Extensible Storage Engine (ESE) Database File (EDB) format specification

Analysis of the Extensible Storage Engine (ESE) Database File (EDB) format

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

Summary

The Extensible Storage Engine (ESE) Database File (EDB) format is used by many Microsoft application to store data such as Windows Mail, Windows Search, Active Directory and Exchange. This specification is based on some available documentation but mainly on reverse engineering of the file format.

This document is intended as a working document for the Extensible Storage Engine (ESE) Database File (EDB) format 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 information about the Extensible Storage Engine Database File format

Classification: Public

Keywords: Extensible Storage Engine, ESE, ESENT, EDB

License

Version

Version	Author	Date	Comments
0.0.1	J.B. Metz	September 2009 October 2009	Worked on initial version.
0.0.2	J.B. Metz	October 5, 2009 October 6, 2009	Added information about page B+-trees.
0.0.3	J.B. Metz	October 8, 2009	Added information about tagged data types for EDB revision 2.
0.0.4	J.B. Metz	November 16, 2009 November 18, 2009	Additional information about indexes, page flags, MSysDefrag2 table.
0.0.5	J.B. Metz	February 22, 2010	Additional Windows 7 Search information.
0.0.6	J.B. Metz	May 14, 2010	Change amount of in number of Additional long value information.
0.0.7	J.B. Metz	May 17, 2010	Additional common page key information.
0.0.8	J.B. Metz	May 20, 2010 May 26, 2010	Additional information about template tables (thanks to J. Aloysius), root and branch pages.
0.0.9	J.B. Metz	June 2010	Additional multi value information.
0.0.10	J.B. Metz	July 2010	Additional index leaf page entry information.
0.0.11	J.B. Metz	September 2010	Windows 7 seems to use extended page format for 32 KiB pages, but not for 4 KiB pages. Currently assumed that 16 KiB pages also use the extended format.
0.0.12	J.B. Metz	November 2010	Additional information about streaming file.
0.0.13	J.B. Metz	December 2010	License version update
0.0.14	J.B. Metz	August 2011	Small addition to page value flags.
0.0.15	J.B. Metz	September 2011	Addition to page flags and 7-bit Unicode compression.
0.0.16	J.B. Metz	October 2011	Updates for space tree leaf page entry, 7-bit and XPRESS compression, scrubbed page flags.

Version	Author	Date	Comments
0.0.17	J.B. Metz	October 2011	Textual changes.
0.0.18	J.B. Metz	May 2012	Updates for Windows 8 Consumer Preview.
0.0.19	J.B. Metz	July 2012	Email update.

Table of Contents

1. Overview

The Extensible Storage Engine (ESE) Database File (EDB) format is used by many Microsoft application to store data such as Windows Mail, Windows Search, Active Directory and Exchange. The The Extensible Storage Engine (ESE) is also known as JET Blue.

There are multiple types of ESE:

Name	Usage
ESENT	The database engine for Active Directory and many Microsoft Windows components. Unlike other versions of ESE (which use 5-MiB log files and 4-KiB page sizes), the Active Directory implementation of ESENT uses 10-MiB log files and 8-KiB pages.
ESE97	The database engine in Exchange Server 5.5.
ESE98	The database engine in Exchange 2000 Server, Exchange Server 2003, and Exchange Server 2007. Exchange 2000 and 2003 use 4-KiB page sizes and 2007 8-KiB.

ESE is used to store data for various Microsoft applications like:

- Active Directory (NTDS)
- File Replication service (FRS)
- Windows Internet Name service (WINS)
- DHCP
- Security Configuration Engine (SCE)
- Certificate Server
- Terminal Services Session folder
- Terminal Services Licensing service
- Catalog database
- Help and Support Services
- Directory Synchronization service (MSDSS)
- Remote Storage (RSS)
- Phone Book service
- Single Instance Store (SIS) Groveler
- Windows NT Backup/Restore
- Exchange store
- Microsoft Exchange folder (SRS and DXA)
- Key Management service (KMS)
- Instant Messaging
- Content Indexing

1.1. Test version

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

- Exchange 2003, 2007; with corresponding eseutil
- Windows Search XP, Vista, 7 and 8; with corresponding esentutl

1.2. File structure

An ESE database (EDB) file consist of the following distinguishable elements:

- file header
- fixed size pages

Characteristics	Description
Byte order	little-endian
Date and time values	in both UTC and local time
Character string	ASCII strings are stored in extended ASCII with a codepage. Unicode strings are stored in UTF-16 little-endian without the byte order mark (BOM).

The pages contain the database, which basically consists of tables and indexes.

A table is made up out of:
- rows (also referred to as records)
- columns

An EDB contains several metadata tables, these are tables needed for maintaining the database. The metadata tables are:
- the space tree
- the catalog and the backup catalog

Because ESE stores the database data in fixed size pages, long values are used to store values that are larger than the page size.

2. (Database) file header

The (database) file header is stored in the first database page. The byte value in the remainder of the page are set to 0. A copy of the (database) file header is stored in the second page.

The (database) file header is (at least) 668 bytes of size and consists of:

offset	size	value	description
0	4		Checksum The checksum is a XOR over the 32-bit little-endian values in the header starting at offset 8 to offset 4096. The value 0x89abcdef is used as the initial value.
4	4	“\xef\xcd\xab\x89”	The signature
8	4		File format version
12	4		File type See section: 2.1 File type
16	8		Database time Consists of a database time See section: 5.1.1 Database time
24	28		Database signature Consists of a database signature See section: 5.1 Database signature
52	4		Database state See section: 2.3 Database state

offset	size	value	description
56	8		Consistent position Consists of a log position See section: 8.2 Log position This is the log position that was used when the database was last brought to a clean shutdown state or NULL if the database is in a dirty state.
64	8		Consistent date and time Consists of a log time See section: 8.3 log time This is the time when the database was last brought to a clean shutdown state or NULL if the database is in a dirty state.
72	8		Attach date and time Consists of a log time See section: 8.3 log time The date and time when the database was last attached.
80	8		Attach position Consists of a log position See section: 8.2 Log position The log position that was used the last time the database was attached.
88	8		Detach date and time Consists of a log time See section: 8.3 log time The date and time when the database was last detached.
96	8		Detach position Consists of a log position See section: 8.2 Log position The log position that was used the last time the database was detached.
104	28		Log signature Consists of a database signature See section: 5.1 Database signature
132	4	0	Unknown Empty value
136	24		Previous full backup Consists of a backup information See section: 7.1 Backup information
160	24		Previous incremental backup Consists of a backup information See section: 7.1 Backup information
184	24		Current full backup Consists of a backup information

offset	size	value	description
			See section: 7.1 Backup information
208	4		Shadowing disabled
212	4		Last object identifier The last object identifier in the database
216	4		Major version Represents the Windows NT major version when the databases indexes were updated.
220	4		Minor version Represents the Windows NT minor version when the databases indexes were updated.
224	4		Build number Represents the Windows NT build number when the databases indexes were updated.
228	4		Service pack number Represents the Windows NT service pack number when the databases indexes were updated.
232	4		File format revision
236	4		Page size Value in bytes
240	4		Repair count
244	8		Repair date and time Consists of a log time See section: 8.3 log time
252	28	0	Unknown2 See below
280	8		Scrub database time Consists of a database time See section: 5.1.1 Database time
288	8		Scrub date and time Consists of a log time See section: 8.3 log time
296	8		Required log Consists of 2x 32-bit values
304	4		Upgrade Exchange 5.5 format
308	4		Upgrade Free Pages
312	4		Upgrade Space Map Pages
316	24		Current shadow copy backup Consists of a backup information

