

How to restore the injured zip files

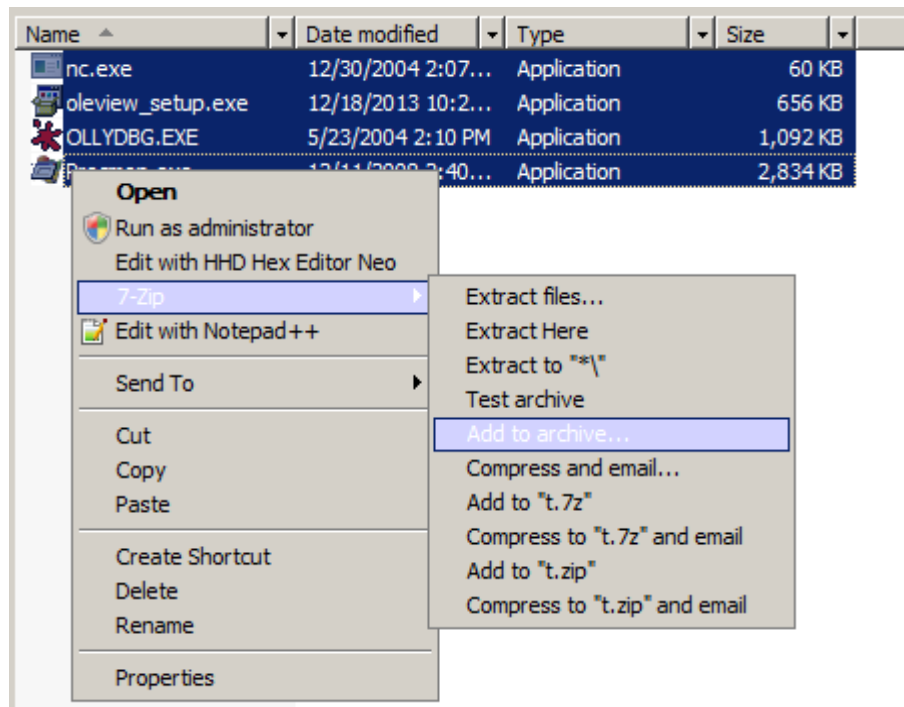
Table of Contents

How to restore the injured zip files.....	1
single part zip file.....	2
Create a sample file.....	2
Zip file structure.....	6
Central directory entry structure.....	8
Local file header structure.....	9
Reconstruction of central directory entries.....	13
End of central directory structure.....	15
Reconstruction of "End of central directory" entry.....	16
Multipart zip file.....	18
Create a sample file.....	18
Restore the central directory structure.....	23
End of central directory structure.....	31
Reconstruction of "End of central directory" entry.....	31

