EWF specification

Expert Witness Compression Format specification

By Joachim Metz <joachim.metz@gmail.com>

Summary

EWF is short for Expert Witness Compression Format, according to [ASR02]. It is a file type used to store media images for forensic purposes. It is currently widely used in the field of computer forensics in proprietary tooling like EnCase en FTK. The original specification of the format is provided by ASRDATA, for the SMART application.

The EWF format was succeeded by the Expert Witness Compression Format version 2 in EnCase 7 (EWF2-Ex01 and EWF2-Lx01). EnCase 7 also uses a different version of EWF-L01 then its predecessors.

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

Document information

Author(s): Joachim Metz <joachim.metz@gmail.com>

Abstract: This document contains the EWF file format specification.

Classification: Public

Keywords: Expert Witness Compression Format, EWF, EnCase file format, SMART

License

Version

Version	Author	Date	Comments
0.0.1	J.B. Metz	March 2006	Initial version
0.0.2	J.B. Metz	March 2006	Additional information.
0.0.3	J.B. Metz	March 2006	Additional information.
0.0.4	J.B. Metz	March 2006	Additional information.
0.0.5	J.B. Metz	March 2006	Additional information, regarding data and header2 section.
0.0.6	J.B. Metz	March 2006	Additional information, regarding data and header2 section.
0.0.7	J.B. Metz	March 2006	Additional information, regarding data, hash and header2 section.
0.0.8	J.B. Metz	March 2006	Additional information, regarding data section.
0.0.9	J.B. Metz	March 2006	Additional information, regarding chunk and compression, offset array CRC and error2 section.
0.0.10	J.B. Metz	March 2006	Correction regarding EnCase 3 and compression MSB.
0.0.11	J.B. Metz	March 2006	Additions regarding EnCase 2.
0.0.12	J.B. Metz	March 2006	Small changes regarding unknown in volume and data. Removed some spelling errors. Added the information regarding when a chunk is compressed or not.
0.0.13	J.B. Metz	April 2006	Additions regarding EnCase 1.
0.0.14	J.B. Metz	April 2006	Additions regarding endian.
0.0.15	J.B. Metz	April 2006	Additions regarding disk section.
0.0.16	J.B. Metz	April 2006	Small adjustments regarding header section.
0.0.17	J.B. Metz	April 2006	Adjustments in error2 section information.
0.0.18	J.B. Metz	May 2006	Adjustments in hash section information.
0.0.19	J.B. Metz	August 2006	Fixed error in Encase 4 header2 layout information.

Version	Author	Date	Comments
0.0.20	J.B. Metz	August 2006	Added information regarding SMART format generated by FTK Imager. Corrected error about gzip compression in header section.
0.0.21	J.B. Metz	August 2006	Added information regarding SMART format generated by FTK Imager.
0.0.22	J.B. Metz	August 2006	Added information about segment file extension naming.
0.0.23	J.B. Metz	September 2006	Added information about EWF-L01 (LVF) format.
0.0.24	J.B. Metz	September 2006	Added information from EWF-L01 analysis.
0.0.25	J.B. Metz	September 2006	Changes after comments by Guy Voncken.
0.0.26	J.B. Metz	October 2006	Corrected error regarding EnCase 1 and SMART header specification.
0.0.27	J.B. Metz	October 2006	Added theoretical maximum media size.
0.0.28	J.B. Metz	October 2006	Additional information about section start size in EnCase (EWF-E01) next and done sections.
0.0.29	J.B. Metz	November 2006	Additional information about CRC algorithm.
0.0.30	J.B. Metz	November 2006	Fixed error regarding the location of the actual chunks in the EnCase 1 format, which actually is the table sections and not the sectors section.
0.0.31	J.B. Metz	November 2006	Additional information about the EnCase linen 5 (EWF-E01) format.
0.0.32	J.B. Metz	December 2006	Additional information about GUID.
0.0.33	J.B. Metz	December 2006	Corrected error regarding header sections in EnCase 1 format.
0.0.34	J.B. Metz	December 2006	Added new information regarding the table section after encountering a bug in FTK for EWF files with more than 16 * 1024 offset table entries.
0.0.35	J.B. Metz	December 2006	Corrected misinterpretation of original specifications, regarding additional table sections.
0.0.36	J.B. Metz	January 2007	Added information about EnCase 6.
0.0.37	J.B. Metz	January 2007	Added information about linen 6.
0.0.38	J.B. Metz	January 2007	Added information about EnCase6/linen6 header. Adjustments regarding media type and media flags.
0.0.39	J.B. Metz	January 2007	Added information about header values.
0.0.40	J.B. Metz	January 2007	Added information about EWF-X
0.0.41	J.B. Metz	August 2007	Added information about EnCase 6.7 >2Gb segment file support.
0.0.42	J.B. Metz	August 2007	Added information about EnCase 6.7 >2Gb segment file support and CD/DVD image session sector.
0.0.43	J.B. Metz	September 2007	Added information about EnCase 6.7 >2Gb segment file support.
0.0.44	J.B. Metz	September 2007	Added page numbers.
0.0.45	J.B. Metz	November 2007	Added information about session section.
0.0.46	J.B. Metz	March 2008	Added information about session section.
0.0.47	J.B. Metz	March 2008	Added information about EnCase 6 >2GiB segment file format.
0.0.48	J.B. Metz	June 2008	Textual corrections.
0.0.49	J.B. Metz	June 2008	Added information about EnCase 6.11 winen file format.
0.0.50	J.B. Metz	February 2009	Added information about EnCase 6.12 SHA1 hash support and

Version	Author	Date	Comments
			header values.
0.0.51	J.B. Metz	April 2009	Added information about EnCase software version header value limitation.
0.0.52	J.B. Metz	April 2009	Added information about EnCase 6.13 Tableau write blocker support.
0.0.53	J.B. Metz	November 2009	Small changes.
0.0.54	J.B. Metz	December 2009 January 2010	Added information about ltree section.
0.0.55	J.B. Metz	January 2010	Update for linen 6.12 and later.
0.0.56	J.B. Metz	May 2010	Corrected amount of into number of. Email change
0.0.57	J.B. Metz	September 2010	Minor changes.
0.0.58	J.B. Metz	September 2010	Changed CRC to checksum.
0.0.59	J.B. Metz	October 2010	Additional session section information with thanks to M. Nohr Updated some tables to the newer format. Minor changes.
0.0.60	J.B. Metz	November 2010	Minor changes and improvements with thanks to G. Voncken. Updated some tables to the newer format.
0.0.61	J.B. Metz	December 2010	License version update Additional information about optical discs. Additional information about AD encryption.
0.0.62	J.B. Metz	January 2011	Minor changes
0.0.63	J.B. Metz	February 2011	Additional audio tracks information with thanks to M. Nohr
0.0.64	J.B. Metz	May 2011	Changes to FTK imager format
0.0.65	J.B. Metz	June 2011	Updated Logical File Evidence (LVF) format flag information with thanks to B. Baron.
0.0.66	J.B. Metz	September 2011	Updated Logical File Evidence (LVF) format flag information with thanks to N. Harris
0.0.67	J.B. Metz	December 2011	Small refinement in compressed vs uncompressed chunk data.
0.0.68	J.B. Metz	February 2012	Added information about EnCase header values limitations thanks to G. Voncken.
0.0.69	J.B. Metz	June 2012	Added information about EnCase 6.19 and 7, EWF-E01 and EWF-L01 format. Email change; text clean up; some corrections and additions.
0.0.70	J.B. Metz	July 2012	Changes to match EWF version 2 documentation.
0.0.71	J.B. Metz	July 2012	Updates regarding ltree header.
0.0.72	J.B. Metz	July 2012	Updates files created by Expert Witness 1.35 (for Windows). Other small corrections.
0.0.73	J.B. Metz	August 2012	Updates regarding ltree header.
0.0.74	J.B. Metz	August 2012	Updates regarding incomplete section and corruption scenarios with thanks to B. Johnson for pointing out the dual image scenario.
0.0.75	J.B. Metz	September 2012	Additional information regarding L01 map entry.
0.0.76	J.B. Metz	January 2013	Corrected some typos, thanks to A. Bridge for pointing these out.
0.0.77	J.B. Metz	March 2013	Additional information regarding Logicube created E01 files with thanks to D. Kovar and Digital Assembly LLC.

Version	Author	Date	Comments

Table of Contents

1. Overview

The Expert Witness Compression Format (EWF) is used to store media images. It allows to store disk and partition images, compressed or non-compressed. EWF can store a single image in one or more *segment files*. Each *segment file* consist of a standard header, followed by multiple *sections*. A single *section* cannot span multiple files. *Sections* are arranged back-to-back.

Specifications

- In this document when referred to the EWF format it refers to the original specification by [ASR02]. The newer formats like that of EnCase are deducted from the original specification and will be referred to as the EWF-E01, because of the default file extension. Whereas the Logical File Evidence (LVF) format introduced in EnCase 5, which is also stored in the EWF format will be referred to as EWF-L01. The SMART format is viewed separately to allow for discussion if the implementation differs from the specification by [ASR02] and will be referred to as the EWF-S01, because of the default file extension.
- All offsets are relative to the beginning of an individual *section*, unless otherwise noted. EnCase allows a maximum size of a *segment file* to be 2000 MiB. This has to do with the size of the offset of the chunk of media data. This is a 32 bit value where the most significant bit (MSB) is used as a compression flag. Therefore the maximum offset size (31 bit) can address about 2048 MiB. In EnCase 6.7 an addition was made to the table value to provide for a base offset to allow for segment files greater than 2048 MiB.
- A chunk is defined as the sector size (per default 512 bytes) multiplied by the block size, the number of sectors per chunk (block) (per default 64 sectors). The data within the EWF format is stored in little-endian. The terms block and chunk are used intermittently.