offset	size	value	description
			See section: 7.1 Backup information
340	4		Creation file format version
344	4		Creation file format revision
348	16		Unknown3 See below
364	4		Old repair count
368	4		ECC fix success count
372	8		Last ECC fix success date and time Consists of a log time See section: 8.3 log time
380	4		Old ECC fix success count
384	4		ECC fix error count
388	8		Last ECC fix error date and time Consists of a log time See section: 8.3 log time
396	4		Old ECC fix error count
400	4		Bad checksum error count
404	8		Last bad checksum error date and time Consists of a log time See section: 8.3 log time
412	4		Old bad checksum error count
416	4		Committed log Consists of the lower 32-bit value
420	24		Previous (shadow) copy backup Consists of a backup information See section: 7.1 Backup information
444	24		Previous differential backup Consists of a backup information See section: 7.1 Backup information
468	40		Unknown Empty values
508	4		NLS major version Introduced in Windows 7 part of OS version
512	4		NLS minor version Introduced in Windows 7 part of OS version
516	148		Unknown Empty values
664	4		Unknown flags See below

```
unknown2:
00000000: a4 88 3d 00 14 07 0f 07  03 6a 00 00 00 00 00 00   ..=..... .j......
00000010: 00 00 00 00 00 00 00 00  00 00 00 00               ........ ....

found in stm
```

```
unknown3:
00000000: 2f 1d 07 0d 09 6b 00 00  00 00 00 00 00 00 00 00   /....k.. ........

found in tmp.edb
```

Unknown flags

Value	Identifier	Description
0x01000000		If not set the ECC and checksum counts and date and time values are not shown by eseutil, could be some extended data flag
0x02000000		Found in STM

```
Find location of:
fUpgradeDb value at offset 132?

   Streaming File: No (implied by file type)
             Dbid: 1

signSLV, fSLVExists

  Last checksum finish Date: 00/00/1900 00:00:00
Current checksum start Date: 00/00/1900 00:00:00
      Current checksum page: 0
```

Some of the values in the file header corresponds correspond with those in the miscellaneous database information (JET_DBINFOMISC).

In a clean database the consistent position, date and time matches the detach position, date and time.

2.1. File type

Value	Identifier	Description
0		Database Contains a hierarchical page-based storage
1		Streaming file Contains streamed data.

Note that the rest of the format specification largely applies to the database file type.

2.2. File format version and revision

According to [MSDN] the file format version and revision consist of the following values:

Version	Revision	Description
0x00000620	0x00000000	Original operating system Beta format (4/22/97).
0x00000620	0x00000001	Add columns in the catalog for conditional indexing and OLD (5/29/97).
0x00000620	0x00000002	Add the fLocalizedText flag in IDB (6/5/97).
0x00000620	0x00000003	Add SPLIT_BUFFER to space tree root pages (10/30/97).
0x00000620	0x00000002	Revert revision in order for ESE97 to remain forward-compatible (1/28/98).
0x00000620	0x00000003	Add new tagged columns to catalog ("CallbackData" and "CallbackDependencies").
0x00000620	0x00000004	Super Long Value (SLV) support: signSLV, fSLVExists in db header (5/5/98).
0x00000620	0x00000005	New SLV space tree (5/29/98).
0x00000620	0x00000006	SLV space map (10/12/98).
0x00000620	0x00000007	4-byte IDXSEG (12/10/98).
0x00000620	0x00000008	New template column format (1/25/99).
0x00000620	0x00000009	Sorted template columns (6/24/99). Used in Windows XP SP3
0x00000620	0x0000000b	Contains the page header with the ECC checksum Used in Exchange
0x00000620	0x0000000c	Used in Windows Vista (SP0)
0x00000620	0x00000011	Support for 2 KiB, 16 KiB and 32 KiB pages. Extended page header with additional ECC checksums. Column compression. Space hints. Used in Windows 7 (SP0)
0x00000623	0x00000000	New Space Manager (5/15/99).

2.3. Database state

The database state consist of the following values:

Value	Identifier	Description
1	JET_dbstateJustCreated	The database was just created.
2	JET_dbstateDirtyShutdown	The database requires hard or soft recovery to be run in order to become usable or movable. One should not try to move databases in this state.
3	JET_dbstateCleanShutdown	The database is in a clean state. The database can be attached without any log files.

Value	Identifier	Description
4	JET_dbstateBeingConverted	The database is being upgraded.
5	JET_dbstateForceDetach	Internal. This value is introduced in Windows XP

3. Hierarchical page-based storage

The EDB file uses a fixed size page to store data. The size of the page is defined in the file header.

In a database file these pages are ordered in a B+-tree. The pages can B+-tree references to other pages or data. These page B+-trees make up the database tables and indexes.

Every page B+-tree refers to a 'Father of the Data Page' (FDP) object identifier, which is basically a unique number for the specific page B+-tree.

A page consists of:
- a page header
- the page values
- the page tags (page value index)

The page (file) offset and number can be calculated as following:

```
page offset = ( page number x page size ) + page size
            = ( page number + 1 ) x page size
```

```
page number = ( page offset -  page size ) / page size
            = ( page offset / page size ) - 1
```

3.1. Page header

The page header is 40 or 80 bytes of size and consists of:

offset	size	value	description
Before Exchange 2003 SP1 and Windows Vista			
0	4		The XOR checksum The checksum is a XOR over the 32-bit little-endian values in the header starting at offset 4 to the end of the page. The value 0x89abcdef is used as the initial value.
4	4		Page number Used for the XOR checksum
Exchange 2003 SP1 and Windows Vista and later *(As of version 0x620 revision 0x0b)* *The new record format page flag must be set*			
0	4		The XOR checksum The checksum is a XOR over the 32-bit little-endian values in the header starting

offset	size	value	description
			at offset 8 to the end of the page. The page number is used as the initial value.
4	4		The ECC checksum [TODO]
Windows 7 and later *(As of version 0x620 revision 0x11)*			
0	8		Checksum [TODO]
Common			
8	8		Database last modification time Consists of a database time See section: 5.1.1 Database time This value indicates the database time the page was last modified.
16	4		Previous page number This value indicates the page number of the adjacent left page on the leaf.
20	4		Next page number This value indicates the page number of the adjacent right page on the leaf.
24	4		Father Data Page (FDP) object identifier This value indicates which page B+-tree this page belongs to.
28	2		Available data size The number of bytes available within the page.
30	2		Available uncommitted data size The number of uncommitted bytes within the page. Uncommitted bytes are free but available for reclaim by rollback on the page.
32	2		(First) available data offset The offset is relative from the end of the page header
34	2		(First) available page tag
36	4		Page flags See section: 3.1.3 Page flags
Extended page header Windows 7 and later *(As of version 0x620 revision 0x11)* *Only for pages of 16 KiB and 32 KiB ?*			
40	8		Extended checksum 1 [TODO]
48	8		Extended checksum 2

offset	size	value	description
			[TODO]
56	8		Extended checksum 3 [TODO]
64	8		Page number
72	8		Unknown Empty values

3.1.1. Changes in Exchange 2003 SP1

According to [MSDN] Exchange Server 2003 Service Pack 1 (SP1) introduces a new feature named Error Correcting Code (ECC) Checksum. ECC Checksum is a new checksum format that enables the correction of single-bit errors in database pages (in the .edb file, .stm file, and transaction log files). This new checksum format uses 64-bits, whereas the earlier checksum format uses 32-bits. Earlier format databases can be used with the new code, but current format databases cannot be used with earlier versions of ESE. After the database engine is updated, all pages that are written to the database have the new checksum format. Pages that are read and not modified do not have their checksum format upgraded.

Database pages with the earlier-format checksum start with a 32-bit checksum, followed by a 32-bit page number, which is used to verify that the requested page is actually read off disk.