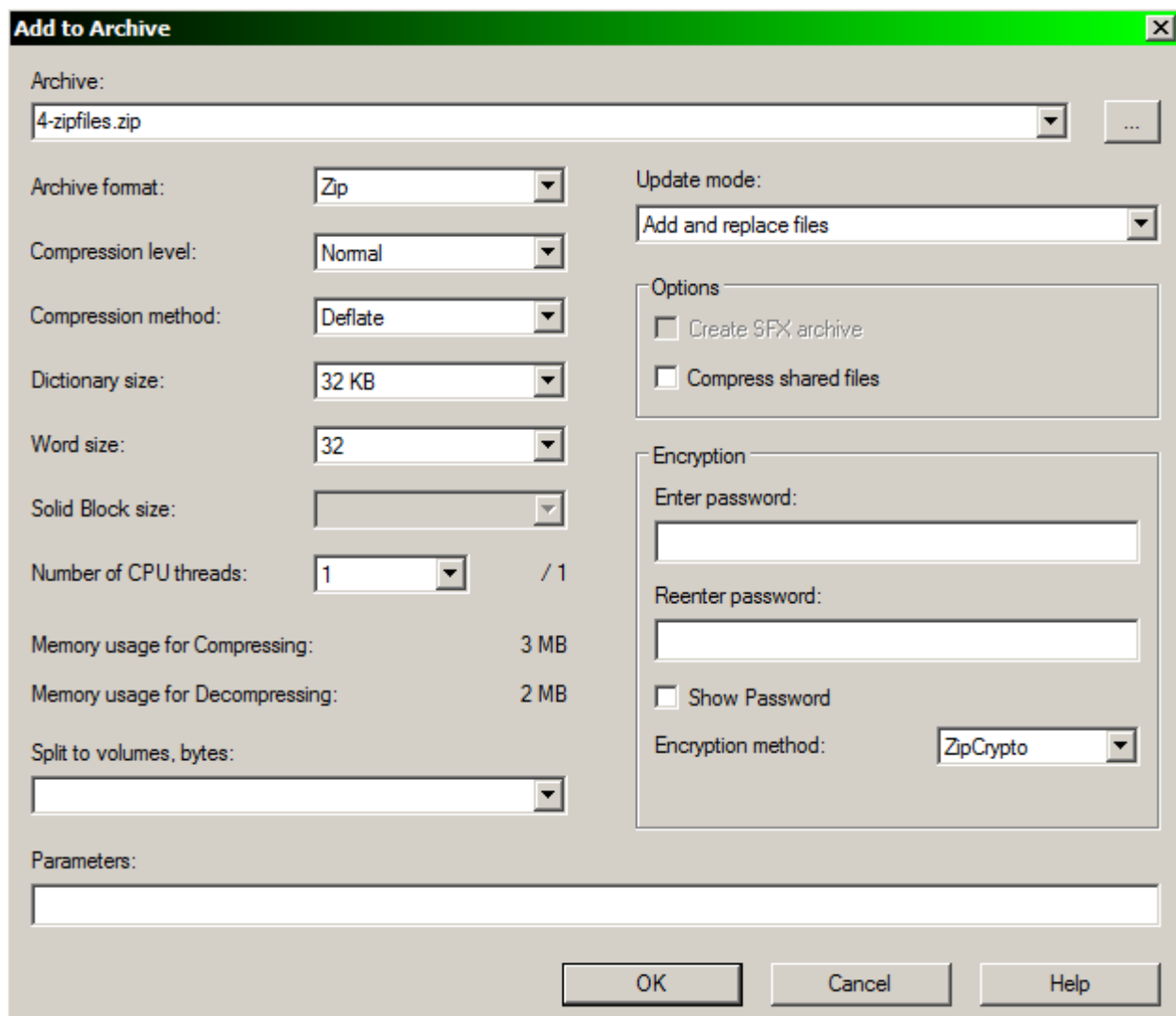
single part zip file

Create a sample file

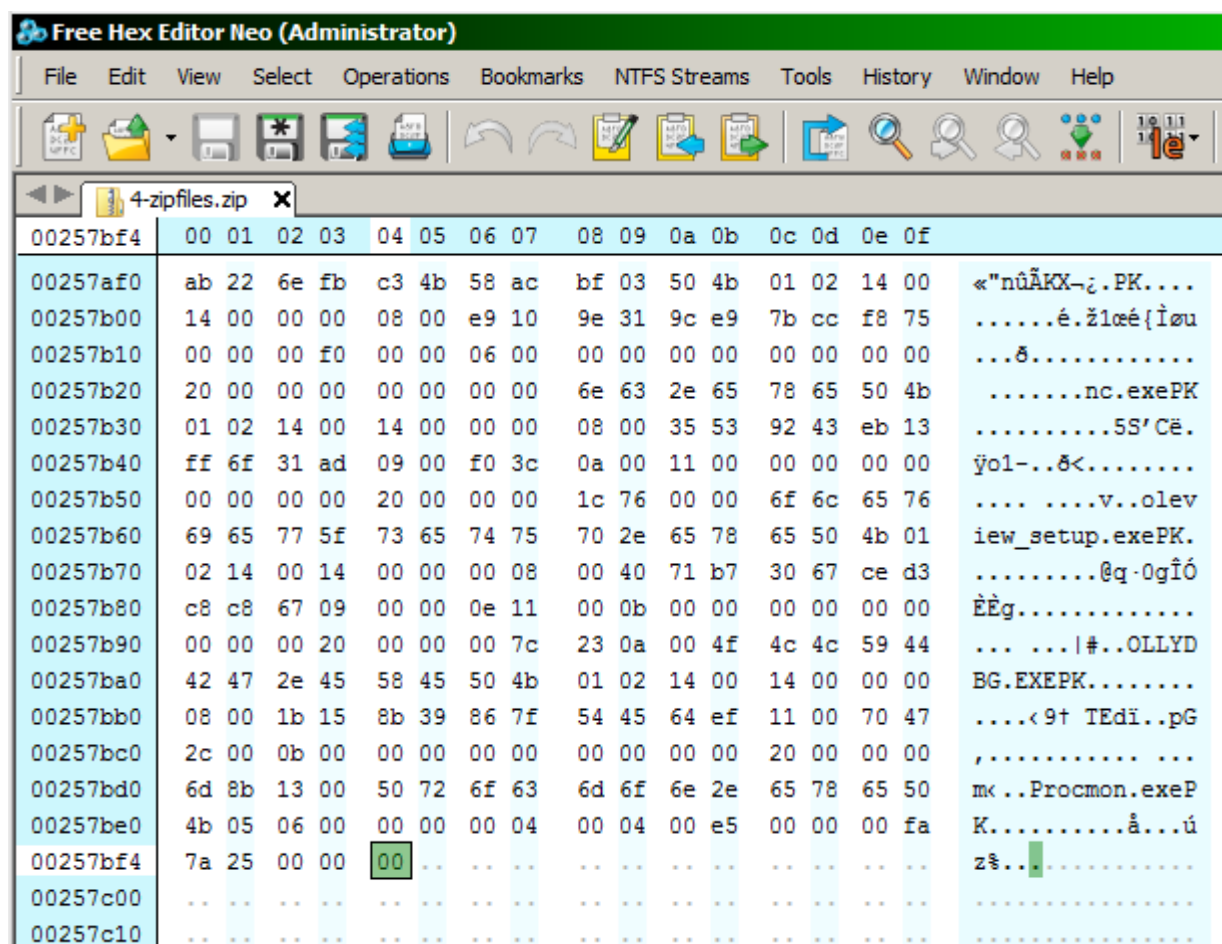
Our first task is to create a zip file. To do it select some files, and compress them with your favourite zip archiver.



