size this a bit signifies if the corresponding index

offset	size	value	description
			node allocation table (Pmap) is: 0 => not full 1 => full
460	1	\x80	Senitinal (bSentinal)
461	1		Encryption type (Encryption method) \x00 is no encryption \x01 encryption type compressible \x02 encryption type high
462	2		Unknown value In older formats (rgbReserved Index) which is 17 bytes of size In newer formats it contains values from offset 474
464	15		Unknown values

Data after file header data probably extended data for AMap

offset	size	value	description
479	33		Empty values
512	4		Unknown value Changes consecutive created pst files
516	4		Unknown value Changes consecutive created pst files
520	4		Unknown value Does not change in consecutive created pst files
524	4		Unknown value Changes consecutive created pst files
528	16880		Empty values

Although encryption type is none some pff files contain compressible encrypted data? Found in exmerged pst without message store and extended tables. However this does not seem to be normal behavior.

2.2. The 64-bit header data

The 64-bit header data is 500 bytes of size

offset	size	value	description
24	8		Unknown value (or 2 x 4 byte values?)
32	8		Unknown value (bidNextP) (value looks similar to back pointer

offset	size	value	description
			always seems to be 1 increment of the largest index back pointer)
40	4		Unknown value
44	128 (32 x 4)		Unknown value (NID high-water marks) NID => Node ID ? Seems to be the equivalent of the items identifiers type=n is the number in the array
172	12		Empty values
184	8		Total file size
192	8		Unknown value (libAMapLast) Points to the last data allocation table
200	8		Unknown value (offset? To what?)
208	8		Unknown values file offset to an 8080 index node
216	8		The descriptor index back pointer the value that should appear in the parent offset of the root node of the descriptor index b-tree
224	8		The descriptor index file offset File offset of the the of the descriptor index b-tree
232	8		The (file) offset index back pointer the value that should appear in the parent offset of the root node of the (file) offset index b-tree
240	8		The (file) offset index file offset File offset of the the of the (file) offset index b-tree
248	4		Unknown value (cARVec) for which signed values less equal to 1022 are allowed some kind of vector? could the following values be related with this value?
252	4		Empty value
256	128		Unknown values (AMap) byte array pointing to the data allocation table starting at 0x4400 every byte represents 0x3e000 bytes scanpst checks if this map contains a minimum of csFree
384	128		Index node allocation table full map (rgbFP)

offset	size	value	description
			bit array containing an overview of which index node allocation table are full or not. The first bit refers to the index node allocation table at offset 0x4600. Every bit represents 0x1f0000 bytes, which is the amount of bytes defined by the index node allocation table. For values smaller than the file size this a bit signifies if the corresponding index node allocation table (Pmap) is: 0 => not full 1 => full
512	1	\x80	Senitinal (bSentinal)
513	1		Encryption type (Encryption method) \x00 is no encryption \x01 encryption type compressible \x02 encryption type high
514	2		Unknown values
516	8		Unknown values
524	4		A weak CRC32 of the previous 516 bytes

Data after file header data probably extended data for AMap

offset	size	value	description
528	16		Empty values
538	8		Unknown value
546	8		Unknown value
552	...		

3. The allocation table

The PFF contains several allocation tables. These tables are used to describe what parts of the PFF are in use and free.

3.1. The 32-bit allocation table

The 32-bit allocation is 512 bytes of size

offset	size	value	description
0	4		Emtpy values

offset	size	value	description
4	496		The allocation table each bit represents a certain amount of bytes (block). A value of 1 means that the block is allocated, 0 if not
500	2		Type indicator
502	2		Emtpy values
504	4		The allocation table offset
508	4		A weak CRC32 of the 496 bytes of the table data

The allocation table at offset 0x4400 with type indicator 0x84 0x84 seems to address 64 byte blocks. Where the first bit refers to offset 0x4400. They are used for the data allocation (PMap).

The allocation table at offset 0x4600 with type indicator 0x83 0x83 seems to address 512 byte blocks. Where the first bit refers to offset 0x4400. They are used for the index node allocation.

3.2. The 64-bit allocation table

The 64-bit allocation is 512 bytes of size

offset	size	value	description
0	496		The allocation table each bit represents a certain amount of bytes (block). A value of 1 means that the block is allocated, 0 if not
496	2		Type indicator
498	2		Emtpy values
500	4		A weak CRC32 of the 496 bytes of the table data
504	8		The allocation table offset

The allocation table at offset 0x4400 with type indicator 0x84 0x84 seems to address 64 byte blocks. Where the first bit refers to offset 0x4400. They are used for the data allocation.

The allocation table at offset 0x4600 with type indicator 0x83 0x83 seems to address 512 byte blocks. Where the first bit refers to offset 0x4400. They are used for the index node allocation.

4. The index b-tree

The PFF consists of multiple index b-trees.

- The (file) offset index b-tree (Block Binary Tree (BBT))
- The (item) descriptor index b-tree (Node Binary Tree (NBT))

These b-trees have a similar basic structure.

An index b-tree consists of:
- branch nodes that point to branch or leaf nodes
- leaf nodes that contain the index data

4.1. The 32-bit index b-tree node

Both the 32-bit branch and leaf node have a similar structure which is 512 bytes of size

offset	size	value	description
0	496		Node items (number of records x item size) Maximum of 496 the remaining values are zeroed
496	1		The amount of items The number of items that are used
497	1		The maximum amount of items
498	1		The size of an item
499	1		Node level A zero value represents a leaf node A value greater than zero branch nodes with the highest level representing the root
500	2		Type indicator 0x80 0x80 is used for offset index nodes 0x81 0x81 is used for descriptor index nodes
502	2		Unknown value Node identifier?
504	4		Back pointer must match the back pointer that pointed to this node
508	4		A weak CRC32 of the first 500 bytes of the index node

There can be additional node items in the node after the amount of items. These seem to be remnants of previously used index nodes which were not zeroed.