The new checksum format removes the 32-bit page number and instead starts with an eight-byte checksum. The page number is used as an input parameter in calculating the checksum. Therefore, if the wrong page is read off disk, there will be a checksum mismatch.

The current checksum format actually consists of two 32-bit checksums. The first is an XOR checksum, calculated much like the earlier format checksum. The page number is used as a seed in the calculation of this checksum. The second 32-bit checksum is an ECC checksum, which allows for the correction of single-bit errors on the page.

3.1.2. Changes in Windows 7

In Windows 7, for pages of 16 KiB and 32 KiB, the page header was extended with mainly additional error recovery checksums.

3.1.3. Page flags

The page flags consist of the following values:

Value	Identifier	Description
0x00000001		The page is a root page
0x00000002		The page is a leaf page
0x00000004		The page is a parent page
0x00000008		The page is empty
0x00000010		

Value	Identifier	Description
0x00000020		The page is a space tree page
0x00000040		The page is an index page
0x00000080		The page is a long value page
0x00000100		
0x00000200		
0x00000400		Unknown
0x00000800		Unknown Does not seems to be the primary page flags? Flag for unique keys?
0x00001000		
0x00002000		New record format New checksum format
0x00004000		Is scrubbed (was zero-ed)
0x00008000		Unknown

Index page unique keys/non-unique keys
PageFlushType = 1 (0x8000) ?

3.2. Page tags

The page tags are stored at the end of the the page. The page tags are stored back to front. The page header indicates the first unused page tag.

Note that there can be more page tags in the page than being used.

3.2.1. Page tag - format revision 12 and earlier

A page tag is 4 bytes of size and consists of:

offset	size	value	description
0.0	13 bits		Value offset The offset is relative after the page header
1.5	3 bits		Page tag flags See section: 3.2.3 Page tag flags
2.0	13 bits		Value size
2.5	3 bits		Unknown Seen 2nd MSB set

3.2.2. Page tag - format revision 17 and later

In Windows 7 (format revision 0x11), for pages of 16 KiB and 32 KiB, the page tags were changed, to support these page sizes. For these page sizes the page tag flags have been moved to the first 16-value in the leaf page entry.

A page tag is 4 bytes of size and consists of:

offset	size	value	description
0.0	15 bits		Value offset The offset is relative after the extended page header
3.7	1 bit		Unknown Sometimes set
2	15 bits		Value size
3.6	1 bit		Unknown Sometimes set

3.2.3. Page tag flags

The page tag flags consist of the following values:

Value	Identifier	Description
0x0001	v	Unknown (Value) The page value contains variable sized data types?
0x0002	d	Defunct The page value is no longer used
0x0004	c	Common key The page value contains a common page key size

3.3. Page B+-tree

In the B+-tree hierarchy there are multiple types of pages:

- root page
- branch page
- leaf page

These different type of pages contain different types of page values.

3.3.1. Empty page

Although empty pages can contain data they are ignored when creating a page B+-tree.

3.3.2. Root page

The root page is identified by the 'is root' flag.

The root page contains different types of values:

- the root page header
- branch or leaf page entries

3.3.2.1. Root page header

The root page header is the first page tag within the page.

The root page header is 16 bytes of size and consists of:

offset	size	value	description
0	4		The initial number of pages The number of pages when the object was first created in the page tree.
4	4		The parent Father Data Page (FDP) number
8	4		Extent space 0x00000000 => single 0x00000001 => multiple
12	4		The space tree page number 0 if not set masks 0xff000000 if not set (pgnoOE)

The FDP flag in the eseutil seems to be implied if the parent Father Data Page (FDP) number (pgnoFDP) is set.

The primary extent represents the the initial number of pages followed by a dash and a letter after the that indicates whether the space for the B-Tree is currently represented using multiple pages ("m") or a single page ("s").

The space tree page number is valid when the extent space > 0.

3.3.3. Branch page

The branch page not identified by any flags, the 'is leaf' flag should not be set. The branch page can contain the 'is parent' flag.

What is the significance of the 'is parent' flag?

Both the branch page contains different types of values:

- the branch page header
- branch page entries

3.3.3.1. Branch page header

The branch page header is the first page tag within the page.

If the branch page has no 'is root' flag the branch page header is variable of size and consists of:

offset	size	value	description
0	...		Common page key

3.3.3.2. Branch page entry

The branch page entry is variable of size and consists of:

offset	size	value	description
If page tag flag 0x04 is set			
0	2		Common page key size
Common for all page flags			
0	2		Local page key size
2	(size)		The local page key The highest page key in the page B+-tree branch Note that the last father data page entry contains an empty page key
...	4		Child page number The child page number is invalid if it exceeds the last page in the file

The actual page key of the page entry is a combination of the part of the common page key, which is stored in the page header, specified by the size of the common page key size value, followed by the local page key stored in the page entry.

3.3.4. Leaf page values

The leaf page is identified by the 'is leaf' flag.

The leaf page contains different types of values:
- the leaf page header
- leaf page entries

There are multiple types of leaf pages:
- index leaf pages; identified by the 'is index' page flag
- long value leaf pages; identified by the 'is long value' page flag
- table leaf pages

Every type of leaf page has a different type of leaf page entry.

3.3.4.1. Leaf page header

The leaf page header is the first page tag within the page.

If the leaf page has no 'is root' flag the leaf page header is variable of size and consists of:

offset	size	value	description
0	...		Common page key

If there is no leaf page header the size of the corresponding page tag is 0.

3.3.4.2. Leaf page entry

The leaf page entries for the different types of leaf pages use a similar entry structure.

Note that the 3 MSB of the first 2 bytes can contain the page tag flags, see format revision 17.

The leaf page entry is variable of size and consists of:

offset	**size**	**value**	**description**
If page tag flag 0x04 is set			
0	2		Common page key size
Common for all page flags			
2	2		Local page key size
4	...		Local page key
...	...		Entry data

The actual page key of the page entry is a combination of the part of the common page key, which is stored in the page header, specified by the size of the common page key size value, followed by the local page key stored in the page entry.

3.3.4.2.1. Leaf page entry - format revision 17 and later

In Windows 7 (format revision 0x11), for pages of 16 KiB and 32 KiB, the size of the page key in the leaf page entry was changed.

The upper 3-bits of the first 16-bit value (either the key type or the size of the page key) contain the page tag flags (See section: 3.2.3 Page tag flags).

3.4. Page values

3.4.1. Space tree page values

The space tree page is identified by the following flags:

- is space tree

Is the root flag always set?

Space tree branch pages are similar to branch pages.

The space tree leaf page contains different types of values:

- the space tree page header
- space tree page entries

The primary space tree page referenced from the father data page contains information about the owned pages. The secondary space tree page which is the primary space tree page number + 1 contains information about the available pages.

3.4.1.1. Space tree leaf page header

The space tree page header is the first page value within the page.

The space tree page header is 16 bytes of size and consists of:

offset	size	value	description
0	16	0	Unknown

When the space tree page was referenced from the father data page the space tree page header contains 0 bytes.

The space tree header can also be empty (have a page value size of 0). related to root flag value?

TODO

```
00000000: 44 03 00 00 01 00 00 00  c6 03 00 00 04 00 00 00   D....... ........
```

3.4.1.2. Space tree leaf page entry

The space tree page entry is variable of size and consists of:

offset	size	value	description
0	2	4	Size of the page key
2	...		Page key value
...	4		number of pages

Owned space	The number of pages of all the space tree page entries in the primary space tree page make up the number of owned space.
Available space	The number of page of all the space tree page entries make up the number of available space.

Note that space tree entries with the defunct page flag (0x02) are not included.

3.4.2. Index page values

The index page is identified by the following flags:

- is index

Index branch pages are similar to branch pages.