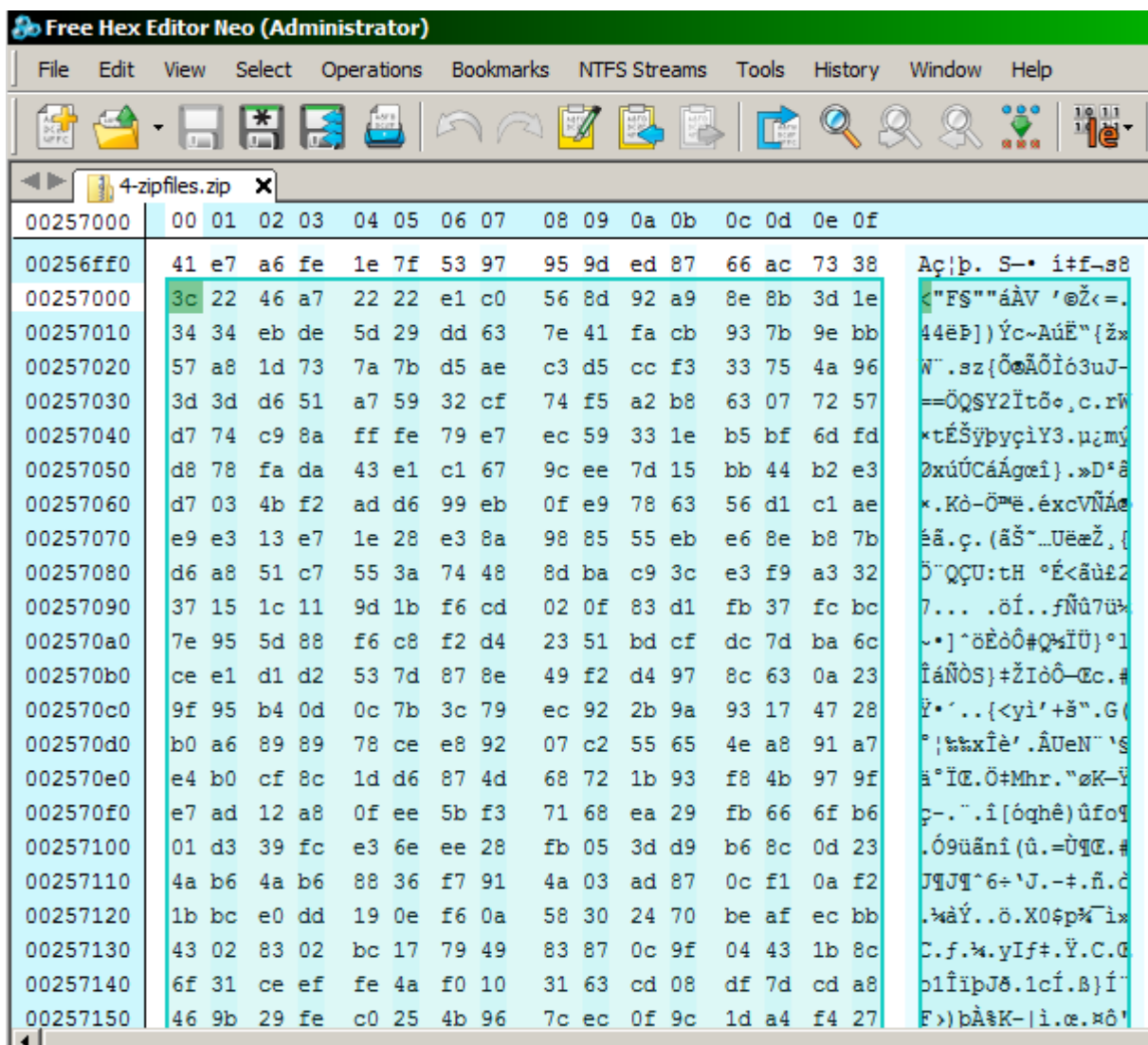
Set the parameters, then click to the OK button, to compress the files.