One of the remnant nodes is the index node with type indicator 0x85 0x85.

offset	size	value	description
0	4		Next node back pointer must match the back pointer of the next node
4	4		Next node offset

offset	size	value	description
8	488		Unknown values Maximum of 488 the remaining values are zeroed
496	1		Unknown value 0x00 in most nodes 0x40
497	1		Unknown value 0x00 in most nodes 0x0d 0x20 in some (last node?)
498	1		Empty value
499	1		Empty value
500	2		Type indicator 0x85 0x85 is used for ???
502	2		Unknown value Node identifier?
504	4		Back pointer must match the back pointer that pointed to this node
508	4		A weak CRC32 of the first 500 bytes of the index node

4.2. The 32-bit index b-tree branch node item

The 32-bit index b-tree node item is 12 bytes of size. It is only present in branch nodes.

offset	size	value	description
0	4		The index identifier of the first child node
4	4		The back pointer
8	4		The (file) offset

The index b-tree node will contain the following values:
The maximum amount of items: 41
The size of an item: 12

4.3. The 32-bit (file) offset index item

The 32-bit (file) offset index item is 12 bytes of size. It is only present in leaf nodes.

offset	size	value	description
0	4		The identifier
4	4		The (file) offset

offset	size	value	description
8	2		The size
10	2		Unknown Which is in most cases 0x02 Perhaps a flag of some kind Or bit mask for identifier value? Encryption flag?

The index b-tree node will contain the following values:
The maximum amount of items: 41
The size of an item: 12

An index b-tree node can contain the same identifier value as a (file) offset index item. This occurs when the leaf node is the lowest identifier in the branch node.

The second lower order bit of the identifier is used to flag if the data its referring to is encrypted.
- 0 = is encrypted;
- 1 = is not encrypted.

What about the first lower order bit (LSB) itself?

When the index tree is searched make sure to clear the lower order bit in the identifier.

Libpst: the two low order bits of the identifier value seem to be flags , Bit one indicates that the item is not encrypted. Note that references to these identifier values elsewhere may have the low order bit set (and I don't know what that means), but when we do the search in this tree we need to clear that bit so that we can find the correct item .

4.4. The 32-bit descriptor index b-tree leaf node item

The 32-bit descriptor index b-tree leaf node item is 16 bytes of size

offset	size	value	description
0	4		The (descriptor) index identifier
4	4		The (file) offset index identifier of the data
8	4		The (file) offset index identifier of the local descriptor list (only used when the item data is a table)
12	4		The parent (descriptor) index identifier

The index b-tree node will contain the following values:
The maximum amount of items: 31
The size of an item: 16

Duplicate identifiers? Libpst builds a tree and updates the values with those from successive leaf node items. What about lost items?

4.5. The 64-bit index b-tree node

Both the 64-bit branch and leaf node have a similar structure which is 512 bytes of size

offset	size	value	description
0	488		Branch node items (number of records x item size) Maximum of 488 the remaining values are zeroed
488	1		The amount of items The number of items that are used
489	1		The maximum amount of items
490	1		The size of an item
491	1		Node level A zero value represents a leaf node A value greater than zero branch nodes with the highest level representing the root
492	4		Empty values
496	2		Type indicator 0x80 0x80 is used for offset index nodes 0x81 0x81 is used for descriptor index nodes
498	2		Unknown value
500	4		A weak CRC32 of the first 496 bytes of the index node
504	8		Back pointer must match the back pointer from the triple that pointed to this node

4.6. The 64-bit index b-tree branch node item

The 64-bit index b-tree branch node item is 24 bytes of size

offset	size	value	description
0	8		The index identifier of the first child node
8	8		The back pointer
16	8		The (file) offset

The index b-tree node will contain the following values:
The maximum amount of items: 20
The size of an item: 24

4.7. The 64-bit (file) offset index item

The 64-bit (file) offset index item is 24 bytes of size

offset	size	value	description
0	8		The index identifier
8	8		The (file) offset
16	2		The size
18	2		Unknown Which is in most cases 0x02 Perhaps a flag of some kind Or bit mask for identifier value? Encryption flag?
20	4		Unknown

The index b-tree node will contain the following values:
The maximum amount of items: 20
The size of an item: 24

An index b-tree node can contain the same identifier value as a (file) offset index item. This occurs when the leaf node is the lowest identifier in the branch node.

The second lower order bit of the identifier is used to flag if the data its referring to is encrypted.

- 0 = is encrypted;
- 1 = is not encrypted.

What about the first lower order bit (LSB) itself?

When the index tree is searched make sure to clear the lower order bit in the identifier.

Libpst: the two low order bits of the identifier value seem to be flags , Bit one indicates that the item is not encrypted. Note that references to these identifier values elsewhere may have the low order bit set (and I don't know what that means), but when we do the search in this tree we need to clear that bit so that we can find the correct item .

4.8. The 64-bit descriptor index b-tree leaf node item

The 64-bit descriptor index b-tree leaf node item is 32 bytes of size

offset	size	value	description
0	4		The (descriptor) index identifier
4	4		Unknown (empty)
8	8		The (file) offset index identifier of the data
16	8		The (file) offset index identifier of the local descriptor list (only used when the item data is a table)

offset	size	value	description
24	4		The parent (descriptor) index identifier
28	4		Unknown Item type?

The index b-tree node will contain the following values:
The maximum amount of items: 15
The size of an item: 32

Duplicate identifiers? Libpst builds a tree and updates the values with those from successive leaf node items. What about lost items?