3.4.2.1. Index leaf page entry data

The index leaf page entry data is variable of size and consists of:

offset	size	value	description
0	...		Record page key

3.4.3. Long value page values

The long value pages are identified by the following flags:

- is long value

For the format of the long value data definitions see section: 4.4 Long Values.

3.4.4. Table page values

The table page values are not identified by a flag. So basically if none of the previously mentioned flags is defined the page contains table value data definitions. See section: 4 Data definitions for more information.

4. Data definitions

In ESE there are multiple categories of table data definitions, each category uses different data type identifiers.

Data type identifiers	Amount	Category	Description
0x0001 – 0x007f	126	Fixed size	Fixed size data types (columns) use a defined number of space, even if no value is defined.
0x0080 - 0x00ff	127	Variable size	Variable size data types (columns) can contain up to 256 bytes of data. An offset array is stored in the record with the highest variable size data type set. Each array entry requires two bytes.
0x0100 - 0xfffff	64993	Tagged	Tagged data types (columns) are data types that occur rarely or have multiple occurrences. Tagged data types have an unlimited data size. The data type identifier and size are stored with the data. When a tagged data type does not contain data no information about it stored.

The data definitions are stored in (data definition) records. Such a data definition records contains the values of a table row.

According to [MSDN] data type identifiers 10 and 11 can be defined as variable columns

4.1. Data definition header

The data definition header is 4 bytes of size and consists of:

offset	size	value	Description
0	1		Last fixed size data type

offset	size	value	Description
1	1		Last variable size data types
2	2		The offset to the variable size data types The offset is relative from the start of the data definition header

4.2. Data type definitions

The data type definitions is variable of size and consists of:

offset	size	value	Description
0	...		Fixed size data type definitions
...	...		Unknown trailing data used to handle tagged data type definitions?
...	...		The variable size data types size array
...	...		The variable size data types data array Contains data for a variable data type
...	...		The tagged data type definitions

Although the corresponding table definition does not contain fixed size and/or variable size data type definitions the data type definition still can contain them. They need to be handled to find the offset of the tagged data type definitions.

The data type definitions will contain temple table tagged data type identifiers before table tagged data type identifiers. Also see section: 9.3 Template tables.

4.2.1. Variable size data type size array entry

The variable size data type size array entry is 2 bytes of size and consists of:

offset	size	value	Description
0	2		The variable size data type identifier Contains a 2 byte size value for every variable data type. The MSB signifies that the variable size data type is empty. Also the size of the previous variable size data type needs to be subtracted from the current size.

4.2.2. The tagged data type definitions - format revision 2

For EDB format revision 2 the tagged data type definitions consist of multiple entries.

A tagged data type definitions entry is variable of size and consists of:

offset	size	value	Description
0	2		The tagged data type identifier

offset	size	value	Description
2	2		Size of the tagged data type data The offset is relative from the start of the tagged data type offset array flag bits: 0x8000 (?)
4	1		Tagged data type flags Currently only 0x00 values have been seen
5	...		Value

When the 0x8000 flag bit is set the tagged data type offset array entry is directly followed by the value data. The size of the tagged data type data contains the size of the value data. The value is seems to be preceded by the tagged data type flags?

4.2.3. The tagged data type definitions - format revision 9 and later

For format revision 9 and later the tagged data type definitions consist of an an offset and data array.

offset	size	value	Description
0	...		The tagged data types offset array
...	...		The tagged data types data array

4.2.3.1. Tagged data type offset array entry - format revision 9 and later

The tagged data type offset array entry is 4 bytes of size and consists of:

offset	size	value	Description
0	2		The tagged data type identifier
2	2		Size or offset of the tagged data type data The offset is relative from the start of the tagged data type offset array flag bits: 0x4000 (tagged data type flags present) 0x8000 (?)

What does a size of 0 indicate: that the value is empty or contains the default value?

The number of tagged data types is deduced from the first tagged data type data offset?

If the bit 0x4000 is set in the size the value is preceded by the tagged data type flags. The size cannot be greater equal than 0x4000.

Note that as of Windows 7 and later (version 0x620 revision 0x11), for pages of 16 KiB and 32 KiB, the tagged data type flags are always present in database and no longer controlled by the flag bits. The size can be greater equal than 0x4000.

4.2.3.2. Tagged data type flags

Value	Identifier	Description
0x01		Variable size value
0x02		Data is compressed
0x04		Data is stored in a long value The data type definition contains a long value identifier, which is the key of the long value in reverse
0x08		Data contains a multi value See section: 4.5 Mutli values
0x10		Multi value contains size definition instead of offset definitions

Are multi long values used?

Tag data type flags:

```
0x01 => unicode value or single value (not the sparse flag)
0x05 => Long value (4 byte long value identifier or page key)
0x08 => (fixed size type?) multi value
0x09 => (variable size type?) multi value
0x0b => compressed multi value (see below)
0x18 => (fixed size type?) multi value (with size definition)
```

```
column definition name                                  : System_Kind
column definition type                                  : Text (extended
ASCII or Unicode string) (JET_coltypText)
(450) tagged data type identifier                       : 450
(450) tagged data type offset                           : 0x4244 (580)
(450) tagged data type size                             : 24
(450) tag byte                                          : 0x18
(450) tagged data type:
00000000: 08 6c 00 69 00 6e 00 6b  00 70 00 72 00 6f 00 67   .l.i.n.k .p.r.o.g
00000010: 00 72 00 61 00 6d 00                               .r.a.m.

byte size of first value?
```

```
(457) tagged data type flags            : 0x0b
        Is variable size
        Is compressed
        Is multi value

(457) tagged data type:
00000000: 04 00 09 00 13 ec b4 7b  0d 70 00 72 00 6f 00 67   .......{ .p.r.o.g
00000010: 00 72 00 61 00 6d 00                               .r.a.m.

Why is only the first entry is compressed?
```

4.3. Example: the catalog (data type) definition

The data below is an example of the catalog (data type) definition. Also see section: 9.2.1 Catalog (MSysObjects and MSysObjectsShadow)

offset	size	value	Description
Fixed size data type definitions			
0	4		The Father Data Page (FDP) object identifier
4	2		Catalog type See section: 9.2.1.1 Catalog types
6	4		The identifier
If data definition type is 0x0002 (column)			
10	4		Column type See section: 6.1 Column type
Other data definition types			
10	4		The Father Data Page (FDP) number
If data definition type is 0x0001 (table)			
14	4		Space usage The number of pages used by the table
18	4		Flags (or group of bits)
22	4		The (initial) number of pages
If data definition type is 0x0002 (column)			
14	4		Space usage The number of bytes used by the column
18	4		Flags (or group of bits) See section: 6.2 Column flags (group of bits)
22	4		Codepage
If data definition type is 0x0003 (index)			
14	4		Space usage The number of pages used by the index
18	4		Flags (or group of bits)
22	4		The locale identifier (LCID) See section: [NTLCID] The LCID is used for normalizing the string when JET_bitIndexUnicode is not specified in the index flags (group of bits).
If data definition type is 0x0004 (long value)			
14	4		Space usage The number of pages used by the long value

offset	size	value	Description
18	4		Flags (or group of bits) 0x00000000 => single extent 0x00000001 => multiple extent
22	4		The (initial) number of pages
If data definition type is 0x0005 (callback)			
			TODO
All data definition types			
26	1		The root flag
27	2		The record offset The offset of the data type within the record
29	4		The LC map flags
33	2		Key most
35	...		Unknown trailing data used to handle tagged data type definitions?
...	...		The variable data types size array
...	...		The variable data types data array Contains data for a variable data type
If more data is present			
...	...		The tagged data types offset array
If present in the tagged types offset array			
			The tagged data types data array Contains data for a tagged data type

For data definition type is 0x0001 (table) the variable data type 'TemplateTable' is used to store the name of the table used as its template. See section: 9.3 Template tables.