1.1. Test version

The following version of programs were used to test the information within this document:

- FTK Imager 2.3, 2.4, 2.51, 2.9, 3.0 (Windows)
- Expert Witness 1.35 (for Windows) (EnCase 1.35)
- EnCase 1.99l (Windows)
- EnCase 2.17a (DOS)
- EnCase 3.21b (Windows)
- EnCase 4.22 (Windows)
- EnCase 5.04a, 5.05 (Windows)
- EnCase 6.1, 6.7, 6.8, 6.10, 6.11, 6.12, 6.13, 6.14, 6.19 (Windows)
- EnCase 7.04 (Windows)
- Linen 5 (Linux)
- Linen 6.01, 6.19 (Linux)
- Linen 7.01 (Linux)

EnCase 7 no longer provides the fast and best compression options.

2. Segment file

EWF stores data in one or more segment files (or segments). Each segment file consists of:

- A file header.
- One or more sections.

2.1. File header

Each segment file starts with a file header.

[ASR02] defines that the file header consists of 2 parts, namely:
- a signature part
- fields part

2.1.1. EWF, EWF-E01 and EWF-S01

This is file header is defined by [ASR02] and both used by the EWF-E01 and EWF-S01 formats.

The file header is 13 bytes of size and consists

offset	**size**	**value**	**description**
0	8		Signature "EVF\x09\x0d\x0a\xff\x00"
8	1	0x01	Start of fields
9	2		Segment number Must be 1 or higher
11	2	0x0000	End of fields

Segment number contains a number which refers to the number of the segment file, starting with 1 for the first file.

Note: This means there could only be a maximum of 65535 (0xffff) files, if it is an unsigned value.

2.1.2. EWF-L01

This is file header is used by the EWF-L01 format.

The file header is 13 bytes of size and consists

offset	**size**	**value**	**description**
0	8		Signature "LVF\x09\x0d\x0a\xff\x00"
8	1	0x01	Start of fields
9	2		Segment number Must be 1 or higher
11	2	0x0000	End of fields

Segment number contains a number which refers to the number of the segment file, starting with 1 for the first file.

Note: This means there could only be a maximum of 65535 (0xffff) files, if it is an unsigned value.

2.2. Segment file extensions

Both the SMART (EWF-S01) and the EWF-E01 use a different approach for naming the segment files.

EWF-S01

The EWF-S01 extension naming has two distinct parts.

- The first segment file has the extension '.s01'.
  - The next segment file has the extension '.s02.
  - This will continue up to '.s99'.
- After which the next segment file has the extension '.saa'.
  - The next segment file has the extension '.sab'.
  - This will continue up to '.saz'.
  - The next segment file has the extension '.sba'.
  - This will continue up to '.szz'.
  - The next segment file has the extension '.faa'.
  - This will continue up to '.zzz'. (verify this; and then ?)
  - It will even continue to the use the extensions '.{aa'. (not confirmed)

libewf supports extensions up to .zzz

EWF-E01

The EWF-E01 extension naming has two distinct parts.

- The first segment file has the extension '.E01'.
  - The next segment file has the extension '.E02.
  - This will continue up to '.E99'.
- After which the next segment file has the extension '.EAA'.
  - The next segment file has the extension '.EAB'.
  - This will continue up to '.EAZ'.
  - The next segment file has the extension '.EBA'.
  - This will continue up to '.EZZ'.
  - The next segment file has the extension '.FAA'.
  - This will continue up to '.ZZZ'. (verify this; and then ?)
  - It will even continue to the use the extensions '.[AA'. (not confirmed)

libewf supports extensions up to .ZZZ

EWF-L01

The EWF-L01 extension naming has two distinct parts.

- The first segment file has the extension '.L01'.
  - The next segment file has the extension '.L02.
  - This will continue up to '.L99'.
- After which the next segment file has the extension '.LAA'.
  - The next segment file has the extension '.LAB'.
  - This will continue up to '.LAZ'.
  - The next segment file has the extension '.LBA'.
  - This will continue up to '.LZZ'.
  - The next segment file has the extension '.MAA'.

- This will continue up to '.ZZZ'. (verify this; and then ?)
- It will even continue to the use the extensions '.[AA'. (not confirmed)

libewf supports extensions up to .ZZZ

3. The sections

The remainder of the segment file consists of sections. Every section starts with the same data this will be referred to as the section descriptor (previously referred to as section start). The section descriptor could also be referred as the section header, but this allows for unnecessary confusion with the header section.

3.1. Section descriptor

The section descriptor consist of 76 bytes, it contains information about a specific section.

offset	**size**	**value**	**description**
0	16		A string containing the section type definition. E.g. “header”, “volume”, etc.
16	8		Next section offset The offset is relative from the start of the segment file
24	8		Section size
32	40	0x00	Padding
72	4		Checksum Adler-32 of all the previous data within the section descriptor.

Some sections contain additional data, refer to paragraph section types for more information.

Notes:

- In EnCase 2 DOS version the padding itself does not contains zero byte values but data, probably the memory is not wiped.
- Expert Witness 1.35 (for Windows) does not set the section size.

3.2. Section types

There are multiple section types [ASR02] defines the following:

- Header section
- Volume section
- Table section
- Next and Done section

Looking at more recent EnCase file (EWF-E01) formats and [COH] additional section types were found:

- Header2 section
- Disk section
- Sectors section

- Table2 section
- Data section
- Errors2 section
- Session section
- Hash section
- Digest section

Looking at the more recent EnCase file (EWF-L01) format additional section types were found:
- Ltree section
- Ltypes section

3.2.1. Header2 section

The header2 section is identified in the section data type field as “header2”. Some aspects of this section are:
- Found in EWF-E01 in EnCase 4 to 7, and EWF-L01 in EnCase 5 to 7
- Found at the start of the first segment file. Not found in other segment files.
- The same header2 section is found twice directly after one and other.

The additional data this section contains is the following

Offset:	**number of bytes:**	**Meaning:**
76 (0x4c)	(variable)	Information about the acquired media.

The information about the acquired media consists of zlib compressed data. It contains text in UTF16 format specifying information about the acquired media. The text multiple lines separated by an end of line character(s).

The first 2 bytes of the UTF16 string are the byte order mark (BOM):
- 0xff 0xfe for UTF-16 litte-endian
- 0xfe 0xff for UTF-16 big-endian

In the next paragraphs the various variants of the header2 section are described.

EnCase 4

EWF-E01

In EnCase 4 the header2 information consist of 5 lines, and contains the equivalent information as the header section.

Line number	**Meaning**	**Value**
1	The number of categories provided	1
2	The name/type of the category provided	main
3	Identifiers for the values in the 4th line	
4	The data for the different identifiers in the 3rd line	
5	(an empty line)	

The end of line character(s) is a newline (0x0a).

Note: This end of line character differs from the one used in the header section.

The 3rd and the 4th line consist of the following tab (0x09) separated values.

Identifier number	Character in 3rd line	Value in 4th line
1	a	Unique description
2	c	Case number
3	n	Evidence number
4	e	Examiner name
5	t	Notes
6	av	Version The EnCase version used to acquire the media
7	ov	Platform The platform/operating system used to acquire the media
8	m	Acquired date
9	u	System date
10	p	Password hash

For more information see section: Header2 values

Note: the hashing algorithm is the same as for the header section.

EnCase 5 to 7

EWF-E01

The header2 information consist of 18 lines

The remainder of the string contains the following information:

Line number	Meaning	Value
1	The number of categories provided	3
2	The name/type of the category provided	main
3	Identifier for the values in the 4th line	
4	The data for the different identifiers in the 3rd line	
5	(an empty line)	
6	The name/type of the category provided	srce
7		
8	Identifier for the values in the category	
9		
10		
11	(an empty line)	
12	The name/type of the category provided	sub
13		
14	Identifier for the values in the category	
15		
16		
17	(an empty line)	

The end of line character(s) is a newline (0x0a).

Main

The 3rd and the 4th line consist of the following tab (0x09) separated values. Note that the actual values in this category are dependent on the version of EnCase 6.

Identifier number	Character in 3rd line	Value in 4th line
1	a	Unique description
2	c	Case number
3	n	Evidence number
4	e	Examiner name
5	t	Notes
6	md	The model of the media, i.e. hard disk model (introduced in EnCase 6)
7	sn	The serial number of media (introduced in EnCase 6)
8	l	The device label (introduced in EnCase 6.19)
9	av	Version The EnCase version used to acquire the media EnCase limits this value to 12 characters
10	ov	Platform The platform/operating system used to acquire the media
11	m	Acquired date
12	u	System date
13	p	Password hash
14	pid	Process identifier The identifier of the process memory acquired (introduced in EnCase 6.12/Winen 6.11)
15	dc	Unknown
16	ext	Extents The extents of the process memory acquired (introduced in EnCase 6.12/Winen 6.11)

Both the acquiry and system date are empty in a file created by winen.

The date values in the header section (not header2) are set to: Thu Jan 1 00:00:00 1970. Where the time is dependent on the time zone and daylight savings.