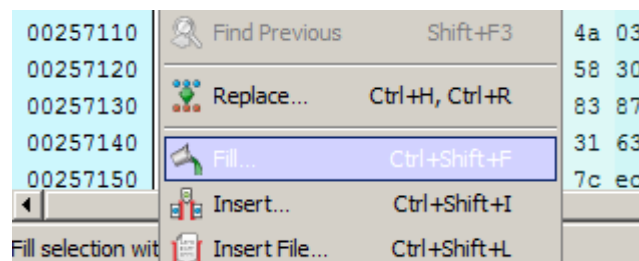
Open the file in a hex editor, and go to the end of the file:



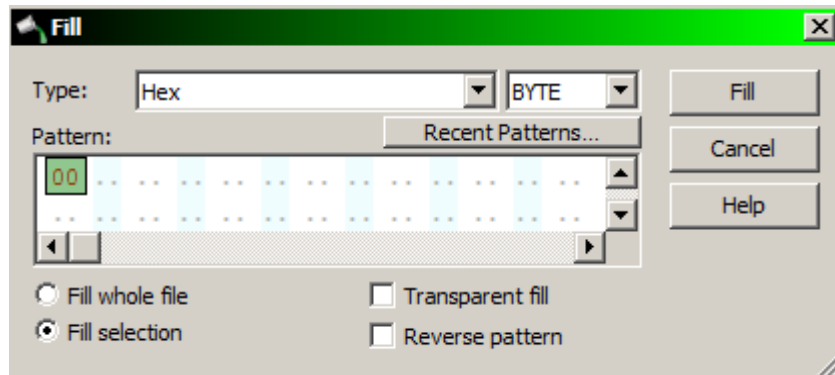
To simulate the destroyed last cluster go back from the end of the file until the first round 0x1000 value (0x1000 = 4096 the default cluster size). For me it will be the position 0x00257000. Select this range



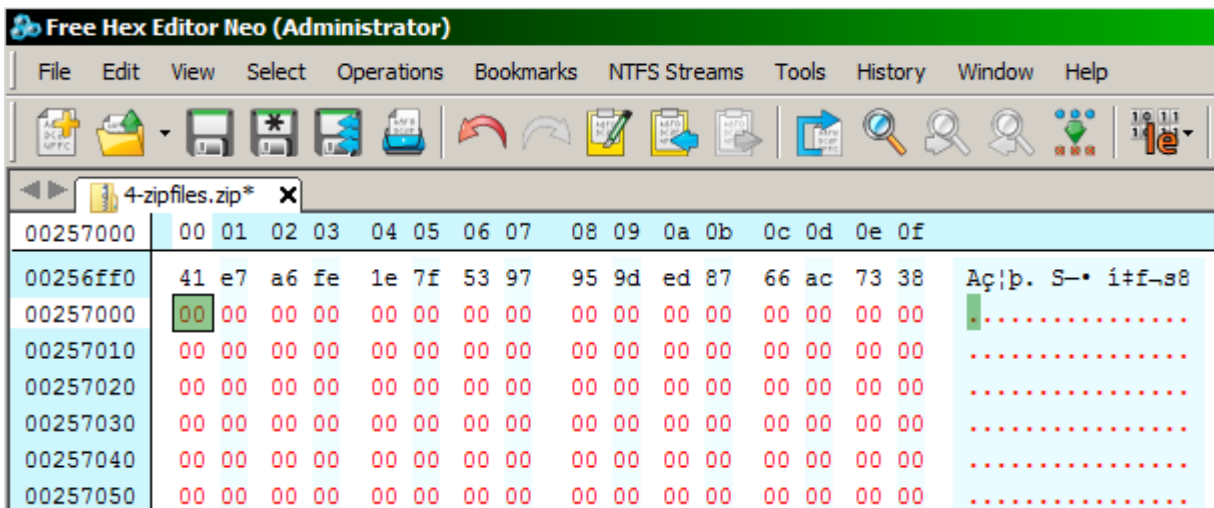
Select the fill command, and fill the range until the end of the file with 00 byte, to simulate the destroyed last cluster.