For data definition type is 0x0005 (callback) the variable data type 'TemplateTable' is used to store the name of the DLL and function to call.

4.4. Long Values

The actual long values are stored in a separate page tree. The corresponding page key of the long value is the long value identifier in reverse byte order. E.g. a long value identifier of: 0xa7000000 relates to a page key of 0x000000a7. In version 0x620 and revision 0x0c the page key contains the leading 0 values in revision 0x09 these leading 0 values are not present.

The long value page key refers to a page value in the long value page tree corresponding to the table page tree as defined in the catalog.

This page value contains the long value header. The long value header is 8 bytes of size and consists of:

offset	size	value	Description
0	4		Unknown Value is 1 Value is 0 in some defunct long values
4	4		Unknown Last segment offset Hypothesis: the total long value size, holds for a lot of single segment long values but not for some multi segment long values Largest segment size?l

The corresponding segments can be found by combining the long value page key with a 4 byte segment offset, starting with offset 0. E.g. the first segment for the long value identifier 0xa7000000 is the page key 0x000000a7 followed by the segment offset 0x00000fae (4014), therefore 0x000000a7000000fae.

One long value page tree per table?

Inverse key stored in data type definition

The offset (+ data size) of the last segment can exceed the total long value size?

4.5. Mutli values

The multi value is variable of size and consists of:

offset	size	value	Description
0	...		Value offset array Consists of 16-bit offset values The offset is relative to the start of the multi value flag bits: 0x8000 (?)
...	...		Value data array

```
column definition identifier                                    : 625
column definition name                                          : ML827a
column definition type                                          : Integer 32-bit
signed (JET_coltypLong)
(625) tagged data type identifier                               : 625
(625) tagged data type offset                                   : 0x43cb (971)
(625) tagged data type size                                     : 31
(625) tag byte                                                  : 0x08
(625) tagged data type:
00000000: 0a 00 0e 00 12 00 16 00  1a 00 17 80 00 00 37 80   ........ ......7.
00000010: 00 00 16 3a 00 00 19 80  00 00 18 80 00 00         ...:.... ......

00000000: 06 00 0a 00 0e 00 80 80  00 00 90 80 00 00 a0 80   ........ ........
00000010: 00 00                                              ..
```

```
2 byte offset(s)
fixed size value(s)
```

```
column definition identifier                                    : 318
column definition name                                          : MN667f
column definition type                                          : Large binary data
(JET_coltypLongBinary)
(318) tagged data type identifier                               : 318
(318) tagged data type offset                                   : 0x4173 (371)
(318) tagged data type size                                     : 45
(318) tag byte                                                  : 0x09
(318) tagged data type:
00000000: 04 00 18 00 44 0d 4a ae  39 18 8f 40 a0 0d be 80   ....D.J. 9..@....
00000010: cb bf cd ad 00 00 00 00  5a 1f 4f 36 67 80 6b 4f   ........ Z.O6g.kO
00000020: a1 81 89 f2 bb 7e 6b 39  00 00 00 00              .....~k9 ....

2 byte offset(s)
variable size value(s)
```

```
column definition identifier            : 296
column definition name                  : MS8053
column definition type                  : Large text (extended ASCII or
Unicode string) (JET_coltypLongText)
(296) tagged data type identifier       : 296
(296) tagged data type offset           : 0x429b (667)
(296) tagged data type size             : 3019
(296) tagged data type flags            : 0x09
        Is variable size
        Is multi value

(296) tagged data type:
00000000: 42 00 9e 00 f8 00 58 01  bc 01 1c 02 7a 02 d8 02   B.....X. ....z...
00000010: 40 03 a8 03 0c 04 72 04  d4 04 2e 05 98 05 f6 05   @.....r. ........
00000020: 64 06 d6 06 30 07 8a 07  ee 07 52 08 c6 08 26 09   d...0... ..R...&.
00000030: 88 09 e8 09 44 0a a2 0a  02 0b 64 0b be 8b c2 8b   ....D... ..d.....
00000040: c6 8b 75 00 72 00 6e 00  3a 00 73 00 63 00 68 00   ..u.r.n. :.s.c.h.

MSB contains some flag (defunct?)
```

```
0x8000 flag

00000000: 42 00 9e 00 f8 00 58 01  bc 01 1c 02 7a 02 d8 02   B.....X. ....z...
00000010: 40 03 a8 03 0c 04 72 04  d4 04 2e 05 98 05 f6 05   @.....r. ........
00000020: 64 06 d6 06 30 07 8a 07  ee 07 52 08 c6 08 26 09   d...0... ..R...&.
00000030: 88 09 e8 09 44 0a a2 0a  02 0b 64 0b be 8b c2 8b   ....D... ..d.....
00000040: c6 8b                                             ..

00000040:       75 00 72 00 6e 00  3a 00 73 00 63 00 68 00     u.r.n. :.s.c.h.
00000050: 65 00 6d 00 61 00 73 00  2d 00 6d 00 69 00 63 00   e.m.a.s. -.m.i.c.
00000060: 72 00 6f 00 73 00 6f 00  66 00 74 00 2d 00 63 00   r.o.s.o. f.t.-.c.
00000070: 6f 00 6d 00 3a 00 6f 00  66 00 66 00 69 00 63 00   o.m.:.o. f.f.i.c.
00000080: 65 00 3a 00 6f 00 66 00  66 00 69 00 63 00 65 00   e.:.o.f. f.i.c.e.
00000090: 23 00 41 00 75 00 74 00  68 00 6f 00 72 00         #.A.u.t. h.o.r.

00000090:                                             75 00                 u.
000000a0: 72 00 6e 00 3a 00 73 00  63 00 68 00 65 00 6d 00   r.n.:.s. c.h.e.m.
```

```
00000bb0: 65 00 23 00 54 00 69 00  74 00 6c 00 65 00 43 00   e.#.T.i. t.l.e.C.
00000bc0: 00 00 44 00 00 00 45 00  00 00                    ..D...E. ..
```

5. Database

5.1. Database signature

The database signature (JET_SIGNATURE) is 28 bytes of size and consists of:

offset	size	value	description
0	4		A randomly assigned number
4	8		Creation date and time Consists of a log time See section: 8.3 log time
12	16		The NetBIOS computer name ASCII string terminated by a NUL-character Unused bytes are filled with 0

5.1.1. Database time

The database time (DBTIME) is 8 bytes of size and consists of:

offset	size	value	description
0	2		Hours Value should be [0 - 23]
2	2		Minutes Value should be [0 - 59]
4	2		Seconds Value should be [0 – 59]
6	2	0	Padding

6. Columns

6.1. Column type

The column type (JET_COLTYP) consist of the following values:

Value	Identifier	Description
0	JET_coltypNil	Invalid Invalid column type.
1	JET_coltypBit	Boolean Boolean column type that can be true, or false but cannot be NULL. This type of column is one byte of size and is a fixed size.