For more information see section: Header2 values

Sources

Line 6 the srce category contains information about sources

Line 7 consists of 2 values, namely the values are "0 1".

The 8th line consist of the following tab (0x09) separated values. Note that the actual values in this category are dependent on the version of EnCase 6.

Identifier number	Character in 8th line	Meaning
1	p	
2	n	
3	id	Identifier, unique name

Identifier number	Character in 8th line	Meaning
4	ev	Evidence number
5	tb	Total bytes
6	lo	Logical offset
7	po	Physical offset
8	ah	Acquire hash
9	sh	Unknown (introduced in EnCase 6.19)
10	gu	GUID
11	pgu	Unknown (introduced in EnCase 7)
12	aq	Acquire date

Line 9 consists of 2 values, namely the values are "0 0".

Line 10 contains the values defined by line 8. Note that the default values of some of these values has changed around EnCase 6.12.

Subjects

Line 12 the sub category contains information about sources

Line 13 consists of 2 values, namely the values are "0 1".

The 14th line consist of the following tab (0x09) separated values.

Identifier number	Character in 14th line	Meaning
1	p	
2	n	
3	id	Identifier, unique name
4	nu	Number
5	co	Comment
6	gu	GUID

Line 15 consists of 2 values, namely the values are "0 0".

Line 16 contains the values defined by line 14. Note that the default values of some of these values has changed around EnCase 6.12.

EWF-L01

The EWF-E01 header2 section specification also for the EWF-L01 format. However:

- both the acquired date and system date are not set

Header2 values

Identifier	Description	Notes
a	Unique description	Free form string Note that EnCase might not respond when this value is large e.g. >= 1 MiB

c	Case number	Free form string EnCase limits this string to 3000 - 1 characters
n	Evidence number	Free form string EnCase limits this string to 3000 - 1 characters
e	Examiner name	Free form string EnCase limits this string to 3000 - 1 characters
t	Notes	Free form string EnCase limits this string to 3000 - 1 characters
md	Model	Free form string EnCase limits this string to 3000 - 1 characters
sn	Serial Number	Free form string EnCase limits this string to 3000 - 1 characters
l	Device label	Free form string
av	Version	Free form string EnCase limits this string to 12 - 1 characters
ov	Platform	Free form string EnCase limits this string to 24 - 1 characters
m	Acquired date	String containing Unix 32-bit epoch timestamp E.g. “1142163845”> which represents the date: March 12 2006, 11:44:05
u	System date	String containing Unix 32-bit epoch timestamp E.g. “1142163845”> which represents the date: March 12 2006, 11:44:05
p	Password hash	String containing the password hash. If no password is set it should be simply the character '0'.
pid	Process identifier	String containing the process identifier (pid) number
dc	Unknown	
ext	Extents	extents contains: • number of entries • entries that consist of: S <1> <2> <3>

Note that the restrictions were tested with Encase 7.02.01, older versions could have a restriction of 40 characters instead of 3000 characters.

3.2.2. Header section

The header section is identified in the *section data* type field as “header”. Some aspects of this section are:

- It is defined in the EWF format [ASR02].
- Found in EWF-E01 in EnCase 1 to 7 or linen 5 to 7 or FTK Imager, EWF-L01 in EnCase 5 to 7, and SMART (EWF-S01)
- Found at the start of the first segment file or in EnCase 4 to 7 after the header2 section in the first segment file. Not found in other segment files.

The additional data this section contains is the following

Offset:	number of bytes:	Meaning:
76 (0x4c)	(variable)	Information about the acquired media.

The information about the acquired media consists of zlib compressed data. It contains text in ASCII format specifying information about the acquired media. The text multiple lines separated by an end of line character(s).

In the next paragraphs the various variants of the header section are described. In all cases the information consists of at least the four lines:

Line number	Meaning	Value
1	The number of categories provided	1
2	The name/type of the category provided	main
3	Identifiers for the values in the 4th line	
4	The data for the different identifiers in the 3rd line	

An additional 5th line is found in FTK Imager, EnCase 1 to 7 (EWF-E01). This line is empty.

Line number	Meaning	Value
5		

EWF format

Some aspects of this section are:
- [ASR02] specifies the end of line character(s) is a newline (0x0a).

According to [ASR02] the header contains:

The 3rd and the 4th line consist of the following tab (0x09) separated values.

Identifier number	Character in 3rd line	Value in 4th line
1	c	Case number
2	n	Evidence number
3	a	Unique description
4	e	Examiner name
5	t	Notes
6	m	Acquired date
7	u	System date
8	p	Password hash
9	r	Compression level

For more information see section: Header values

[ASR02] states that the Expert Witness Compression uses 'f', fastest compression.

EnCase 1 (EWF-E01)

Some aspects of this section are:
- The header section is defined only once.
- It is the first section of the first segment file. It is not found in other segment files.

- The header data itself is compressed using zlib.
- The end of line character(s) is a carriage return (0x0d) followed by a newline (0x0a).

The header contains:

The 3rd and the 4th line consist of the following tab (0x09) separated values.

Identifier number	Character in 3rd line	Value in 4th line
1	c	Case number
2	n	Evidence number
3	a	Unique description
4	e	Examiner name
5	t	Notes
6	m	Acquired date
7	u	System date
8	p	Password hash
9	r	Compression level

For more information see section: Header values

SMART (EWF-S01)

Some aspects of this section are:
- The header section is defined once.
- It is the first section of the first segment file. It is not found in other segment files.
- The header data is always processed by zlib, however the same compression level is used as for the chunks. This could mean compression level 0 which is no compression.

The SMART format uses the FTK Imager (EWF-E01) specification for this section.
Note that this could be something FTK Imager specific.

EnCase 2 and 3 (EWF-E01)

Some aspects of this section are:
- The same header section defined twice.
- It is the first and second section of the first segment file. It is not found in other segment files.
- The header data itself is compressed using zlib.
- The end of line character(s) is a carriage return (0x0d) followed by a newline (0x0a).

The header contains:

The 3rd and the 4th line consist of the following tab (0x09) separated values.

Identifier number	Character in 3rd line	Value in 4th line
1	c	Case number
2	n	Evidence number
3	a	Unique description
4	e	Examiner name
5	t	Notes
6	av	Version
7	ov	Platform The platform/operating system used to acquire the media

Identifier number	Character in 3rd line	Value in 4th line
8	m	Acquired date
9	u	System date
10	p	Password hash
11	r	Compression level

For more information see section: Header values

EnCase 4 to 7 (EWF-E01)

Some aspects of this section are:
- The header is defined only once.
- It resides after the header2 sections of the first segment file. It is not found in other segment files.
- The header data itself is compressed using zlib.
- The end of line character(s) is a carriage return (0x0d) followed by a newline (0x0a).

The header contains:

The 3rd and the 4th line consist of the following tab (0x09) separated values.

Identifier number	Character in 3rd line	Value in 4th line
1	c	Case number
2	n	Evidence number
3	a	Unique description
4	e	Examiner name
5	t	Notes
6	av	Version
7	ov	Platform The platform/operating system used to acquire the media
8	m	Acquired date
9	u	System date
10	p	Password hash

For more information see section: Header values

linen 5 to 7 (EWF-E01)

Some aspects of this section are:
- The same header section defined twice.
- It is the first and second section of the first segment file. It is not found in other segment files.
- The header data itself is compressed using zlib.
- The end of line character(s) is a newline (0x0a).

The header information consist of 18 lines

The remainder of the string contains the following information:

Line number	Meaning	Value
1	The number of sections provided	3
2	The name/type of the section provided	main
3	Identifier for the values in the 4th line	

Line number	Meaning	Value
4	The data for the different identifiers in the 3rd line	
5	(an empty line)	
6	The name/type of the section provided	srce
7		
8	Identifier for the values in the section	
9		
10		
11	(an empty line)	
12	The name/type of the section provided	sub
13		
14	Identifier for the values in the section	
15		
16		
17	(an empty line)	

The end of line character(s) is a newline (0x0a).

Main

The 3rd and the 4th line consist of the following tab (0x09) separated values.

Identifier number	Character in 3rd line	Value in 4th line
1	a	Unique description
2	c	Case number
3	n	Evidence number
4	e	Examiner name
5	t	Notes
6	md	The model of the media, i.e. hard disk model (Introduced in linen 6)
7	sn	The serial number of media (Introduced in linen 6)
	l	The device label (Introduced in linen 6.19)
	av	Version
	ov	Platform The platform/operating system used to acquire the media
	m	Acquired date
	u	System date
	p	Password hash
	pid	Process identifier The identifier of the process memory acquired (Introduced in linen 6.19 or earlier)
	dc	Unknown (Introduced in linen 6)
	ext	Extents The extents of the process memory acquired (Introduced in linen 6.19 or earlier)

Note that as of linen 6.19 the acquire date (and time) is in UTC and the system date is in local time.

Where as before both date values were in local time.

For more information see section: Header values

Sources

Line 6 the srce category contains information about sources

Line 7 consists of 2 values, namely the values are "0 1".

The 8th line consist of the following tab (0x09) separated values.

Identifier number	Character in 8th line	Meaning
1	p	
2	n	
3	id	Identifier, unique name
4	ev	Evidence number
5	tb	Total bytes
6	lo	Logical offset
7	po	Physical offset
8	ah	Acquire hash
9	sh	Unknown (Introduced in linen 6.19 or earlier)
10	gu	GUID
11	aq	Acquire date

Line 9 consists of 2 values, namely the values are "0 0".