5. The list

The list identifier in the descriptor index b-tree leaf node entry refers to a (file) offset index entry which contains the file offset and data size of the list.

5.1. The 32-bit local descriptor list

The local descriptor list contains descriptor (file) offset mappings for table data. The local descriptor list is of type 2. The 32-bit local descriptor is variable in size.

offset	size	value	description
0	1	0x02	The list type
1	1		The list level
2	2		The amount of items
4	(amount x element size)		The list elements

Could the second byte signify the level of leaf/branch lists?

5.1.1. The 32-bit local descriptor list element

The level 0 32-bit local descriptor list element is 12 bytes of size. It is only present in level 0 lists.

offset	size	value	description
0	4		The list descriptor identifier
4	4		The (file) offset index identifier of the data
8	4		The (file) offset index identifier of the sub local descriptor list.

The lower bit in data identifier should be cleared before searching the value in the (file) offset index.

The (file) offset index identifier of the sub local descriptor list is mainly used in email items for attachments. It refers to the local descriptor list of the attachment item.

5.1.2. The 32-bit local descriptor sub list element

The type 1 32-bit local descriptor list element is 8 bytes of size. It is only present in non level 0 lists.

offset	size	value	description
0	4		The list descriptor identifier
4	4		The (file) offset index identifier of the sub local descriptor list.

The lower bit in data identifier should be cleared before searching the value in the (file) offset index.

The (file) offset index identifier of the sub local descriptor list is mainly used in email items with a large amount of local descriptors. It refers to the local descriptor list of the e-mail and attachment items.

5.2. The 64-bit local descriptor list

The local descriptor list contains descriptor (file) offset mappings for table data. The local descriptor list is of type 2. The 64-bit local descriptor is variable in size.

offset	size	value	description
0	1	0x02	The list type
1	1		The list level
2	2		The amount of items
4	4		Empty value
8	(amount x element size)		The list elements

5.2.1. The 64-bit local descriptor list element

The type 64-bit local descriptor list element is 24 bytes of size. It is only present in level 0 lists.

offset	size	value	description
0	4		The list descriptor identifier
4	4		Unknown value
8	8		The (file) offset index identifier of the data
16	8		The (file) offset index identifier of the sub local descriptor list.

The lower bit in data identifier should be cleared before searching the value in the (file) offset index.

The (file) offset index identifier of the sub local descriptor list is mainly used in email items for attachments. It refers to the local descriptor list of the attachment item.

6. The array

The array is used when a (file) offset index identifier contains more data than can fit in a single (descriptor) data block. The array contains a set of (file) offset index identifiers.

Note is there a value that signifies the maximum amount of data for a (descriptor) data block? 8180? (32-bit)

The array is used for both table as for item value data.

The reference array is variable of size.

offset	size	value	description
0	2	\x01\x01	The array signature
2	2		The amount of array entries
4	4		The total data size of the array entries
8	(amount x 4)		4 byte array entries containing (file) offset index identifiers

The total data size should equal the sum of all the (file) offset index entry sizes referenced by the array.

The data of the individual array entries should be concatenated to each other in order.

7. The table

The table contains entries which make up the pff items like email or contact. If the encryption type was set in the file header data the entire table is encrypted. Note that the not encrypted flag in the offset identifier can overwrite the table being encrypted.

The data identifier in the descriptor index b-tree leaf node entry refers to a (file) offset index entry which contains the file offset and data size of the table.

The table is variable of size.

offset	size	value	description
0	2		The table index offset
2	2		The table type \xec\x7c => 7c table \xec\x9c => 9c table

offset	size	value	description
			\xec\xa5 => a5 table \xec\xac => ac table \xec\xbc => bc table
4	4		The table value reference
8	4		Empty values
12	...		Table values
(offset)	...		The table index

In case of a table array the the first array (file) offset index entry refers to the data block that contains the table type.

7.1. The table value reference

The table value reference is formatted in different ways, it can point to data either in within the table block or in some other block.

- internal table value references have the all the low order 4 bits and high order 16 bits zero e.g. 0x0020, the value needs to be right shifted by 4 bits, e.g. 0x0002. This value is a byte offset which needs to be added to the table index offset + 2 (for the table index amount), so it points to a table index value pair e.g. (0xc, 0x14);
- external table value references have some of the low order 4 bits all set and the high order 16 bits are zero. They are descriptor list identifiers that refer to another location of data.

What about the 16 MSB and the array index?

7.1.1. Internal table value reference

An internal table value reference refers to the first table index value pair that contain the table values descriptor.

The internal table value reference for:

- a table type \xec\x7c points to a 7c table header;
- a table type \xec\x9c points to a 9c table header;
- a table type \xec\xa5 contains no value reference;
- a table type \xec\xac points to a ac table header;
- a table type \xec\xbc points to a b5 table header.

7.2. The table index

The table index is variable of size

offset	size	value	description
0	2		the amount (minus 1) of table index items
2	(amount + 2) x 2		Table index items

The amount contains a value that contains the amount of table index items minus 1.

The first table index item is probably referred to as number 0. The internal table reference value / 2 equals the table index item number.

7.2.1. The table index item

The table index item is 4 bytes of size.

offset	size	value	description
0	2		Table value start offset
2	4		Table value end offset

The offsets are relative to the start of the table.

The table index items are overlapping. The end offset of the first table index item is the start offset of the second. Therefore the table index contains 2 additional 2 byte values.

The last table value end offset does not always match the table index offset.

7.3. The b5 table header

The b5 table header is used in all table types. It contains information how the table entries are formatted. It consists of 8 bytes:

offset	size	value	description
0	1	\xb5	Table header type
1	1		The size of the item entry record
2	1		The size of the item entry value record
3	1	\x00	Emtpy value
4	4		Table entries reference

The table entry index reference refers to the table index value pair that points to table entries. If the table entries reference is zero there are no table entries.