Value	Identifier	Description
2	JET_coltypUnsignedByte	Integer 8-bit unsigned
3	JET_coltypShort	Integer 16-bit signed
4	JET_coltypLong	Integer 32-bit signed
5	JET_coltypCurrency	Currency (64-bit) An 8-byte signed integer that can consist of values between - 9223372036854775808 and 9223372036854775807.
6	JET_coltypIEEESingle	Floating point single precision (32-bit)
7	JET_coltypIEEEDouble	Floating point double precision (64-bit)
8	JET_coltypDateTime	Date and time (64-bit) The date and time is stored as a little-endian filetime A double-precision (8-byte) floating point number that represents a date in fractional days since the year 1900. This column type is identical to the variant date type (VT_DATE).
9	JET_coltypBinary	Binary data A fixed or variable size, raw binary column that can be up to 255 bytes in size.
10	JET_coltypText	Text (Extended ASCII or Unicode) A fixed or variable size text column that can be up to 255 ASCII characters in size or 127 Unicode characters in size. The text need not be null terminated, but embedded null characters can be stored.
11	JET_coltypLongBinary	Large binary data A fixed or variable size, raw binary column that can be up to 2147483647 bytes of size.
12	JET_coltypLongText	Large text (Extended ASCII or Unicode) A fixed or variable size, text column that can be up to 2147483647 ASCII characters in size or 1073741823 Unicode characters in size.
Values introduced in Windows XP		
13	JET_coltypSLV	Super Large Value This column type is obsolete. A record in the .edb file contains a column (of data type JET_coltypSLV) that references a list of pages in the streaming file that contains the raw data. Space usage (maximum of four kilobytes of page numbers) and checksum data for the data in the streaming file is stored in the .edb file. SLV = Super Long Value
Values introduced in Windows Vista		
14	JET_coltypUnsignedLong	Integer 32-bit unsigned
15	JET_coltypLongLong	Integer 64-bit signed

Value	Identifier	Description
16	JET_coltypGUID	GUID (128-bit)
17	JET_coltypUnsignedShort	Integer 16-bit unsigned

ASCII strings are always treated as case insensitive for sorting and searching purposes. Further, only the characters preceding the first null character (if any) are considered for sorting and searching. Unicode strings use the Win32 API LCMapString to create sort keys that are subsequently used for sorting and searching that data. By default, Unicode strings are considered to be in the U.S. English locale and are sorted and searched using the following normalization flags: NORM_IGNORECASE, NORM_IGNOREKANATYPE, and NORM_IGNOREWIDTH. In Windows 2000, it is possible to customize these flags per index to also include NORM_IGNORENONSPACE. In Windows XP and later releases, it is possible to request any combination of the following normalization flags per index: LCMAP_SORTKEY, LCMAP_BYTEREV, NORM_IGNORECASE, NORM_IGNORENONSPACE, NORM_IGNORESYMBOLS, NORM_IGNOREKANATYPE, NORM_IGNOREWIDTH, and SORT_STRINGSORT.
In all releases, it is possible to customize the locale per index. Any locale may be used as long as the appropriate language pack has been installed on the machine. Finally, any null characters encountered in a Unicode string are completely ignored.

6.2. Column flags (group of bits)

The column flags consist of the following values:

Value	Identifier	Description
0x00000001	JET_bitColumnFixed	Is fixed size The column will always use the same size (within the row) regardless of how much data is stored in the column.
0x00000002	JET_bitColumnTagged	Is tagged The column is tagged. A tagged columns does not take up any space in the database if it does not contain data.
0x00000004	JET_bitColumnNotNULL	Not empty The column is not allow to be set to an empty value (NULL).
0x00000008	JET_bitColumnVersion	Is version column The column is a version column that specifies the version of the row.
0x00000010	JET_bitColumnAutoincrement	The column will automatically be incremented. The number is an increasing number, and is guaranteed to be unique within a table. The numbers, however, might not be continuous. For example, if five rows are inserted into a table, the "autoincrement" column could contain the values { 1, 2, 6, 7, 8 }. This bit can only be used on columns of type JET_coltypLong or JET_coltypCurrency.
0x00000020	JET_bitColumnUpdatable	This bit is valid only on calls to JetGetColumnInfo.

Value	Identifier	Description
0x00000040	JET_bitColumnTTKey	This bit is valid only on calls to JetOpenTable.
0x00000080	JET_bitColumnTTDescending	This bit is valid only on calls to JetOpenTempTable.
0x00000400	JET_bitColumnMultiValued	The column can be multi-valued. A multi-valued column can have zero, one, or more values associated with it. The various values in a multi-valued column are identified by a number called the itagSequence member, which belongs to various structures, including: JET_RETINFO, JET_SETINFO, JET_SETCOLUMN, JET_RETRIEVECOLUMN, and JET_ENUMCOLUMNVALUE. Multi-valued columns must be tagged columns; that is, they cannot be fixed-length or variable-length columns.
0x00000800	JET_bitColumnEscrowUpdate	Specifies that a column is an escrow update column. An escrow update column can be updated concurrently by different sessions with JetEscrowUpdate and will maintain transactional consistency. An escrow update column must also meet the following conditions: • An escrow update column can be created only when the table is empty. • An escrow update column must be of type JET_coltypLong. • An escrow update column must have a default value (that is cbDefault must be positive). • JET_bitColumnEscrowUpdate cannot be used in conjunction with JET_bitColumnTagged, JET_bitColumnVersion, or JET_bitColumnAutoincrement.
0x00001000	JET_bitColumnUnversioned	The column will be created in an without version information. This means that other transactions that attempt to add a column with the same name will fail. This bit is only useful with JetAddColumn. It cannot be used within a transaction.
Values introduced in Windows 2003		
0x00002000	JET_bitColumnDeleteOnZero	The column is an escrow update column, and when it reaches zero, the record will be deleted. A common use for a column that can be finalized is to use it as a reference count field, and when the field reaches zero the record gets deleted. JET_bitColumnDeleteOnZero is related to JET_bitColumnFinalize. A Delete-on-zero column must be an escrow update column.

Value	Identifier	Description
		JET_bitColumnDeleteOnZero cannot be used with JET_bitColumnFinalize. JET_bitColumnDeleteOnZero cannot be used with user defined default columns.
Values introduced in Windows XP		
0x00002000	JET_bitColumnMaybeNull	Reserved for future use.
0x00004000	JET_bitColumnFinalize	Use JET_bitColumnDeleteOnZero instead of JET_bitColumnFinalize. JET_bitColumnFinalize that a column can be finalized. When a column that can be finalized has an escrow update column that reaches zero, the row will be deleted. Future versions might invoke a callback function instead (For more information, see JET_CALLBACK). A column that can be finalized must be an escrow update column. JET_bitColumnFinalize cannot be used with JET_bitColumnUserDefinedDefault.
0x00008000	JET_bitColumnUserDefinedDefault	The default value for a column will be provided by a callback function. See JET_CALLBACK. A column that has a user-defined default must be a tagged column. Specifying JET_bitColumnUserDefinedDefault means that pvDefault must point to a JET_USERDEFINEDDEFAULT structure, and cbDefault must be set to sizeof(JET_USERDEFINEDDEFAULT). JET_bitColumnUserDefinedDefault cannot be used in conjunction with JET_bitColumnFixed, JET_bitColumnNotNULL, JET_bitColumnVersion, JET_bitColumnAutoincrement, JET_bitColumnUpdatable, JET_bitColumnEscrowUpdate, JET_bitColumnFinalize, JET_bitColumnDeleteOnZero, or JET_bitColumnMaybeNull.

6.3. Compression

As of Windows 7 the column types JET_coltypLongBinary and JET_coltypLongText can be compressed [MSDN-WIN7].

The first byte in the data indicates which compression is used. If the value is 0x18 the data is XPRESS compressed. The data is 7-bit compressed for any other value.

6.3.1. 7-bit compression

7-bit compression is used for columns with less than 1 KiB (1024 bytes) uncompressed data that consists of only 7-bit values. These are stored as a continuous stream of 7-bit values.

To decompress:

1. check if the leading byte does not contain 0x18.
    1. If the column type is the JET_coltypLongText
        1. If the lead byte contains 0x10 and the data is ASCII text
        2. Otherwise the data is either ASCII or UTF16 little-endian
2. start reading at offset 1
3. while not at end of stream
    1. read a 7-bit value from the stream and convert it into an 8-bit value

If the column type is JET_coltypLongText the uncompressed data either contains an ASCII or an UTF-16 little-endian string.