Line 10 contains the values defined by line 8. Note that the default values of some of these values has changed around linen 6.19 or earlier.

Subjects

Line 12 the sub category contains information about sources

Line 13 consists of 2 values, namely the values are "0 1".

The 14th line consist of the following tab (0x09) separated values.

Identifier number	Character in 14th line	Meaning
1	p	
2	n	
3	id	Identifier, unique name
4	nu	Number
5	co	Comment
6	gu	GUID

Line 15 consists of 2 values, namely the values are "0 0".

Line 16 contains the values defined by line 14. Note that the default values of some of these values has changed around linen 6.19 or earlier.

FTK Imager (EWF-E01)

Some aspects of this section are:
- In FTK Imager (EWF-E01) the same header section defined twice.
- It is the first and second section of the first segment file. It is not found in other segment files.
- The header data itself is compressed using zlib. Note that the compression level can be none and therefore the header looks uncompressed.
- In FTK Imager the end of line character(s) is a newline (0x0a).

In FTK Imager (EWF-E01) the header contains:

The 3rd and the 4th line consist of the following tab (0x09) separated values.

Identifier number	Character in 3rd line	Value in 4th line
1	c	Case number
2	n	Evidence number
3	a	Unique description
4	e	Examiner name
5	t	Notes
6	av	Version The FTK Imager version used to acquire the media
7	ov	Platform The platform/operating system used to acquire the media
8	m	Acquired date
9	u	System date
10	p	Password hash
11	r	char

For more information see section: Header values

EnCase 5 to 7 (EWF-L01)

The EWF-E01 header section specification also for the EWF-L01 format. However:
- In EnCase 5 both the acquired date and system date are set to 0.
- In EnCase 6 and 7 both the acquired date and system date are set to Jan 1, 1970 00:00:00 (the time is dependent on the local timezone and daylight savings)

Header values

Identifier	Description	Notes
a	Unique description	Free form string Note that EnCase might not respond when this value is large e.g. >= 1 MiB
c	Case number	Free form string EnCase limits this string to 3000 - 1 characters
n	Evidence number	Free form string EnCase limits this string to 3000 - 1 characters
e	Examiner name	Free form string EnCase limits this string to 3000 - 1 characters

t	Notes	Free form string EnCase limits this string to 3000 - 1 characters
md	Model	Free form string EnCase limits this string to 3000 - 1 characters
sn	Serial Number	Free form string EnCase limits this string to 3000 - 1 characters
l	Device label	Free form string
av	Version	Free form string EnCase limits this string to 12 - 1 characters
ov	Platform	Free form string EnCase limits this string to 24 -1 characters
m	Acquired date	In EnCase: String containing a date and time value E.g. 2002 3 4 10 19 59", which represents March 4, 2002 10:19:59. In linen: String containing Unix 32-bit epoch timestamp E.g. "1142163845"> which represents the date: March 12 2006, 11:44:05
u	System date	In EnCase: String containing a date and time value E.g. 2002 3 4 10 19 59", which represents March 4, 2002 10:19:59. In linen: String containing Unix 32-bit epoch timestamp E.g. "1142163845"> which represents the date: March 12 2006, 11:44:05
p	Password hash	String containing the password hash. If no password is set it should be simply the character '0'.
pid	Process identifier	String containing the process identifier (pid) number
dc	Unknown	
ext	Extents	extents contains: • number of entries • entries that consist of: S <1> <2> <3>
r	Compression	Single character that represent the compression level

Note that the restrictions were tested with Encase 7.02.01, older versions could have a restriction of 40 characters instead of 3000 characters.

Value of char	**Meaning**
b	Best compression is used
f	Fastest compression is used
n	No compression is used

Notes

Note: there should not be tab, carriage return and newline characters within the text in the 4th line. Or is there a method to escape these characters? [ASR02] states that these characters should not be used in the free form text. Need to confirm this, the specification only speaks of a newline character.

Note: currently the password has no a additional value than allow an application check it. The data itself is not protected using the password.
The password hashing algorithm is unknown. Need to find out. And does the algorithm differ per EnCase version? probably not. The algorithm does not differ in EnCase 1 to 7. FTK Imager does not bother with a password.

3.2.3. Volume section

The volume section is identified in the *section data* type field as “volume”. Some aspects of this section are:

- Defined in the EWF format [ASR02].
- Found in EWF-E01 in EnCase 1 to 7 or linen 5 to 7 or FTK Imager, EWF-L01 in EnCase 5 to 7, and SMART (EWF-S01)
- Found after the header section of the first segment file. Not found in other segment files.

In the next paragraphs the various versions of the volume section are described.

EWF specification

The specification according to [ASR02].

The additional volume section data is 94 bytes of size and consists of:

offset	**size**	**value**	**description**
0	4		Reserved according to [ASR02] Contains 0x01 Reserved for what?
4	4		The chunk count *Contains the number of chunks within the all segment files.*
8	4		The number of sectors per chunk Contains 64 per default.
12	4		The number of bytes per sectors Contains 512 per default
16	4		The sectors count, the number of sectors within all *segment files*
20	20	0x00	Reserved Reserved for what?
40	45	0x00	Padding
85	5		Signature (Reserved) Contains the EWF file header signature
90	4		Checksum

offset	size	value	description
			Adler-32 of all the previous data within the additional volume section data.

The chunk count is a 32-bit value this means it maximum of addressable chunks would be: 4294967295 (= 2^32 – 1). For a chunk size of 32768 x 4294967295 = about 127 Tb. The maximum segment file amount is 2^16 - 1 = 65535. This allows for an equal number of storage if a segment file is filled to its maximum number of chunks.

However libewf is restricted at 14295 segment files, due to the extension naming schema of the segment files.

SMART (EWF-S01)

The SMART format uses the EWF specification for this section.

In SMART the signature (reverse) value is the string “SMART” (0x53 0x4d 0x41 0x52 0x54) instead of the file header signature.

FTK Imager, EnCase 1 to 7 and linen 5 to 7 (EWF-E01)

The specification for FTK Imager, EnCase 1 to 7 and linen 5 to 7.

The additional volume section data is 1052 bytes of size and consists of:

offset	size	value	description
0	1		Media type See section: Media type
1	3	0x00	Unknown (empty values)
4	4		The chunk count Contains the number of chunks within the all *segment files*.
8	4		The number of sectors per chunk (or block size) Contains 64 per default. EnCase 5 is the first version which allows this value to be different than 64.
12	4		The number of bytes per sector
16	8		The sectors count Contains the number of sectors within all *segment files* This value probably has been changed in EnCase 6 from a 32-bit value to a 64-bit value to support media >2TiB
24	4		The number of cylinders of the C:H:S value Most of the time this value is empty (0x00)
28	4		The number of heads of the C:H:S value

offset	size	value	description
			Most of the time this value is empty (0x00)
32	4		The number of sectors of the C:H:S value Most of the time this value is empty (0x00)
36	1		Media flags See section: Media flags
37	3	0x00	Unknown (empty values)
40	4		PALM volume start sector
44	4	0x00	Unknown (padding/empty values)
48	4		SMART logs start sector Contains an offset relative from the end of media E.g. a value of 10 would refer to sector = number of sectors – 10
52	1		Compression level (Introduced in EnCase 5) See section: Compression level
53	3	0x00	Unknown (empty values) these values seem to be part of the compression type
56	4		The sector error granularity Contains the error block size (Introduced in EnCase 5)
60	4	0x00	Unknown (empty values)
64	16		Segment file set identifier Contains a GUID/UUID generated on the acquiry system probably used to uniquely identify a set of segment files (Introduced in EnCase 5)
80	963		Unknown (padding/empty values)
1043	5		Signature (Reserved) Contains 0x00
1048	4		Checksum Adler-32 of all the previous data within the additional volume section data.

A value that could be in the volume is the raid stripe size

EnCase requires for media that contains no partition table that the is physical media flag is not set and vice versa. FTK checks the media data.

EnCase 5 to 7 (EWF-L01)

The EWF-L01 format uses the EnCase 5 (EWF-E01) volume section specification. However:

- the volume type contains 0x0e
- the number of chunks is 0
- The number of bytes per sectors is some kind of block size value (4096), perhaps the source file system block size
- The sectors count, represents some other value because (sector_size * sector_amount != total_size) the total size is in the ltree section

Media type

Value	Identifier	Description
0x00		A removable storage media device
0x01		A fixed storage media device
0x03		An optical disc (CD/DVD/BD)
0x0e		Logical Evidence File (LEV or L01)
0x10		Physical Memory (RAM)

FTK imager versions, before 2.9?, set the storage media to fixed (0x01).

Media flags

Value	Identifier	Description
0x01		Is an image file in FTK Imager, EnCase 1 to 7 this bit is always set, when not set EnCase seems to see the image file as a device
0x02		Is physical device or device type 0 => a non physical device (logical) 1 => a physical device
0x04		Fastbloc write blocker used
0x08		Tableau write blocker used This was added in EnCase 6.13

If both the the Fastbloc and Tableau write blocker media flags are set EnCase only shows the Fastbloc.

Compression level

Value	Identifier	Description
0x00		no compression
0x01		good compression
0x02		best compression

3.2.4. Disk section

The disk section is identified in the *section data* type field as “disk”. Some aspects of this section are:

- Not defined in the EWF format [ASR02].
- Not found in SMART (EWF-S01).
- It is found in FTK Imager, EnCase 1 to 7 and linen 5 to 7 (EWF-E01) files.