The size of an individual table entry is the size of entry record and the size of value record combined.

The b5 table header differs for different tables:

Table	size of entry record	size of value record	size of table entry	Usage table entries
bc	2	6	8	Contains the item entry values
7c	4	2	6	Contain references to the start of the value arrays

Table	size of entry record	size of value record	size of table entry	Usage table entries
7c	4	4	8	Contain references to the start of the value arrays
9c	10	4	(14?)	Unknown
ac	4	4	8	Unknown

The individual table sections provide more information about the values in the table entries.

7.4. The bc table

The bc table has table values that contain:
- a b5 table header
- table entries that contain the item type/value information
- item entry value data

7.4.1. The b5 table header entry

The bc table uses the b5 table header with a size of entry record of 2 and the size of value record of 6. The table entries reference refers to the table entries. If the table entries reference is zero there are no table entries.

7.4.2. The table entries

The table entries in the bc table contain item entries. This type of table entry consists of 8 bytes:

offset	size	value	description
0	2		The item entry type
2	2		The item entry value type
4	4		The item entry value or value reference

For item entry value types that fit into 32-bit, like Integer 16-bit signed (0x0002), Integer 32-bit signed (0x0003), Boolean (0x000b), the item entry value is used directly. Otherwise, the item entry value is a value reference, which is either a descriptor list identifier, or a table index reference. If the item entry value is 0 the value is empty.

7.5. The 7c table

The bc table has table values that contain:
- a b5 table header
- a 7c table header
  - 7c table entry definitions that contain the item type information
- table entries that contain the value array information
- value array (table) entries that contain the item value information

7.5.1. The 7c table header

The 7c table header consists of 22 bytes:

offset	size	value	description
0	1	\x7c	Table header type
1	1		The amount of entry definitions
2	6		Unknown values (3 x 2 byte values) The values are related to the size of the values arrays.
8	2		The values array size
10	4		The b5 table header index reference
14	4		Values array entries index reference
18	4		Empty values (2 x 2 byte values ?)

If the b5 header table index reference is zero the table should not contain any table entries. If the value array entries index reference is zero the table does not contain any value array entries.

The table entries contain references to the table value array entries. So if the table contains no values the value array should be empty.

In some tables the b5 table header index reference contains a references to a b5 table header with an empty table entries reference. The value array entries index reference in the 7c table header is also emtpy.

It is possible for the table to have table header entries but no values array entries. The reverse is unknown.

7.5.2. The 7c entry definition

The remaining data in the 7c table header contains multiple entry definitions. The entry definitions describe the format of the data in the values array entries. The 7c entry definition consist of 8 bytes:

offset	size	value	description
0	2		The item entry value type
2	2		The item entry type
4	2		The values array entry offset
6	1		The values array entry size
7	1		The values array entry number (0 represents the first entry)

If the table contains values array entries the values array entry offset contains the offset of the value in the value array (table) entries.

In case of a value reference the actual value is found by reading the value size amount of bytes from the value array entries at the specified value array entries offset. A value array entries offset of 0 points to the beginning of the value array entries.

7.5.3. The b5 table header entry

The 7c table uses the b5 table header with a size of entry record of 4 and the size of value record of 2 or 4. The table entries reference refers to the table entries. If the table entries reference is zero there are no table entries.

7.5.4. The table entries

A b5 table header with a size of entry record of 4 and the size of value record of 2 refers to a specific type of table entry. This type of table entry consists of 6 bytes:

offset	size	value	description
0	4		Value array entry identifier The first value in the value array
4	2		Value array number

A b5 table header with a size of entry record of 4 and the size of value record of 4 refers to a specific type of table entry. This type of table entry consists of 8 bytes:

offset	size	value	description
0	4		Value array entry identifier The first value in the value array
4	4		Value array number

7.5.5. The values array entries

The values array entries contain item entries values. The 7c header entries define the format of the entry/value data within an array entry. The value size and value array entries offset in the 7c header entries refer to the item value in the value arrays.

The value array consist of multiple values of different sizes.

The value array seems to be ended by an unknown value of variable size.

For item entry value types that fit into the specified size the item entry value is used directly, i.e. 32-bit, like Integer 32-bit signed (0x0003) or 64-bit, like FileTime (0x0040). Otherwise, the item entry value is a value reference, which is either a descriptor list identifier, or a table index reference. If the item entry value is 0 the value is empty. Unlike the bc table the 7c table does store values smaller than 32-bit in lesser amount of bytes.

If a values array reference is an external reference and the values array is stored in a data array there is additional padding at the end of the last value array in a certain data array block. If the data in the data array is assumed continuous this causes a misalignment for the value array in the next data array block. The value array entry identifier in the table entries can be used to realign.

7.6. The 9c table

The 9c table has table values that contain:
- a b5 table header
- a 9c table header
- table entries that contain GUID descriptor values

7.6.1. The 9c table header

The ac table header consists of 4 bytes:

offset	size	value	description
0	4		B5 table header index reference

7.6.2. The b5 table header entry

The 9c table uses the b5 table header with a size of entry record of 16 and the size of value record of 4. The table entries reference refers to the table entries. If the table entries reference is zero there are no table entries.

7.6.3. The table entries

A b5 table header with a size of entry record of 16 and the size of value record of 4 refers to a specific type of table entry. This type of table entry consists of 20 bytes:

offset	size	value	description
0	16		A GUID
16	4		A descriptor identifier

7.7. The a5 table

The a5 table has table values that contain:
- table entries that contain table entry values

The a5 table is used by the ac table entry definitions as an array of table entry values.

The internal table value reference for the a5 table is 0.