Notes: Contains unicode 0x09, 0x0b, 0x0d, 0x0f on Win7 but not in Exchange 2010

6.3.2. XPRESS compression

Microsoft XPRESS compression is used for columns with more than 1 KiB (1024 bytes) uncompressed data. This compression method is a combination of the LZ77 and DIRECT2 algorithms. The compression method is similar to the LZNT1, which is used in NTFS compression.

The compressed data is variable in size and consists of:

offset	size	value	Description
0	1	0x18	Leading byte
1	3		Uncompressed data size
3	...		XPRESS compressed data

If the column type is JET_coltypLongText the uncompressed data either contains an ASCII or an UTF-16 little-endian string.

TODO: what about data > 2^16

7. Backup

7.1. Backup information

The backup information (JET_BKINFO) is 24 bytes of size and consists of:

offset	size	value	description
0	8		The backup position Consists of a log position See section: 8.2 Log position Contains an identifier of the backup
8	8		The backup creation date and time Consists of a backup log time See section: 8.3 log time
16	4		Generation lower number The lower log generation number associated with the backup.

offset	size	value	description
20	4		Generation upper number The upper log generation number associated with the backup.

8. Transaction log

8.1. Log information

The log position (JET_LOGINFO) is 16 bytes of size and consists of:

offset	size	value	description
0	4	16	Size of the structure
4	4		Generation lower number The lower log generation number associated with the transaction.
8	4		Generation upper number The upper log generation number associated with the transaction.
12	4		Log filename prefix The prefix used to name the transaction log files.

Transaction log files are named according to the instance base name and the generation number of the log file. The name is of the format BBBXXXXX.LOG. BBB corresponds to the base name for the log file and is always three characters in length. XXXXX corresponds to the generation number of the log file in zero padded hexadecimal and is always five characters in length. LOG is the file extension that is always given to transaction log files by the engine.

8.2. Log position

The log position (JET_LGPOS) is 8 bytes of size and consists of:

offset	size	value	description
0	2		block
2	2		sector
4	4		generation

8.3. (Backup) log time

The backup log time and log time (JET_BKLOGTIME and JET_LOGTIME) is 8 bytes of size and consist of:

offset	size	value	description
0	1		Seconds Value should be [0 - 60]
1	1		Minutes

offset	size	value	description
			Value should be [0 - 60]
2	1		Hours Value should be [0 - 24]
3	1		Days Value should be [0 - 31]
4	1		Months Value should be [0 - 12]
5	1		Years The year 0 represents 1900.
6	1	0	Filler byte
7	1	0	Filler byte

In a backup log time the LSB of the second filler byte can be overloaded to contains the backup type bit. The backup type bit consists of one of the following values:

Value	Identifier	Description
0		streaming backup
1		snapshot backup

The backup log time was introduced in Windows Vista.

9. Tables

9.1. Table flags (group of bits)

The table group of bits consist of the following values:

Value	Identifier	Description
0x00000001	JET_bitTableCreateFixedDDL	Setting JET_bitTableCreateFixedDDL prevents DDL operations on the table (such as adding or removing columns).
0x00000002	JET_bitTableCreateTemplateTable	Setting JET_bitTableCreateTemplateTable causes the table to be a template table. New tables can then specify the name of this table as their template table. Setting JET_bitTableCreateTemplateTable implies JET_bitTableCreateFixedDDL.
Values introduced in Windows XP		
0x00000004	JET_bitTableCreateNoFixedVarColumnsInDerivedTables	Deprecated. Do not use.

9.2. metadata tables

9.2.1. Catalog (MSysObjects and MSysObjectsShadow)

The “MSysObjects” table contains the definitions of all the tables, indexes and long values that are stored within the database. It is also referred to a the catalog (metadata table). A backup (or copy) of the catalog is maintained in the “MSysObjectsShadow” table.

The page values (in the leaf pages) that make up the catalog contain the following information for every table in the database:

- a table definition
- one or more column definition
- one or more index definitions; there is always at least one index for a table
- zero or more long value definitions

The catalog also contains its own table definition. The catalog table definition consist of:

Column identifier	**Column name**	**Column type**	**Description**
Fixed size data definition types			
1	ObjidTable	Long	Object or table identifier
2	Type	Short	Type See section: 9.2.1.1 Catalog types
3	Id	Long	Identifier
4	ColtypOrPgnoFDP	Long	Column type or FDP page number
5	SpaceUsage	Long	Space usage
6	Flags	Long	Flags
7	PagesOrLocale	Long	Number of pages or codepage
8	RootFlag	Bit	Root flag
9	RecordOffset	Short	Record offset
10	LCMapFlags	Long	Flags for the LCMapString function.
Introduced in Windows Vista (version 0x620 revision 0x0c)			
11	KeyMost	Short	
Variable size data definition types			
128	Name	Text	Name
129	Stats	Binary	
130	TemplateTable	Text	Name of the template 'table'
131	DefaultValue	Binary	Default value
132	KeyFldIDs	Binary	For the index column identifiers
133	VarSegMac	Binary	
134	ConditionalColumns	Binary	
135	TupleLimits	Binary	

Column identifier	Column name	Column type	Description
Introduced in Windows Vista (version 0x620 revision 0x0c)			
136	Version	Binary	
Tagged data definition types			
256	CallbackData	Large binary data	Data used in callback
257	CallbackDependencies	Large binary data	Dependencies for callback
Introduced in Windows 7 (version 0x620 revision 0x11)			
258	SeparateLV	Large binary data	
259	SpaceHints	Large binary data	
260	SpaceDeferredLVHints	Large binary data	

A codepage of 1200 can represent either UTF-8 (or even byte stream?) or UTF-16 little-endian. The way to tell is that the size of the UTF-16 stream should be a multitude of 2. If so try to decode the string as UTF-16 first.

9.2.1.1. Catalog types

Value	Identifier	Description
0x0001		Table
0x0002		Column
0x0003		Index
0x0004		Long value
0x0005		Callback
0x0006		Related to SLVAvail (part of object 1)
0x0007		Related to SLVSpaceMap (part of object 1)

9.2.1.2. KeyFldIDs

The KeyFldIDs contain the index column identifiers of the primary and secondary keys.

A index column identifier entry is 4 bytes of size and consists of:

offset	size	value	Description
0	2		Unknown
2	2		Index column identifier

offset	size	value	Description
			Contains the data type identifier of the column

Id

```
00000000: 00 00 01 00 00 00 02 00  00 00 03 00

Id column identifier (3)
```

Name

```
00000000: 00 00 01 00 00 00 02 00  00 00 80 00

Name column identifier (128)
```

RootObjects

```
00000000: 00 00 08 00 00 00 80 00
```

9.2.2. MSysObjids

Column identifier	Column name	Column type
256	objid	Integer 32-bit signed
257	objidTable	Integer 32-bit signed
258	type	Integer 16-bit signed

First seen in Windows 8 Consumer Preview Windows.edb

9.2.3. MSysLocales

Column identifier	Column name	Column type
1	Type	Integer 8-bit unsigned
2	iValue	Integer 32-bit signed
128	Key	Binary data

First seen in Windows 8 Consumer Preview Windows.edb

9.2.4. MSysUnicodeFixupVer1

Column identifier	Column name	Column type
1	autoinc	Currency
256	objidTable	Long
257	objidIndex	Long
258	keyPrimary	Long

Column identifier	Column name	Column type
259	keySecondary	Long
260	lcid	Long
261	sortVersion	Long
262	definedVersion	Long
263	itag	Long
264	ichOffset	Long

9.2.5. MSysUnicodeFixupVer2

The “MsysUnicodeFixupVer2” table was introduced in Windows Vista (SP0)?