According to [COH] the disk section is the same as the volume section. This was confirmed with a disk section in an FTK Imager 2.3 (EWF-E01) image.

Note: the disk was found only in FTK Imager 2.3 when acquiring a physical disk not a floppy. This requires additional research. Is the disk section some old method to differentiate between a partition (volume) image or a physical disk image?

3.2.5. Data section

The data section is identified in the *section data* type field as “data”. Some aspects of this section are:

- It is not defined in the EWF format [ASR02].
- Found in EWF-E01 in EnCase 1 to 7 or linen 5 to 7 or FTK Imager, and EWF-L01 in EnCase 5 to 7. Not found in SMART (EWF-S01).
- For multiple segment files it does not reside in the first segment file. For a single segment file it does.
- Found after the last table2 section in a single segment file or for multiple segment files at the start of the segment files, except for the first.
- The data section has data it should should contain the same information as the volume section.

FTK Imager, EnCase 1 to 7 and linen 5 to 7 (EWF-E01)

The additional data section data is 1052 bytes of size and consists of:

offset	**size**	**value**	**description**
0	1		Media type See section: Media type
1	3	0x00	Unknown (empty values)
4	4		The chunk count Contains the number of chunks within the all *segment files*.
8	4		The block size (number of sectors per chunk) Contains 64 per default. EnCase 5 is the first version which allows this value to be different than 64.
12	4		The number of bytes per sector
16	8		The sectors count Contains the number of sectors within all *segment files* This value probably has been changed in

offset	size	value	description
			EnCase 6 from a 32-bit value to a 64-bit value to support media >2TiB
24	4		The number of cylinders of the C:H:S value Most of the time this value is empty (0x00)
28	4		The number of heads of the C:H:S value Most of the time this value is empty (0x00)
32	4		The number of sectors of the C:H:S value Most of the time this value is empty (0x00)
36	1		Media flags See section: Media flags
37	3	0x00	Unknown (empty values)
40	4		PALM volume start sector
44	4	0x00	Unknown (padding/empty values)
48	4		SMART logs start sector Contains an offset relative from the end of media E.g. a value of 10 would refer to sector = number of sectors – 10
52	1		Compression level (Introduced in EnCase 5) See section: Compression level
53	3	0x00	Unknown (empty values) these values seem to be part of the compression type
56	4		The sector error granularity Contains the error block size (Introduced in EnCase 5)
60	4	0x00	Unknown (empty values)
64	16		Segment file set identifier Contains a GUID/UUID generated on the acquiry system probably used to uniquely identify a set of segment files (Introduced in EnCase 5)
80	963		Unknown (padding/empty values)
1043	5		Signature (Reserved) Contains 0x00
1048	4		Checksum Adler-32 of all the previous data within the additional data section data.

Notes:

- In Logicube products (Talon, Forensic dossier) the checksum is not calculated and set to 0.

EnCase 5 to 7 (EWF-L01)

The EWF-L01 format uses the EnCase 5 (EWF-E01) data section specification. However:

- the data type contains 0x0e
- the number of chunks is 0
- The number of bytes per sectors is some kind of block size value (4096), perhaps the source file system block size
- The sectors count, represents some other value because (sector_size * sector_amount != total_size) the total size is in the ltree section

3.2.6. Sectors section

The sectors section is identified in the *section data* type field as “sectors”. Some aspects of this section are:

- Not defined in the EWF format [ASR02].
- Found in EWF-E01 in EnCase 2 to 7, or linen 5 to 7 or FTK Imager, EWF-L01 in EnCase 5 to 7. Not found in EnCase 1 (EWF-E01) or SMART (EWF-S01).
- The first sectors section can be found after the volume section in the first segment file or at the after the data section in other segment files. Successive sector data sections are found after the sector table2 section.

The sectors section contains the actual chunks of media data.

- The sectors section can contain multiple chunks.
- The default size of a chunk is 32k byte of data (64 standard sectors, with a size of 512 bytes per sector).
  It is possible in EnCase 5 and 6 and linen 5 and 6 to change the number of sectors per block to 64, 128, 256, 1024, 2048, 4096, 8192, 16384 or 32768. In EnCase 7 and linen 7 this has been reduced to 64, 128, 256, 1024.

When the evidence file is compressed a segment file can contain, at the same time, both compressed and uncompressed chunks. If the compressed data (with checksum) is larger than the uncompressed data (without the checksum) the chunk is left uncompressed.

The first chunk is often located directly after the section descriptor, although the format does not require this.

Data chunk

A data chunk is variable of size and consists of:

offset	size	value	description
0	...		Chunk data compressed chunk of the storage media data
...	4		Checksum Adler-32 of the chunk data

Note that a compressed chunk data the checksum actually is part of the deflate/RFC1951 data

Chunk data contains either compressed or uncompressed data, the size of chunk data should not be larger then the default chunk size.

Optical disc images

For a MODE-1 CD-ROM optical disc image EnCase only seems to support 2048 bytes per sector (the data).

The raw sector size of a MODE-1 CD-ROM is 2352 bytes of size and consists of:

offset	size	value	description
0	16		Synchronization bytes
16	2048		Data
2054	4		Error detection
2058	8		Empty values
2066	276		Error correction

TODO Mode-2 and Mode-XA

3.2.7. Table section

The table section is identified in the section data type field as “table”. Some aspects of this section are:

- Defined in the EWF format [ASR02].
- Found in EWF-E01 in EnCase 1 to 7 or linen 5 to 7 or FTK Imager, EWF-L01 in EnCase 5 to 7, and SMART (EWF-S01)

Note that the offsets within the section descriptor are 8 bytes (64 bits) of size while the offsets in the table entry array are 4 bytes (32 bits) of size.

In the next paragraphs the various versions of the table section are described.

EWF specification

Some aspects of this section are:

- The first table section resides after the volume section in the first segment file or after the file header in other segment files.
- It can be found in every segment file.

The table section consists of:

- the table header
- an array of table entries
- the data chunks

Table header

The table header is 24 bytes of size and consists of:

offset	size	value	description
0	4		The number of entries Note that according to [ASR02] it contains 0x01

offset	size	value	description
4	16	0x00	Padding
20	4		Checksum Adler-32 of all the previous data within the additional volume section data.

According to [ASR02] the table can hold 16375 entries if more entries are required an additional table section should be created.

Table entry

The table entry is 4 bytes of size and consists of:

offset	size	value	description
0	4		Chunk data offset

The most significant bit (MSB) in the chunk data offset indicates if the chunk is compressed (1) or uncompressed (0).

A chunk data offset points to the start of the chunk of media data, which resides in the same table section within the segment file. The offset contains a value relative to the start of the file.

Data chunk

A data chunk is variable of size and consists of:

offset	size	value	description
0	...		Chunk data compressed chunk of the storage media data
...	4		Checksum Adler-32 of the chunk data

Note that a compressed chunk data the checksum actually is part of the deflate/RFC1951 data

The default size of a chunk is 32k byte of data (64 standard sectors).

A chunk is always processed by deflate even when no compression is required. This provides for a checksum for each chunk. Therefore the size of a chunk data can be larger than the default chunk size. This however was deducted from the behavior of FTK Imager for EWF-S01

SMART (EWF-S01)

The SMART format uses the EWF specification for this section.

EnCase 1 (EWF-E01)

Some aspects of this section are:

- The table section resides after the volume section in the first segment file or after the file header

in other segment files.
- It can be found in every segment file.

The table section consists of:
- the table header
- an array of table entries
- the table footer
- the data chunks

Table header

The table header is 24 bytes of size and consists of:

offset	size	value	description
0	4		The number of entries
4	16	0x00	Padding
20	4		Checksum Adler-32 of all the previous data within the additional volume section data.

The table can hold 16375 entries if more entries are required an additional table section should be created.

Table entry

The table entry is 4 bytes of size and consists of:

offset	size	value	description
0	4		Chunk data offset

The most significant bit (MSB) in the chunk data offset indicates if the chunk is compressed (1) or uncompressed (0).

A chunk data offset points to the start of the chunk of media data, which resides in the same table section within the segment file. The offset contains a value relative to the start of the file.

Table footer

The table footer is 4 bytes of size and consists of:

offset	size	value	description
0	4		Checksum Adler-32 of the offset array

Data chunk

A data chunk is variable of size and consists of:

offset	size	value	description
0	...		Chunk data

offset	size	value	description
			compressed chunk of the storage media data
...	4		Checksum Adler-32 of the chunk data

Note that a compressed chunk data the checksum actually is part of the deflate/RFC1951 data

Chunk data contains either compressed or uncompressed data, the size of chunk data should not be larger then the default chunk size.

The default size of a chunk is 32k byte of data (64 standard sectors).

FTK Imager and EnCase 2 to 5 and linen 5 (EWF-E01)

Some aspects of this section are:
- The table section resides after the sectors section.
- It can be found in every segment file.
- The data chunks are no longer stored in this section but in the sectors section instead.
- The table2 section contains a mirror copy of the table section. In EWF-E01 it is always present.

The table section consists of:
- the table header
- an array of table entries
- the table footer

Table header

The sector table header is 24 bytes of size and consists of:

offset	size	value	description
0	4		The number of entries
4	16	0x00	Padding
20	4		Checksum Adler-32 of all the previous data within the additional volume section data.