7.8. The ac table

The ac table has table values that contain:

- a b5 table header
- a ac table header
- ac table entry definitions that contain the item type information
  - a5 tables containing the actual table entry values
- table entries that contain the value array information
- value array (table) entries that contain the item value information

7.8.1. The ac table header

The ac table header consists of 40 bytes:

offset	size	value	description
0	1	\xac	Table header type
1	1		Empty value
2	6		Unknown values Probably 3 x 2 byte values
8	2		The values array size
10	4		B5 table header index reference
14	4		Values array entry reference
18	4		Empty value
22	2		Amount of entry definitions
24	4		Table entry definitions reference
28	8		Empty value
36	4		Unknown value (Weak CRC?)

7.8.2. The ac entry definition

The table entry definitions reference refers to the ac entry definitions. The entry definitions describe the format of the data in the values array entries. The ac entry definition consist of 16 bytes:

offset	size	value	description
0	2		The item entry value type
2	2		The item entry type
4	2		The values array entry offset
6	2		The values array entry size
8	2		The values array entry number (0 represents the first entry)
10	2		Empty value
12	4		The descriptor identifier of the item

offset	size	value	description
			entry values table (a5 table)

If the table contains values array entries the values array entry offset contains the offset of the value in the value array (table) entries.

In case of a value reference the actual value is found by reading the value size amount of bytes from the value array entries at the specified value array entries offset. A value array entries offset of 0 points to the beginning of the value array entries.

7.8.3. The b5 table header entry

The ac table uses the b5 table header with a size of entry record of 4 and the size of value record of 4. The table entries reference refers to the table entries. If the table entries reference is zero there are no table entries.

It might be that the 4 + 2 variant like for the 7c table is also possible for the ac table.

7.8.4. The table entries

A b5 table header with a size of entry record of 4 and the size of value record of 4 refers to a specific type of table entry. This type of table entry consists of 8 bytes:

offset	size	value	description
0	4		Value array entry identifier The first value in the value array
4	4		Value array number

7.8.5. The values array entries

The values array entries contain item entries values. The ac header entries define the format of the entry/value data within an array entry. The value size and value array entries offset in the ac header entries refer to the item value in the value arrays.

The value array consist of multiple values of different sizes.

The value array seems to be ended by an unknown value of variable size.

For item entry value types that fit into the specified size the item entry value is used directly, i.e. 32-bit, like Integer 32-bit signed (0x0003) or 64-bit, like FileTime (0x0040). Otherwise, the item entry value is a value reference, which is either a descriptor list identifier, or a table index reference. If the item entry value is 0 the value is empty. Unlike the bc table the ac table does store values smaller than 32-bit in lesser amount of bytes.

Some entry definitions have a descriptor identifier of the item entry values table. This descriptor identifier refers to an a5 table which contains an array of table entry values. In this case the value in the values array actually contains an item index of the a5 table.

If a values array reference is an external reference and the values array is stored in a data array there is additional padding at the end of the last value array in a certain data array block. If the data in the data array is assumed continuous this causes a misalignment for the value array in the next data array block. The value array entry identifier in the table entries can be used to realign.

7.9. The item and item value types

The item and item value types are defined in the MAPI definitions document.

The item types are also referred to as the MAPI Property Names/Identifiers (PR_) or columns by scanpst. The item value types are also referred to as the MAPI Property (Data) Types (PT).

8. The PFF items

The pff items are stored in table entries. Different tables make up different pff items.

8.1. The message store

The descriptor index identifier 33 (0x21) refers to the message store.

This table contains the following entries:

- 0x0e34 (Unknown containing GUID)
- 0x0e38 (Unknown)
- 0x0ff9 (Binary record header 1)
  contains the GUID which is also in the folder information references
- 0x3001 (Display Name)
  “Personal Folders”
- 0x3416 (Folder information reference to "ItemProcSearch")
- 0x35df (Valid Folder Mask)
- 0x35e0 (Folder information reference to “Top of Personal Folder”)
- 0x35e2 (Folder information reference to “Outbox”)
- 0x35e3 (Folder information reference to “Deleted Items”)
- 0x35e4 (Folder information reference to “Sent Items”)
- 0x35e5 (Folder information reference to “IPM_VIEWS”)
- 0x35e6 (Folder information reference to “IPM_COMMON_VIEWS”)
- 0x35e7 (Folder information reference to “Search Root item”)
- 0x67ff (Password checksum)

PR_IPM_SUBTREE_ENTRYID
IPM_SUBTREE

Some pff files do not contain a message store.

8.2. The folder information

The (file) offset index identifier in the Folder information references of the message store refer to a folder information table.

This table contains the following entries:
- 0x3001 (Display Name)
- 0x3004 (Comment)
- 0x3602 (Amount of content items)
- 0x3603 (Amount of unread content items)
- 0x360a (Has sub folders)
- 0x3613 (Container class)

8.3. The name-to-id map

The descriptor index identifier 97 (0x61) refers to the the name-to-id map.

The name-to-id map table contains the following entries:
- 0x0001 (Unknown)
- 0x0002 (Name-to-ID Map Class identifiers)
- 0x0003 (Name-to-ID Map Entries)
- 0x0004 (Name-to-ID Map Strings)
- 0x1000 and up (Name-to-ID Map Validation Entries)

The entry 0x0002 (Name-to-ID Map Class Identifiers) is of type 0x0102 (Binary data) and contains an array of class identifiers (CLSID).

The entry 0x0003 (Name-to-ID Map Entries) is of type 0x0102 (Binary data) and contains an array of name-to-id map entries. An name-to-id map entry consist of 8 bytes.

offset	**size**	**value**	**description**
0	4		The name-to-id map entry value or value reference
4	2		The name-to-id map entry type
6	2		The name-to-id map entry number

The lowest bit in the name-to-id map entry type signifies where to find the name-to-id map value.
- If set it contains an offset into the 0x0004 (Name-to-ID Map Strings) array;
- If not set it contains the entry type to which the name-to-id is mapped.