Column identifier	Column name	Column type
1	autoinc	Currency
256	objidTable	Long
257	objidIndex	Long
258	keyPrimary	Long
259	keySecondary	Long
260	lcid	Long
261	sortVersion	Long
262	definedVersion	Long
263	rgitag	Long
264	ichOffset	Long

9.2.6. MSysDefrag1

Column identifier	Column name	Column type
1	ObjidFDP	Integer 32-bit signed
2	DefragType	Integer 8-bit unsigned
3	Sentinel	Integer 32-bit signed
4	Status	Integer 16-bit signed
256	CurrentKey	Large binary data

9.2.7. MSysDefrag2

Column identifier	Column name	Column type
1	ObjidFDP	Integer 32-bit signed
2	Status	Integer 16-bit signed
3	PassStartDateTime	Integer 64-bit signed

Column identifier	Column name	Column type
4	PassElapsedSeconds	Integer 64-bit signed
5	PassInvocations	Integer 64-bit signed
6	PassPagesVisited	Integer 64-bit signed
7	PassPagesFreed	Integer 64-bit signed
8	PassPartialMerges	Integer 64-bit signed
9	TotalPasses	Integer 64-bit signed
10	TotalElapsedSeconds	Integer 64-bit signed
11	TotalInvocations	Integer 64-bit signed
12	TotalDefragDays	Integer 64-bit signed
13	TotalPagesVisited	Integer 64-bit signed
14	TotalPagesFreed	Integer 64-bit signed
15	TotalPartialMerges	Integer 64-bit signed
256	CurrentKey	Large binary data

9.3. Template tables

A table definition which uses a template table definition, basically uses a copy of the template table and appends the defined column definitions.

E.g. if the template table defines 446 columns and the definition of the last column is a tagged data type:

Column identifier	Column name	Column type
669	Q65a0	Binary data

The first column definition in the table will be column number 447:

Column identifier	Column name	Column type
256	N67b9	Large binary data

Note that table column identifier is 256 and will also be defined as such in the tagged data type definitions.

What about non tagged data types?

10. Indexes

The FDP value in the catalog definition of an index, refers to the FDP of an index page B+-tree except for the first index (Id). It will point to the parent table and does not contain index page values. It is assumed that this index is build-in.

10.1. Index flags (group of bits)

The column flags consist of the following values:

Value	Identifier	Description
0x00000001	JET_bitIndexUnique	Duplicate index entries (keys) are disallowed. This is enforced when JetUpdate is called, not when JetSetColumn is called.
0x00000002	JET_bitIndexPrimary	The index is a primary (clustered) index. Every table must have exactly one primary index. If no primary index is explicitly defined over a table, then the database engine will create its own primary index.
0x00000004	JET_bitIndexDisallowNull	None of the columns over which the index is created may contain a NULL value.
0x00000008	JET_bitIndexIgnoreNull	Do not add an index entry for a row if all of the columns being indexed are NULL.
0x00000010		Unknown Set if the index contains 3 column identifiers?
0x00000020	JET_bitIndexIgnoreAnyNull	Do not add an index entry for a row if any of the columns being indexed are NULL.
0x00000040	JET_bitIndexIgnoreFirstNull	Do not add an index entry for a row if the first column being indexed is NULL.
0x00000080	JET_bitIndexLazyFlush	Specifies that the index operations will be logged lazily. JET_bitIndexLazyFlush does not affect the laziness of data updates. If the indexing operations is interrupted by process termination, Soft Recovery will still be able to able to get the database to a consistent state, but the index may not be present.
0x00000100	JET_bitIndexEmpty	Do not attempt to build the index, because all entries would evaluate to NULL. grbit MUST also specify JET_bitIgnoreAnyNull when JET_bitIndexEmpty is passed. This is a performance enhancement. For example if a new column is added to a table, then an index is created over this newly added column, all of the records in the table would be scanned even though they would never get added to the index anyway. Specifying JET_bitIndexEmpty skips the scanning of the table, which could potentially take a long time.
0x00000200	JET_bitIndexUnversioned	JET_bitIndexUnversioned causes index creation to be visible to other transactions. Normally a session in a transaction will not be able to see an index creation operation in another session. This flag can be useful if another transaction is likely to create the same index, so that the second index-create will simply fail instead of potentially causing many

Value	Identifier	Description
		unnecessary database operations. The second transaction may not be able to use the index immediately. The index creation operation needs to complete before it is usable. The session must not currently be in a transaction to create an index without version information.
0x00000400	JET_bitIndexSortNullsHigh	Specifying this flag causes NULL values to be sorted after data for all columns in the index.
0x00000800	JET_bitIndexUnicode	Specifying this flag affects the interpretation of the lcid/pidxunicde union field in the structure. Setting the bit means that the pidxunicode field actually points to a JET_UNICODEINDEX structure. See JET_UNICODEINDEX. JET_bitIndexUnicode is not required to index Unicode data. It is only needed to customize the normalization of Unicode data.
Values introduced in Windows XP		
0x00001000	JET_bitIndexTuples	Specifies that the index is a tuple index. See JET_TUPLELIMITS for a description of a tuple index.
Values introduced in Windows 2003		
0x00002000	JET_bitIndexTupleLimits	Specifying this flag affects the interpretation of the cbVarSegMac/ptuplelimits union field in the structure. Setting this bit means that the ptuplelimits field actually points to a JET_TUPLELIMITS struct to allow custom tuple index limits (implies JET_bitIndexTuples). See JET_TUPLELIMITS.
Values introduced in Windows Vista		
0x00004000	JET_bitIndexCrossProduct	Specifying this flag for an index that has more than one key column that is a multi-valued column will result in an index entry being created for each result of a cross product of all the values in those key columns. Otherwise, the index would only have one entry for each multi-value in the most significant key column that is a multi-valued column and each of those index entries would use the first multi-value from any other key columns that are multi-valued columns. For example, if you specified this flag for an index over column A that has the values "red" and "blue" and over column B that has the values "1" and "2" then the following index entries would be created: "red", "1"; "red", "2"; "blue", "1"; "blue", "2". Otherwise, the following index entries would be created: "red", "1"; "blue", "1".

Value	Identifier	Description
0x00008000	JET_bitIndexKeyMost	Specifying this flag will cause the index to use the maximum key size specified in the cbKeyMost field in the structure. Otherwise, the index will use JET_cbKeyMost (255) as its maximum key size.
0x00010000	JET_bitIndexDisallowTruncation	Specifying this flag will cause any update to the index that would result in a truncated key to fail with JET_errKeyTruncated. Otherwise, keys will be silently truncated. For more information on key truncation, see the JetMakeKey function.

11. Notes

11.1. The database metadata table

The database metadata table contains space tree information about the database. The database metadata table is always stored as FDP object identifier 1 with parent FDP page number 1.

Appendix A. References

[MSDN]
Title: Microsoft Developer Network
URL: http://technet.microsoft.com/en-us/library/bb310772%28EXCHG.80%29.aspx
URL: http://technet.microsoft.com/en-us/library/cc961824.aspx
URL: http://msdn.microsoft.com/en-us/library/dd207764(v=PROT.13).aspx
URL: http://msdn.microsoft.com/en-us/library/ee441458(v=PROT.13).aspx

[MSDN-WIN7]
Title: 6 New ESENT features in Windows 7
URL: http://blogs.msdn.com/b/laurionb/archive/2009/08/18/6-new-esent-features-in-windows-7.aspx

[NTLCID]
Tile: Locale identifier (LCID) definitions
URL: https://downloads.sourceforge.net/project/libpff/documentation/MAPI%20definitions/

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.