The table section can hold 16375 entries. A new table section should be created to hold more entries. Both FTK Imager and EnCase 5 can handle more than 16375, FTK 1 cannot. To contain more than 16375 chunks new sectors, table and table2 sections need to be created after the table2 section.

Table entry

The table entry is 4 bytes of size and consists of:

offset	size	value	description
0	4		Chunk data offset

The most significant bit (MSB) in the chunk data offset indicates if the chunk is compressed (1) or

uncompressed (0).

A chunk data offset points to the start of the chunk of media data, which resides in the preceding sectors section within the segment file. The offset contains a value relative to the start of the file.

Table footer

The table footer is 4 bytes of size and consists of:

offset	size	value	description
0	4		Checksum Adler-32 of the offset array

EnCase 6 to 7 and linen 6 to 7 (EWF-E01)

Some aspects of this section are:

- Every segment file contains its own table section.
- It resides after the sectors section.
- The data chunks are no longer stored in this section but in the sectors section instead.
- The table2 section contains a mirror copy of the table section. In EWF-E01 it is always present.

The table section consists of:

- the table header
- an array of table entries
- the table footer

Table header

The sector table header is 24 bytes of size and consists of:

offset	size	value	description
0	4		The number of entries
4	4	0x00	Padding
8	8		The table base offset
16	4	0x00	Padding
20	4		Checksum Adler-32 of all the previous data within the additional volume section data.

As of EnCase 6 the number of entries is no longer restricted to 16375 entries. The new limit seems to be 65534.

Table entry

The table entry is 4 bytes of size and consists of:

offset	size	value	description
0	4		Chunk data offset

The most significant bit (MSB) in the chunk data offset indicates if the chunk is compressed (1) or uncompressed (0).

A chunk data offset points to the start of the chunk of media data, which resides in the preceding sectors section within the segment file. The offset contains a value relative to the start of the file.

In EnCase 6.7.1 the sectors section can be larger than 2048Mb. The table entries offsets are 31 bit values in EnCase6 the offset in a table entry value will actually **use the full 32 bit** if the 2048Mb has been exceeded. This behavior is no longer present in EnCase 6.8 so it is assumed to be a bug. Libewf currently assumes that the if the 31 bit value overflows the following chunks are uncompressed. This allows EnCase 6.7.1 faulty EWF files to be converted by libewf.

Table footer

The table footer is 4 bytes of size and consists of:

offset	size	value	description
0	4		Checksum Adler-32 of the offset array

EnCase 6 to 7 (EWF-L01)

The EWF-L01 format uses the EnCase 6 to 7 (EWF-E01) table section specification.

3.2.8. Table2 section

The table2 section is identified in the *section data* type field as “table2”. Some aspects of this section are:

- Not defined in the EWF format [ASR02].
- Found in EWF-E01 in EnCase 2 to 7, or linen 5 to 7 or FTK Imager, EWF-L01 in EnCase 5 to 7. Not found in EnCase 1 (EWF-E01) or SMART (EWF-S01).
- Uses the same format as the table section.
- Resides directly after the table section.

FTK Imager and EnCase 2 to 7 and linen 5 to 7 (EWF-E01)

The table2 section contains a mirror copy of the table section. Probably intended for recovery purposes.

EnCase 5 to 7 (EWF-L01)

The EWF-L01 format uses the EWF-E01 table2 section specification.

3.2.9. Next section

The next section is identified in the *section data* type field as “next”. Some aspects of this section are:

- Defined in the EWF format [ASR02].
- Found in EWF-E01 in EnCase 1 to 7 or linen 5 to 7 or FTK Imager, EWF-L01 in EnCase 5 to 7,

and SMART (EWF-S01)
- The last section within a segment other than the last segment file.
- The offset to the next section in the section descriptor of the next section point to itself (the start of the next section).
- It should be the last section in a segment file, other than the last segment file.

SMART (EWF-S01)

It resides after the table or table2 section.

FTK Imager, EnCase and linen (EWF-E01)

It resides after the data section in a single segment file or for multiple segment files after the table2 section.

In the EnCase (EWF-E01) format the size in the section descriptor is 0 instead of 76 (the size of the section descriptor).

Note that FTK imager versions, before 2.9?, set the section size to 76.

3.2.10. Ltypes section

The ltypes section is identifier in the *section data* type field as “ltypes”. Some aspects of this section are:
- Found in EWF-L01 in of EnCase 7
- Found in the last segment file after table2 section before tree section.

The additional ltypes section data is 6 bytes of size and consists of:

offset	**size**	**value**	**description**
0	2		Unknown
2	2		Unknown
4	2		Unknown

3.2.11. Ltree section

The ltree section is identifier in the *section data* type field as “ltree”. Some aspects of this section are:
- Found in EWF-L01 in of EnCase 5 to 7
- Found in the last segment file after ltypes section and before data section.

The ltree section consists of:
- ltree header
- ltree data

Ltree header

The ltree header is 6 bytes of size and consists of:

offset	size	value	description
0	16		Integrity hash Contains the MD5 of the ltree data
16	4		Data size
20	4		Unknown (empty values)
24	4		Checksum Adler-32 of all the data within the ltree header where the checksum value itself is zeroed out.
28	20		Unknown (empty values)

Ltree data

The ltree data string consists of an UTF-16 little-endian encoded string without the UTF-16 endian byte order mark.

The ltree data string contains the following information:

Line number	Meaning	Value
1	The number of sections provided	5
2	Probably the type of information provided	rec
3	Identifier for the values in the 4th line	
4	The data for the different identifiers in the 3rd line	
5	(an empty line)	
6	Information about single file permissions	perm
7		
8	Identifier for the values in the section	
9		
10		
11		
12		
13	(an empty line)	
14	Probably the type of information provided (the data source)	srce
15		
16	Identifier for the values in the section	
17		
18		
19		
20		
21	(an empty line)	
22	Probably the type of information provided	sub
23		
24	Identifier for the values in the section	
25		
26	(an empty line)	
27	Information about single file entries	entry
28		
29	Identifier for the values in the section	
30	The entries of files and directories	
...	(an empty line)	

Note that the actual line numbering can vary.

The end of line character(s) is a newline (0x0a).

Records

The rec category contains information about records.

The 3rd and the 4th line consist of the following tab (0x09) separated values.

Identifier number	Character in 3rd line	Value in 4th line
1	tb	Total bytes The size of the logical file data (media data)
2	cl	Clusters ???
3	n	Unknown (introduced in EnCase 6.19)
4	fp	Unknown (introduced in EnCase 7)
5	pg	Unknown (introduced in EnCase 7)
6	lg	Unknown (introduced in EnCase 7)
7	ig	Unknown (introduced in EnCase 7)

Permissions

The perm section contains information about single file permissions.

Line 6 consist of perm

Line 7 consists of 2 values.

The 8th line consist of the following tab (0x09) separated values.

Identifier number	Character in 8th line	Meaning
1	p	
2	n	Name
3	s	NT security identifier (SID)
4	pr	Property
5	nta	NT permission (ACE) ???
6	nti	Permission ???
7	nts	Permission ??? (Removed in EnCase 6)

Property: (2 => allow, empty => owner, 1 => group)

Sources

Line 12 the srce category contains information about sources

Line 13 consists of 2 values, namely the values are "0 1".

The 14th line consist of the following tab (0x09) separated values.

Identifier number	Character in 9th line	Meaning
1	p	Unknown
2	n	Unknown
3	id	Identifier, unique name
4	ev	Evidence number
5	tb	Total bytes
6	lo	Logical offset -1 when not set
7	po	Physical offset -1 when not set
8	ah	Acquire hash
9	sh	Unknown (introduced in EnCase 6.19)
10	gu	GUID
11	pgu	Unknown (introduced in EnCase 7)
12	aq	Acquire date

- "Acquire date" is in the form of: "1142163845", which is a Unix epoch time stamp and represents the date: March 12 2006, 11:44:05.

Subjects

sub probably are subjects within EnCase

The 21th line consist of the following tab (0x09) separated values.

Identifier number	Character in 15th line	Meaning
1	p	Unknown
2	n	Unknown
3	id	Identifier, unique name
4	nu	Number
5	co	Comment
6	gu	GUID

Entries

EnCase 5 and 6 (EWF-L01)

The 29th line consist of the following tab (0x09) separated values.

Identifier number	Character in 29th line	Meaning
1	p	Is parent 1 => if the single file entry is a directory (null) => if single file entry is a file
2	n	Name
3	id	Unknown
4	opr	Flags See section: Single file entry flags
5	src	Possible the source identifier

Identifier number	Character in 29th line	Meaning
6	sub	Possible the subject identifier
7	cid	Unknown
8	jq	Unknown
9	cr	Creation date
10	ac	Access date (precision is date only)
11	wr	(File) modification (last written) date
12	mo	(File system) entry modification date
13	dl	Unknown
14	aq	Unknown
15	ha	Hash The MD5 hash of the file data
16	ls	File size The file size specified in bytes If the file size is 0 the data size should be 1
17	du	Duplicate data offset Relative from the start of the media data
18	lo	Unknown
19	po	The segment file in which the start of the data is stored, -1 for a single segment file ?
20	mid	Unknown (identifier?) (introduced in EnCase 6.19)
21	cfi	Unknown (introduced in EnCase 6.14)
22	be	Contains 3 values separated by a space: “Unknown Offset Size” Where: • unknown always 1 (number of extents?) • data offset, relative from the start of the media data • data size The offset and size are specified in hexadecimal values. Contains 1 value for the first single file entry.
23	pm	permissions index? -1 has a special meaning?
24	lpt	Unknown (introduced in EnCase 6.19)

- “Creation date”, “Access date” and “ Last written date” are in the form of: “1142163845”, which is a Unix epoch time stamp and represents the date: March 12 2006, 11:44:05.
- “Hash” consist of a MD5 hash string. Only file entries are hashed.