The remaining name-to-id map entry type value refers to a value in the class id array:

```
index number = ( type / 2 ) - 3
```

The correspondent item type is the name-to-id map number + 0x8000.

The entry 0x0004 (Name-to-ID Map Strings) is of type 0x0102 (Binary data) and contains an array of strings. An individual string consists of:

offset	size	value	description
0	4		The amount of bytes in the string
4	...		The string in UTF-16 without the end of string character

The entries s0x1000 and up (Name-to-ID Map Validation Entries) contain values similar to those in the entry 0x0003 (Name-to-ID Map Entries). Except that these are used for validation.

offset	size	value	description
0	4		The name-to-id map entry validation value
4	2		The name-to-id map entry type
6	2		The name-to-id map entry number

The lowest bit in the name-to-id map entry type signifies where to find the name-to-id map validation value.
- If set it contains a weak CRC32 of the string in the 0x0004 (Name-to-ID Map Strings) array;
- If not set it contains a duplicate of the value in the 0x0003 (Name-to-ID Map Entries).

8.4. The folder item

The descriptor index identifier 290 (0x112) refers to the the root folder item table.

The folder item table contains the following entries:
- 0x3001 (Display Name)
  This is an empty string
- 0x3602 (Amount of content items)
- 0x3603 (Amount of unread content items)
- 0x360a (Has sub folders)
- 0x6635 (Unknown)
  Integer 32-bit signed
- 0x6636 (Unknown)
  Integer 32-bit signed

Optional entries:
- 0x3617 (Associate content count)

Additional folder item table entries for Inbox:
- 0x36d0 (Folder information reference to “Agenda”)
- 0x36d1 (Folder information reference to “Contacts”)
- 0x36d2 (Folder information reference to “Journal”)
- 0x36d3 (Folder information reference to “Notes”)
- 0x36d4 (Folder information reference to “Tasks”)
- 0x36d5 (Folder information reference to “Reminders”)
- 0x36d7 (Folder information reference to “Drafts”)
- 0x36da (Unknown)
  Binary data, 6 bytes
- 0x36e4 (Unknown)

Array of binary data

The descriptor index entry of the root item refers to itself as its parent.

The child items can be found by the parent descriptor identifier in the descriptor index entry. The descriptor index entries that are not part of the item hierarchy should not contain parent identifiers.

The amount of content items in a folder is made up from the item count and associated item count .

8.5. The e-mail item

This table contains the following entries:

- 0x0017 (Importance)
- 0x001a (Message class)
- 0x0026 (Priority)
- 0x0036 (Sensitivity)
- 0x0037 (Email Subject)
- 0x0039 (Client submit time)
- 0x003b (Sent representing search key)
- 0x003f (Received by entry identifier)
- 0x0040 (Received by name)
- 0x0041 (Sent representing entry identifier)
- 0x0042 (Sent representing name)
- 0x0043 (Received representing entry identifier)
- 0x0044 (Received representing name)
- 0x0051 (Received by search key)
- 0x0052 (Received representing search key)
- 0x0064 (Sent representing address type)
- 0x0065 (Sent representing e-mail address)
- 0x0070 (Conversation topic)
- 0x0075 (Received by address type)
- 0x0076 (Received by e-mail address)
- 0x0077 (Received representing address type)
- 0x0078 (Received representing e-mail address)
- 0x007d (Mail Internet Headers)
- 0x0c19 (Second sender structure)
- 0x0c1a (Sender name)
- 0x0c1d (Sender search key)
- 0x0c1e (Sender address type)
- 0x0c1f (Sender e-mail address)
- 0x0e04 (Display TO)
- 0x0e06 (Message delivery time)
- 0x0e07 (Message flags)
- 0x0e08 (Message size)
- 0x0e1f (Compressed RTF in Sync)
- 0x1000 (Plain Text Email Body)
  Does not exist if the email doesn’t have a plain text version
- 0x1006 (RTF Sync Body CRC)
- 0x1007 (RTF Sync Body character count)

- 0x1008 (RTF Sync body tag)
- 0x1009 (RTF Compressed body)
  LZFu compression?
- 0x1010 (RTF whitespace prefix count)
- 0x1011 (RTF whitespace tailing count)
- 0x1013 (HTML Email Body)
  Does not exist if the email doesn't have an HTML version
- 0x1035 (Message ID)
- 0x1046 (Return Path)
- 0x3007 (Creation time)
- 0x3008 (Modification time)
- 0x300b (Search key)
- 0x3fde (Unknown)
- 0x6610 (Unknown)
- 0x800f (Unknown)
- 0x8010 (Unknown)
- 0x8011 (Unknown)
- 0x8013 (Unknown)

8.6. The attachments item

When an e-mail contains attachments the local descriptor list contains an entry 1649 (0x0671). This local descriptor refers to a 7c table which contains an attachments item.

Note: does the attachments table need to be of type 7c because it is referenced by its offset index value and has not got a local descriptor list? It also does not have a matching data identifier in the descriptor index.

The attachments item table contains multiple sets (1 per attachment) of the following entries:
- 0x0e20 (Attachment Size)
- 0x3704 (Attachment Filename)
- 0x3705 (Attachment method)
- 0x370b (Attachment Position)
- 0x67f2 (Attachment local descriptor identifier)
- 0x67f3 (Unknown)

The attachment local descriptor identifier refers to an entry in the local descriptor list of the corresponding e-mail item. The list identifier of the local descriptor entry refers to the local descriptor list of the attachment item.