Type the fill pattern, and click to the Fill button

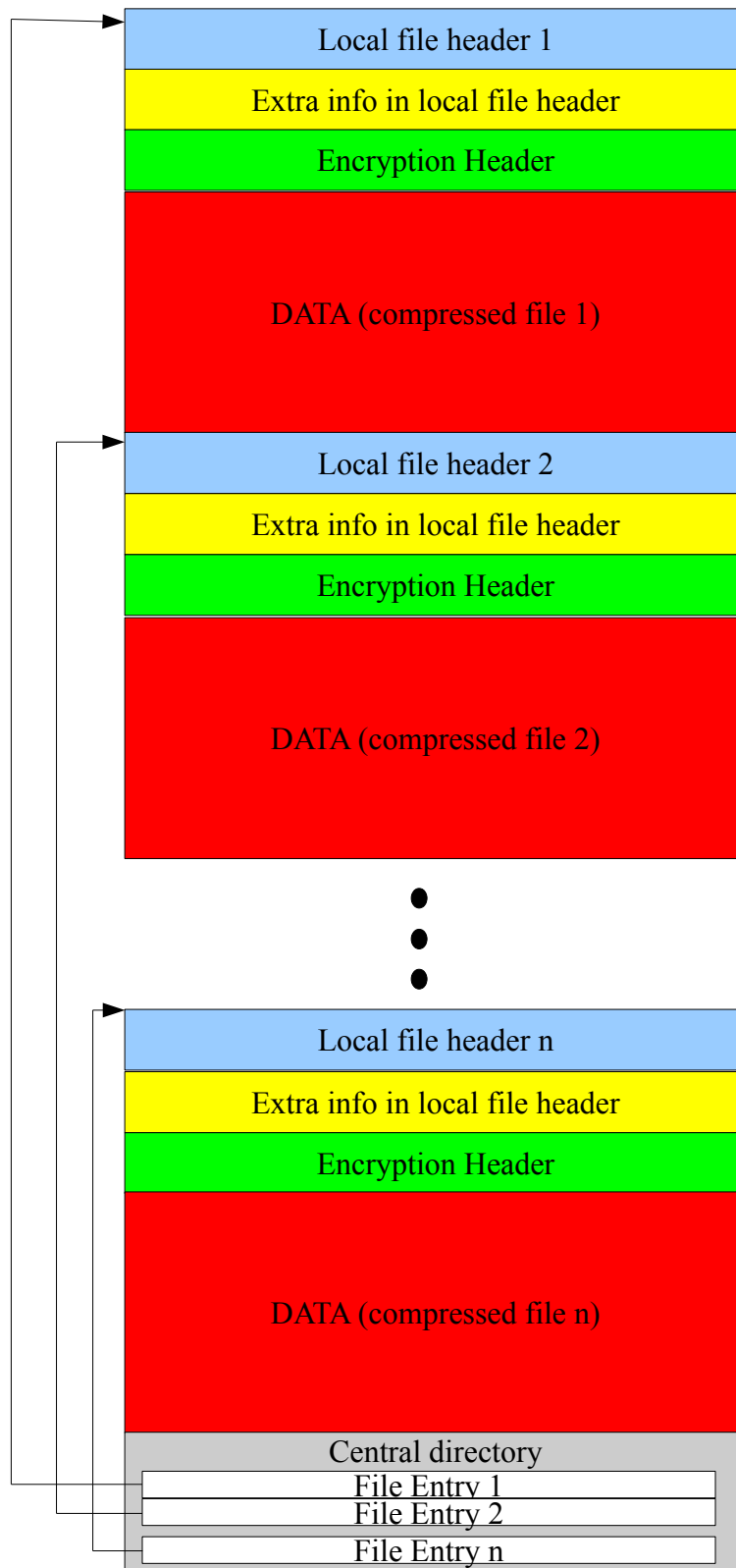


then click to the save button, to save the modifications



Zip file structure

To be able to restore a zip file we should know the structure of it. From high view the zip file looks like as follows:

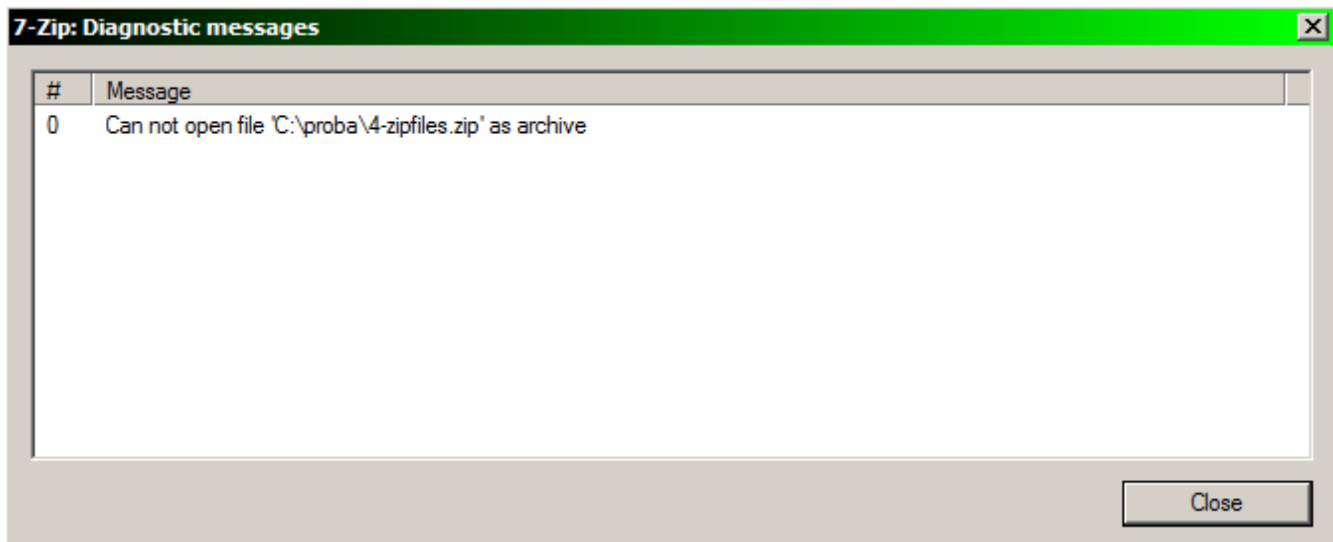


As we can see the catalog info is at the end of the zip file. Practically it means, if the beginning of the

file injures, that not so big problem, only the file there can not be decompressed, but the others will work fine. See 1-zipfiles.zip, where the first 512 bytes of the file is overwritten with 0 simulating a bad sector on the disk, and 2-zipfiles.zip, where the first 4096 bytes is overwritten by 0 simulating a bad cluster at the beginning of the file. Both of these files can be decompressed with any zip tool, just one file will be wrong

If the file is injured at the middle then the situation is the same. See 2-zipfiles.zip. All the files not injured can be extracted easily with any zip tool. The one file injured will be obviously wrong, the others will work fine.

There will be problem, if the Central directory at the end of the file is injured. In this case will will get for example the following error message with 7zip, and none of the files can be extracted



The problem can be fixed by fixing the Central directory structure at the end of the file. To be able to do it we must know the structure of the Central directory which is the following:

Central directory entry structure

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	Central directory file header signature = 0x02014b50				Version made by		Version needed to extract (minimum)		General purpose bit flag		Compression method		File last modification time		File last modification date	
0x0010	CRC-32				Compressed size				Uncompressed size				File name length		Extra field length	
0x0020	File comment length		Disk number where file starts		Internal file attributes		External file attributes				Relative offset of local file header. This is the number of bytes between the start of the first disk on which the file occurs, and the start of the local file header				Filename	
0x0030	filename continue								Extra field							
0x0040	File Comment															