If the “ha” value contains “00000000000000000000000000000000.” this means the MD5 hash is not set.

3.2.11.1. Single file entries

The entries of files and directories consist of entries starting with: 0 followed by the number of sub file entries.

The entries of files and directories:

Line number	Meaning	Value
1	The root directory	(empty)
2	The target drive/mount point	
3	The actual single file entries	

EnCase 7 (EWF-L01)

The 29th line consist of the following tab (0x09) separated values.

Identifier number	Character in 29th line	Meaning
1	mid	Unknown (identifier?)
2	ls	File size The file size specified in bytes If the file size is 0 the data size should be 1
3	be	Contains 3 values separated by a space: “Unknown Offset Size” Where: • unknown always 1 (number of extents?) • data offset, relative from the start of the media data • data size The offset and size are specified in hexadecimal values. Contains 1 value for the first single file entry.
4	id	Unknown
5	cr	Creation date
6	ac	Access date
7	wr	(File) modification (last written) date
8	mo	(File system) entry modification date
9	dl	Unknown
10	sig	Unknown (Introduced in EnCase 7)
11	ha	Hash The MD5 hash of the file data
12	sha	SHA1 hash Judging by the size this value is assumed to be the SHA1 hash of the file data, does not seem to be currently set by EnCase (Introduced in EnCase 7)
13	p	Is parent 1 => if the single file entry is a directory (null) => if single file entry is a file
14	n	Name

Identifier number	Character in 29th line	Meaning
15	du	Duplicate data offset Relative from the start of the media data
16	lo	Unknown
17	po	The segment file in which the start of the data is stored, -1 for a single segment file ?
18	pm	permissions index? -1 has a special meaning?
19	oes	Unknown (Introduced in EnCase 7)
20	opr	Flags See section: Single file entry flags
21	src	Possible the source identifier
22	sub	Possible the subject identifier
23	cid	Unknown
24	jq	Unknown
25	alt	Unknown (Introduced in EnCase 7)
26	ep	Unknown (Introduced in EnCase 7)
27	aq	Unknown
28	cfi	Unknown
29	sg	Unknown (Introduced in EnCase 7)
30	lpt	Unknown

If the “ha” value contains “00000000000000000000000000000000.” this means the MD5 hash is not set. The same applies for the “sha” value when it contains “0000000000000000000000000000000000000000” the SHA1 has is not set.

3.2.11.1. Single file entries

The entries of files and directories consist of entries starting with: 26 followed by the number of sub file entries.

The entries of files and directories:

Line number	Meaning	Value
1	The root directory	LogicalEntries
2	The target drive/mount point	
3	The actual single file entries	

Single file entry flags

Value	Identifier	Description
0x00000008		Archive
0x00400000		is file?

Value	Identifier	Description
0x01000000		
0x02000000		
0x04000000		Data is sparse See remarks below.

If the sparse data flag is set:
- the data size should be 1 and data should consist of a single byte value.
- the data size should be equal to the file size and data should be the same.

If the duplicate data offset value is not set the single byte value in the data should be used to reconstruct the file data. E.g. if the file size is 4096 and the data contains the byte value 0x00 the resulting file should consists of 4096 x 0x00 byte values.

If the duplicate data offset value is set the single byte in the data is ignored and the duplicate data offset refers to the location where the data stored.

3.2.12. Map section

Some aspects of this section are:
- Found in EWF-L01 in of EnCase 7 (First seen in EnCase 7.4.1.10)
- Found in the last segment file after data section before done section.

The map consists of:
- map string
- map entries array

Map string

The map string consists of an UTF-16 little-endian encoded string without the UTF-16 endian byte order mark.

The map string contains the following information:

Line number	Meaning	Value
1	The number of sections provided	1
2	Probably the type of information provided	r
3	Identifier for the values in the 4th line	c
4	The data for the different identifiers in the 3rd line	
5	(an empty line)	

Map string values

Identifier number	Character in 29th line	Meaning
1	C`	Number of map entries (count)

The number of map entries should match the number of single file entries in the ltree.

Map entry

A map entry is 24 bytes of size and consists of:

offset	size	value	description
0	4		Unknown
4	4		Unknown (empty values) Or part of previous value
8	16		Unknown

3.2.13. Session section

The session section is identifier in the section data type field as “session”. Some aspects of this section are:

- It is not defined in the EWF format [ASR02].
- It is not found in SMART (EWF-S01) and FTK Imager (EWF-E01).
- It is found in EnCase 5 and 6 (EWF-E01) files.
- It is only added to the last segment file for images of optical disc (CD/DVD/BD) media.
- It is found after the data section and before the error2 section.

The session section data consists of:

- The session header
- The session entries array
- The session footer

Session header

The session header is 36 byte of size and consists of:

offset	size	value	description
0	4		Number of sessions
4	28	0x00	Unknown (empty values)
32	4		Checksum Adler-32 of all the previous data within the additional session section data.

Session entry

A session entry is 32 byte of size and consists of:

offset	size	value	description
0	4		Flags
4	4		First sector
8	24	0x00	Unknown (empty values)

For a CD the first session sector is stored as 16, although the actual session starts at sector 0. Could

this value be overloaded to indicate the size of the reserved space between the start of the session and the ISO 9660 volume descriptor.

EnCase stores audio tracks as 0 byte data with a sector size of 2048.

Session flags

Value	Identifier	Description
0x00000001		If set the track is an audio track otherwise the track is a data track

Session footer

The session footer is 4 byte of size and consists of:

offset	size	value	description
0	4		Checksum Adler-32 of all the data within the session entries array

3.2.14. Error2 section

The error2 section is identifier in the section data type field as “error2”. Some aspects of this section are:

- It is not defined in the EWF format [ASR02].
- It is not found in SMART (EWF-S01).
- It is found in, EnCase 3 to 7 and linen 5 to 7 (EWF-E01) files.
- It is only added to the last segment file when errors were encountered while reading the input.

Still have to test FTK Imager, Encase 1 and 2 for presence of the error2 section.

It contains the sectors that have read errors. The sector where a read error occurred are filled with zero's during acquiry by EnCase.

The error2 section data consists of:

- The error2 header
- The error2 entries array
- The error2 footer

Error2 header

The error2 header is 520 byte of size and consists of:

offset	size	value	description
0	4		Number of entries
4	512	0x00	Unknown (empty values)
516	4		Checksum Adler-32 of all the previous data within the

offset	size	value	description
			error2 header data.

Error2 entry

An error2 entry is 8 byte of size and consists of:

offset	size	value	description
0	4		First sector
4	4		The number of sectors

Error2 footer

The error2 footer is 4 byte of size and consists of:

offset	size	value	description
0	4		Checksum Adler-32 of all the data within the error2 entries array

3.2.15. Digest section

The digest section is identified in the section data type field as “digest”. Some aspects of this section are:

- It is found in EnCase 6 to 7 files, as of Encase 6.12 and linen 6.12 (EWF-E01).

The digest section contains a MD5 and/or SHA1 hash of the data within the chunks.

The additional digest section data is 80 byte of size and consists of:

offset	size	value	description
0	16		MD5 hash of the media data
16	20		SHA1 hash of the media data
36	40	0x00	Padding
76	4		Checksum Adler-32 of all the previous data within the additional digest section data.

3.2.16. Hash section

The hash section is identified in the section data type field as “hash”. Some aspects of this section are:

- It is defined in the EWF format [ASR02].
- It is found in SMART (EWF-S01) and FTK Imager, EnCase 1 to 7 and linen 5 to 7 (EWF-E01) files.
- It is not found in EnCase 5 (EWF-L01).
- The hash section is optional, it does not need to be present. If it does it resides in the last segment file before the done section.

The hash section contains a MD5 hash of the data within the chunks.

The additional digest section data is 36 byte of size and consists of:

offset	size	value	description
0	16		MD5 hash of the media data
16	16		Unknown
32	4		Checksum Adler-32 of all the previous data within the additional hash section data.

The unknown:
- is zero in SMART
- is zero in EnCase 3 and below
- in EnCase 4 the first 4 bytes are 0, the next 8 bytes seem random, the last 4 bytes seem fixed
- in EnCase 5 and 6 the first 8 bytes seem random, the last 8 bytes equal the file header signature
- in linen 5 the first and last set of 4 bytes seem the same, the second set of 4 bytes seem to be random, the third set of 4 bytes seem to contain a piece of the file header signature
- in linen 6 the first and third set of 4 bytes seem random, the second and last set of 4 bytes seem to be the same
- EnCase5 seems to contain a GUID of the acquired device?

Test with EnCase 4 show that:
- The value does not equal the checksum of the media data
- Does not differentiate for the same media acquired within the same program session, using different formats, but differ for different media and different program sessions

3.2.17. Done section

The done section is identified in the section data type field as “done”. Some aspects of this section are:
- It is defined in the EWF format [ASR02].
- It is found in SMART (EWF-S01), FTK Imager, EnCase 1 to 7 and linen 5 to 7 (EWF-E01) and EnCase 5 (EWF-L01) files.
- The done section is the last section within the last segment file.
- The offset to the next section in the section descriptor of the done section point to itself (the start of the done section).
- It should be the last section in the last segment file.