8.7. The attachment item

The attachment item table contains the following entries:
- 0x0e20 (Attachment Size)
- 0x0e27 (Unknown)
- 0x3007 (Creation time)
- 0x3008 (Modification time)
- 0x3701 (Attachment data object)
- 0x3704 (Attachment filename)

- 0x3705 (Attachment method)
- 0x3707 (Attachment long filename)
- 0x370a (Unknown)
- 0x370b (Attachment Position)
- 0x370e (Attachment mime type)
- 0x3712 (Unknown)
- 0x3713 (Unknown)
- 0x3714 (Unknown)
- 0x3716 (Unknown – some string “inline”)

The Attachment data object (0x3701) can be of type Binary Data (0x0102) and Embedded Object (0x000d). The Embedded Object is used for bounced e-mails in which it contains an embedded PFF table. The Embedded Object can also contain other data like an OLE2 document.

Attachment method (0x3705) value	**Attachment data object (0x3701) entry type**	**Attachment type**
0x00000001	Binary Data (0x0102)	Attached data (Separately attached data)
0x00000005	Embedded Object (0x000d)	Attached item (Embedded PFF item)
0x00000006	Embedded Object (0x000d)	Attached data (Embedded OLE2 document)

Found empty attachment data object (0x3701) value but attachment has size 206. But contains entry 0x0e27 with binary data of size 100.

8.8. The appointment item

The appointment item table contains the following entries:
- 0x0002 (Alternate recipient allowed)
- 0x0017 (Importance)
- 0x001a (Message class)
- 0x0023 (Originator delivery report requested)
- 0x0026 (Priority)
- 0x0029 (Read receipt requested)
- 0x0036 (Sensitivity)
- 0x0037 (Subject)
- 0x0039 (Client submit time)
- 0x003b (Sent representing search key)
- 0x0041 (Sent representing entry identifier)
- 0x0042 (Sent representing name)
- 0x0063 (Response requested)
- 0x0064 (Sent representing address type)
- 0x0065 (Sent representing e-mail address)
- 0x0070 (Conversation topic)
- 0x0071 (Conversation index)
- 0x0c17 (Reply requested)
- 0x0c19 (Sender entry identifier)

- 0x0c1a (Sender name)
- 0x0c1d (Sender search key)
- 0x0c1e (Sender address type)
- 0x0c1f (Sender e-mail address)
- 0x0e01 (Delete after submit)
- 0x0e06 (Message delivery time)
- 0x0e07 (Message flags)
- 0x0e08 (Message size)
- 0x1080 (Unknown)
- 0x3007 (Creation time)
- 0x3008 (Modification time)
- 0x300b (Search key)
- 0x3fde (Unknown)
- 0x8002 (Unknown)
- 0x8003 (Unknown)
- 0x8004 (Unknown)
- 0x8005 (Unknown FileTime) Appointment start time?
- 0x8006 (Unknown) Appointment end time?
- 0x8007 (Unknown)
- 0x8008 (Unknown) Appointment location
- 0x800b (Unknown)
- 0x800c (Unknown)
- 0x800e (Unknown)
- 0x800f (Unknown)
- 0x8010 (Unknown)
- 0x8011 (Unknown)
- 0x8012 (Unknown)
- 0x8013 (Unknown)
- 0x8015 (Unknown)
- 0x8016 (Unknown)
- 0x8017 (Unknown)
- 0x8018 (Unknown)
- 0x8019 (Unknown)
- 0x801a (Unknown) Version number? “11.0”
- 0x801b (Unknown) Boolean
- 0x801c (Unknown) Appointment start time?
- 0x801d (Unknown) Appointment end time?
- 0x801e (Unknown)
- 0x801f (Unknown)
- 0x8020 (Unknown)
- 0x8021 (Unknown)
- 0x8022 (Unknown)
- 0x8023 (Unknown)
- 0x8024 (Unknown)
- 0x8025 (Unknown)
- 0x8026 (Unknown)
- 0x8027 (Unknown)
- 0x8028 (Unknown) Appointment reoccurring
- 0x8029 (Unknown) Appointment timezone
- 0x802a (Unknown) Filetime (time range first occurance)
- 0x802b (Unknown) Filetime (time range last occurance)
- 0x802c (Unknown)

- 0x802d (Unknown)
- 0x802e (Unknown)
- 0x802f (Unknown)
- 0x8030 (Unknown)
- 0x8031 (Unknown) Timezone info?
- 0x8032 (Unknown) Timezone info?
- 0x8033 (Unknown)

subject
from
Start
End
Location
Recurring
Comment => opmerking

8.9. The contact item

The contact item table contains the following entries:
- 0x0002 (Alternate recipient allowed)
- 0x0017 (Importance)
- 0x001a (Message class)
- 0x0023 (Originator delivery report requested)
- 0x0026 (Priority)
- 0x0029 (Read receipt requested)
- 0x0036 (Sensitivity)
- 0x0037 (Subject)
- 0x0039 (Client submit time)
- 0x003b (Sent representing search key)
- 0x0041 (Sent representing entry identifier)
- 0x0042 (Sent representing name)
- 0x0064 (Sent representing address type)
- 0x0065 (Sent representing e-mail address)
- 0x0070 (Conversation topic)
- 0x0071 (Conversation index)
- 0x0c19 (Sender entry identifier)
- 0x0c1a (Sender name)
- 0x0c1d (Sender search key)
- 0x0c1e (Sender address type)
- 0x0c1f (Sender e-mail address)
- 0x0e01 (Delete after submit)
- 0x0e06 (Message delivery time)
- 0x0e07 (Message flags)
- 0x0e08 (Message size)
- 0x1000 (Plain Text Email Body) Remarks?
- 0x1009 (RTF Compressed body) RTF Remarks?
- 0x1080 (Unknown)
- 0x3001 (Display name)
- 0x3007 (Creation time)
- 0x3008 (Modification time)
- 0x300b (Search key)