SMART (EWF-S01)

It resides after the table or table2 section.

FTK Imager, EnCase and linen (EWF-E01)

It resides after the data section in a single segment file or for multiple segment files after the table2 section.

In the EnCase (EWF-E01) format the size in the section descriptor is 0 instead of 76 (the size of the section descriptor).

Note that FTK imager versions, before 2.9?, set the section size to 76.

3.2.18. Incomplete section

The incomplete section is identified in the section data type field as “incomplete”.

This section is seen rarely. It was seen in an EnCase 6.13 (EWF-E01) file as the last last section within the last segment file. The incomplete section was preceded by a hash and digest section, although later in the set of EWF files another hash and digest section were defined.

It is currently assumed that the incomplete section indicates an incomplete image created using remote imaging. The incomplete section contains data but currently there is no indication what purpose the data has.

3.3. Calculating the checksum

The checksum algorithm provided by [ASR02], slightly altered for readability. The algorithm used is Alder-32 and [ASR02] incorrectly refers to it as a CRC.

```
uint32_t Expert_Witness_Compression_CRC(
          uint8_t *buffer,
          size_t buffer_size,
          uint32_t previous_key )
{
          size_t buffer_iterator = 0;
          uint32_t lower_word    = previous_key & 0xffff;
          uint32_t upper_word    = ( previous_key >> 16 ) & 0xffff;

          for( buffer_iterator = 0;
               buffer_iterator < buffer_size;
               buffer_iterator++ )
          {
                    lower_word += buffer[ buffer_iterator ];
                    upper_word += lower_word;

                    if( ( buffer_iterator != 0 )
                     && ( ( buffer_iterator % 0x15b0 == 0 )
                       || ( buffer_iterator == buffer_size - 1 ) ) )
                    {
                              lower_word = lower_word % 0xfff1;
                              upper_word = upper_word % 0xfff1;
                    }
          }
          return( ( upper_word << 16 ) | lower_word );
}
```

Zlib provides the function adler32 which is an optimized version of the algorithm.

3.4. Segment file set identifier GUID/UUID

linen 5 to 6 use a time and MAC address based version (1) of the GUID.

EnCase 5 to 7 and linen 6 to 7 use a random based version (4) of the GUID.

In linen 6 the switch from a version 1 to 4 GUID/UUID was somewhere made between version 6.01 and 6.19.

See RFC4122 for more information

4. EWF-X

EWF-X (extended) is an experimental format to enhance the EWF format. EWF-X is based on the EWF-E01 format. EWF-X does not limit the table entries to 16375. EWF-X is not the same as version 2 of EWF.

4.1. Sections

Additional sections provided in the EWF-X format are:

- xheader
- xhash

4.1.1. Xheader

The xheader section contains a zlib compressed string containing XML data containing the header values.

```
<?xml version="1.0" encoding="UTF-8"?>
<xheader>
        <case_number>1</case_number>
        <description>Description</description>
        <examiner_name>John D.</examiner_name>
        <evidence_number>1.1</evidence_number>
        <notes>Just a floppy in my system</notes>
        <acquiry_operating_system>Linux</acquiry_operating_system>
        <acquiry_date>Sat Jan 20 18:32:08 2007 CET</acquiry_date>
        <acquiry_software>ewfacquire</acquiry_software>
        <acquiry_software_version>20070120</acquiry_software_version>
</xheader>
```

4.1.2. Xhash

The xhash section contains a zlib compressed string containing XML data containing the hash values.

```
<?xml version="1.0" encoding="UTF-8"?>
<xhash>
        <MD5>ae1ce8f5ac079d3ee93f97fe3792bda3</MD5>
        <SHA1>31a58f090460b92220d724b28eeb2838a1df6184</SHA1>
</xhash>
```

4.2. GUID

EWF-X uses a random based version of the GUID

5. Corruption scenarios

This chapter contains several corruption scenarios that have been encountered “in the wild”.

5.1. Corrupt uncompressed chunk

TODO

5.2. Corrupt compressed chunk

TODO

5.3. Corrupt section descriptor

TODO

```
libewf_section_start_read: reading section start from file IO pool entry: 1 at
offset: 415912423
libewf_section_start_read: type                       : table2
libewf_section_start_read: next offset                : 415978027
libewf_section_start_read: size                       : 65604
libewf_section_start_read: checksum                   : 0xf35f03e0
libewf_section_table_header_read: number of offsets   : 16375
libewf_section_table_header_read: base offset         : 0x00000000
libewf_section_table_header_read: checksum            : 0x180d0137

libewf_section_start_read: reading section start from file IO pool entry: 1 at
offset: 415978027
libewf_section_start_read: type                       : sectors
libewf_section_start_read: next offset                : 415978027
libewf_section_start_read: size                       : 0
libewf_section_start_read: checksum                   : 0x1ad00464
```

5.4. Corrupt table section

TODO, with and with out table 2
number of entries
entry data

5.5. Partial segment file

TODO

5.6. Missing segment file(s)

TODO

5.7. Dual image: section size versus offset

The sections descriptors define both the next section offset and the size of the section. If an implementation reads only one of the two to determine the next section, a dual EWF image can be crafted that consists of two separate images including hashes.

A current version of libewf will mark such an image as corrupted, but older versions only checked the section size and will show one of the two valid images.

5.8. Table entries offset overflow

In EnCase 6.7.1 the sectors section can be larger than 2048Mb. The table entries offsets are 31 bit values in EnCase6 the offset in a table entry value will actually **use the full 32 bit** if the 2048Mb has been exceeded. This behavior is no longer present in EnCase 6.8 so it is assumed to be a bug.

Libewf currently assumes that the if the 31 bit value overflows the following chunks are uncompressed. This allows EnCase 6.7.1 faulty EWF files to be converted by libewf.

5.9. Multiple incomplete segment file set identifiers

Although rare it can occur that a set of EWF image files changes its segment file set identifier. This was seen in an image created by EnCase 6.13, presumably using remote imaging. The image contained 3 different segment file set identifiers. The first changes after an incomplete section. The second one changed without any clear indication. The corresponding data section also changed in some extent e.g. compression method and media flags, the is physical flag being dropped. The change was consistent across multiple segment files. It is unlikely that deliberate manipulation is involved. EnCase considers the image as invalid.

Although with some tweaking of libewf the individual segment file sets could be read. In this case the data read from the segment file sets was heavily corrupted. For now a stock libewf does not support reading multiple segment files sets from a single image, but this might change in the future.

5.10. Notes

TODO
metadata (volume, data, headers)

6. AD encryption

As of version 2.8 FTK Imager supports “AD encryption”. Although the output file uses the EWF extensions the file actually is a AES-256 encrypted container. The EWF can be encrypted using a pass-phrase or a certificate.

7. Notes

What about:

- PALM volume

- the SMART logs

7.1. AD encryption

The container probably consists of a 512 byte header followed by the encrypted data.

offset	size	value	description
0	8		File signature “ADCRYPT\x00”
8	4	0x01	Version
12	4	0x0200	Data offset
16	8		Unknown (0x0000ffffffffffff)
24	4	0x03	Unknown
28	4	0x02	Unknown
32	4		Unknown
36	4	0x10	Unknown
40	4	0x20	Unknown
44	4	0x40	Unknown
58	112		Unknown key data
170	342		Unknown (empty data)

7.2. Header

All test headers consist of the where spaces are actually tabs separated values

```
srce
0       1
p       n       id      ev      tb      lo      po      ah      gu      aq
0       0
                                        -1      -1

sub
0       1
p       n       id      nu      co      gu
0       0
                                1
```

7.3. Ltree

When p = 0 n contains a numeric value (the number of child entries of the following entry)
When p = 1 n contains the name of the directory.
When p = is empty n contains the name of the file.

Appendix A. References

[ASR02]
Title: ASR Expert Witness Compression Format specification
Author(s) Andrew Rosen
URL: http://www.asrdata.com/SMART/whitepaper.html

[COH]
Title: PyFlag libevf source code
Author(s) Michael Cohen
URL: http://www.pyflag.net/

[FWIKI]
Title: Forensic Wiki
URL: http://www.forensicswiki.org/index.php/Forensic_file_formats
URL: http://www.forensicswiki.org/index.php/EnCase

[RFC1950]
Title: ZLIB Compressed Data Format Specification
Version: 3.3
Author(s): P. Deutsch, J-L. Gailly
Date: May 1996
URL: http://www.ietf.org/rfc/rfc1950.txt

[RFC1951]
Title: DEFLATE Compressed Data Format Specification
Version: 1.3
Author(s): P. Deutsch
Date: May 1996
URL: http://www.ietf.org/rfc/rfc1951.txt

[RFC4122]
Title: A Universally Unique Identifier (UUID) URN Namespace
Author(s): P. Leach, M. Mealling, R. Salz
Date: July 2005
URL: http://www.ietf.org/rfc/rfc4122.txt

Appendix B. GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful 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, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

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. (Thus, 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. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

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 Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document 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, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. 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, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include 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, PostScript or PDF 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.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

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 (or copies in media that commonly have printed covers) 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 computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. 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 fewer than five), unless they release you from this requirement.
- 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", Preserve 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. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, 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 to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

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, and that you preserve all their Warranty Disclaimers.

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, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which 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 half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket 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, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION
You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally

terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

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. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.