- 0x3a00 (Contact's Account name)
- 0x3a05 (Contacts Suffix)
- 0x3a06 (Contacts First Name)
- 0x3a08 (Business Telephone Number)
- 0x3a09 (Home Telephone Number)
- 0x3a0a (Contacts Initials)
- 0x3a0b (Keyword)
- 0x3a0c (Contact's Language)
- 0x3a0d (Contact's Location)
- 0x3a11 (Contacts Surname)
- 0x3a15 (Default Postal Address)
- 0x3a16 (Company Name)
- 0x3a17 (Job Title)
- 0x3a18 (Department Name)
- 0x3a19 (Office Location)
- 0x3a1c (Mobile Phone Number)
- 0x3a24 (Business Fax Number)
- 0x3a26 (Business Address Country)
- 0x3a27 (Business Address City)
- 0x3a28 (Business Address State)
- 0x3a29 (Business Address Street)
- 0x3a2a (Business Postal Code)
- 0x3a30 (Assistant's Name)
- 0x3a41 (Wedding Anniversary)
- 0x3a42 (Birthday)
- 0x3a44 (Middle Name)
- 0x3a45 (Display Name Prefix (Contact Title))
- 0x3a46 (Profession)
- 0x3a48 (Spouse's Name)
- 0x3a4e (Manager's Name)
- 0x3a4f (Nickname)
- 0x3a51 (Business Home Page)
-

9. LZFu compression

The LZFu compression is used for RTF formatted data [ROTHMAN99].

Compressed LZFu data starts with a LZFu header

offset	**size**	**value**	**description**
0	4		Size of the compressed data including the following 12 bytes of the header
4	4		Size of the uncompressed data
8	4		Compression signature
12	4		A CRC32 of the compressed data.

The signature 0x75465a4c (“LZFu”) that the data is compressed. The signature 0x414c454d (“MELA”) that the data is uncompressed.

The CRC32 is similar to the standard CRC32 algorithm which is mentioned in RFC 1952 with a slight modification: the inversion (or xor with 0xffffffffL) before and after the CRC update is omitted. This inversion is applied in order to avoid a CRC weakness, which is that any number of leading or trailing zero bytes can be added or removed without the CRC detecting the change. For some reason, the compressed RTF CRC32 implementation is the weaker one, without this inversion. It is calculated on the compressed data bytes (excluding the LZFu header).

The compressed data is directly after the header. It consists of 8-unit chunks. Each chunk begins with a single flag byte. The bits within the flag byte are read in LSB order. Each bit in the byte flag is a flag for the corresponding unit in the chunk.

- 0 represent a 1 byte literal which should be copied as-is;
- 1 represent a 2 byte reference.

A 2 byte reference consists of:

offset	size	value	description
0	1.4		Reference offset into the LZ buffer
1.4	0.4		Reference size

The reference offsets represent offsets into the LZ buffer. The size of the reference offset allows for 4096 possible values, which is the size of the LZ buffer. The LZ buffer wraps around as it is filled with the decompressed data. The LZ buffer is preloaded with a common RTF header string (found in RTFLIB32.LIB). The string is represented as a C string of 207 bytes.

```
{\\rtf1\\ansi\\mac\\deff0\\deftab720{\\fonttbl;}{\\f0\\fnil \\froman
\\fswiss \\fmodern \\fscript \\fdecor MS Sans SerifSymbolArialTimes New
RomanCourier{\\colortbl\\red0\\green0\\blue0\n\r\\par \\pard\\plain\\f0\\fs20\\
b\\i\\u\\tab\\tx
```

The reference size is a 4 bit value that represents a value between 2 and 17. A reference size of 0 representing 2. Therefore the reference size needs to be corrected by 2.

The uncompressed size does not entail the 2 trailing zero bytes.

10. Notes

10.1. Root items

```
33      message store
97      name-to-id map
290     root folder
301     folders table (7c table) (secondary message store?)
302     empty 7c table
303     empty 7c table
481     emtpy descriptor (search folder?)
513     (unknown)
609     (unknown)
641     emtpy descriptor
673     (unknown)
801     emtpy descriptor
1549    empty 7c table (template of some kind?)
1550    empty 7c table (template of some kind?)
```

```
1551    empty 7c table (template of some kind?)
1552    empty 7c table (template of some kind?)
1579    7c table
1612    empty 7c table
1649    empty 7c table (template of some kind?)
1682    empty 7c table (template of some kind?)
1718    empty 7c table
1751    empty 7c table
1784    7c table
1840    emails/items table (ac table)
3073    guid lookup table (9c table)
8742    emtpy descriptor
8743    bc table
8752    empty 7c table
32813   7c table
32814   empty 7c table
32815   empty 7c table
32845   7c table
32846   empty 7c table
32877   empty 7c table
32878   empty 7c table
32879   empty 7c table
32909   empty 7c table
32910   emails/items table (7c table)
32911   empty 7c table
524326  emtpy descriptor
524327  bc table
524336  empty 7c table
```

Appendix A. References

[SMITH02]
Title: outlook.pst -- format of MS Outlook .pst file
Author(s): David Smith, Joe Nahmias, Brad Hards, Carl Byington
URL: http://hg.file-ten-sg.com/libpst/

[ROTHMAN99]
Title: The Compressed RTF Format
Author(s): Amichai Rothman
URL: http://www.freeutils.net/source/jtnef/rtfcompressed.jsp

[OPENCHANGE]
Title: Openchange MAPI library
URL: http://www.openchange.org/index.php

[MSDN]
Title: Microsoft Developer Network
URL: http://msdn.microsoft.com/

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.