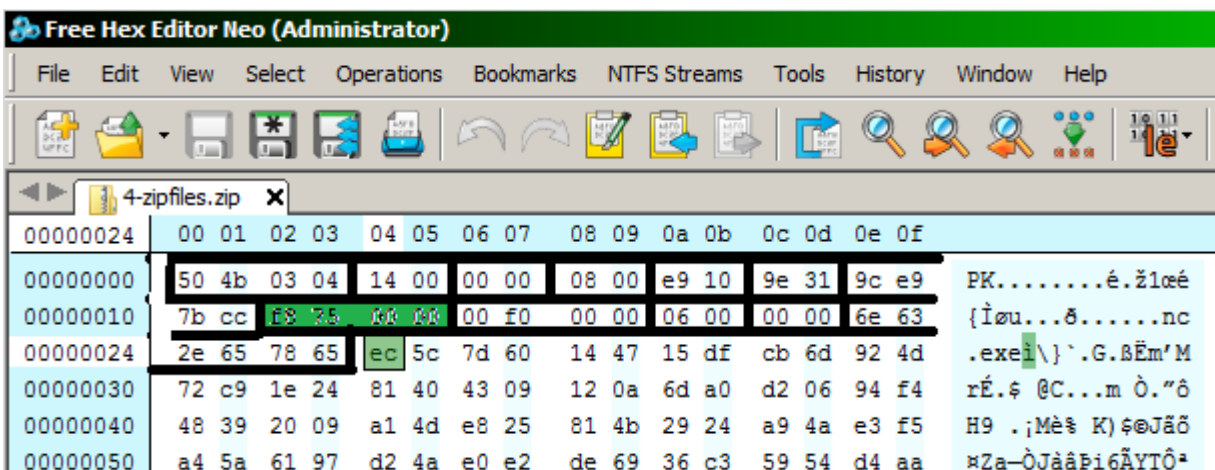
So we must create this structure. The question is where to create it? Fortunately it can be found quite easily, there are the local file headers, and we can reassemble the central directory from these structures.

Local file header structure

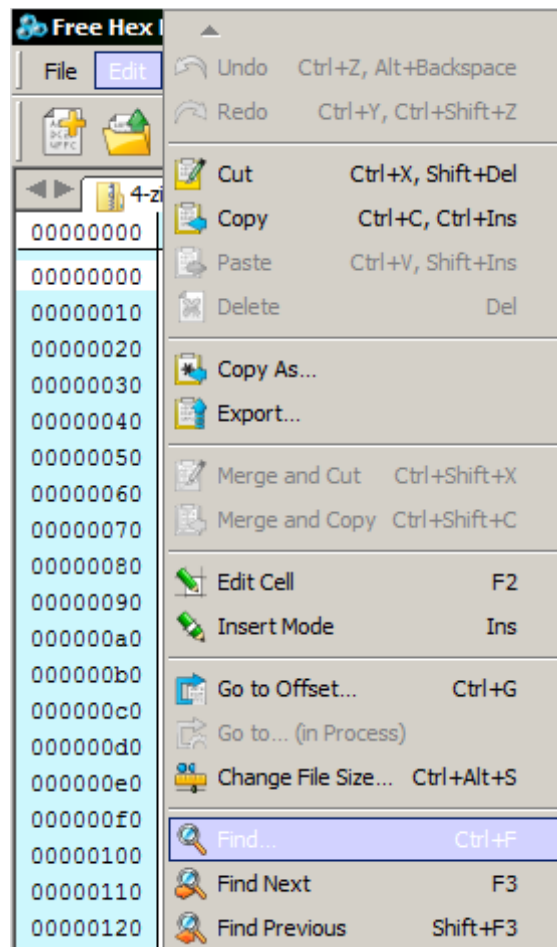
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0x0000	Local file header signature = 0x04034b50 (read as a little-endian number)				Version needed to extract (minimum)		General purpose bit flag		Compression method		File last modification time		File last modification date		CRC-		
0x0010	32		Compressed size				Uncompressed size				File name length		Extra field length		File		
0x0020	name					Extra field											

As we can see every local file header contains the compressed size. So we should find the last local file entry, and and the compressed size to the end of it. The Central directory structure will start there.

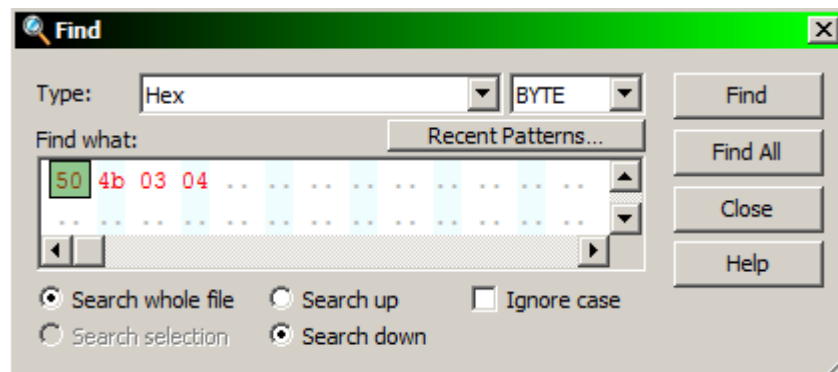
The central file directory should contain every file entry, so it is easier, to collect all the information in one round, not first find only the last one. In my case it will be the following:



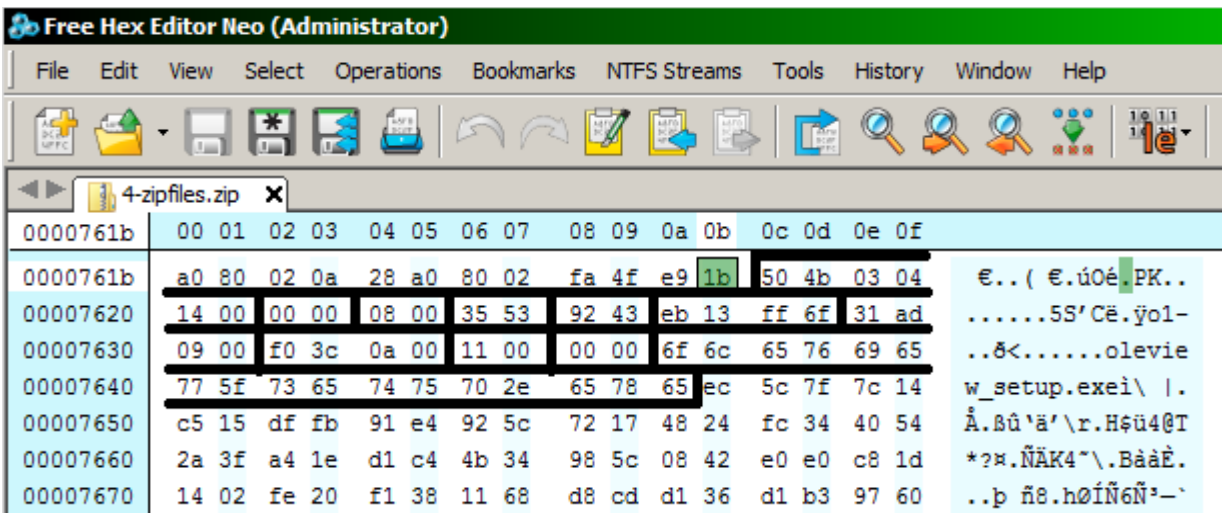
Then we search for the next local file header entry:



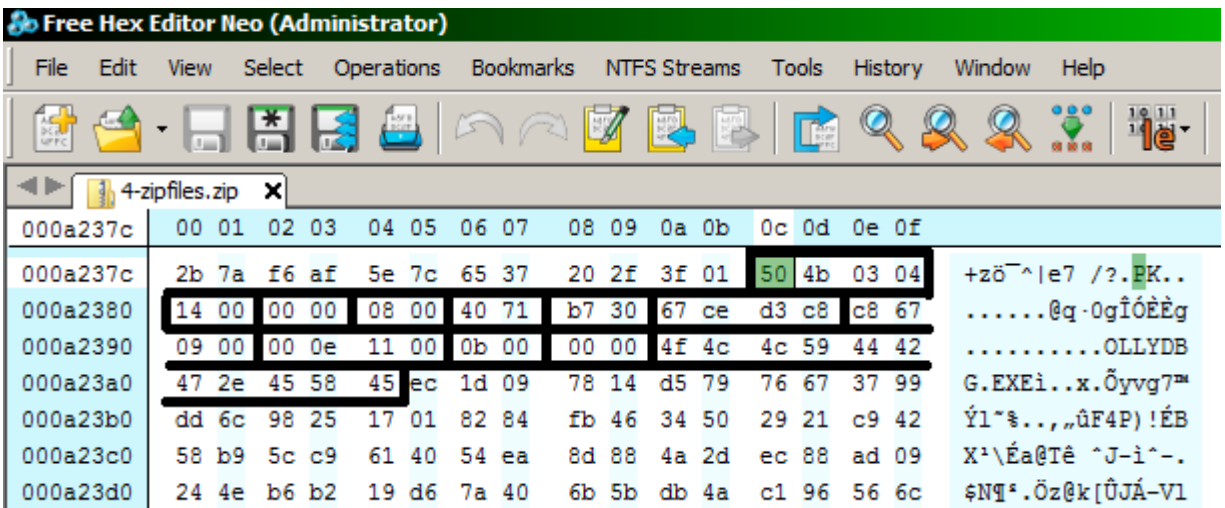
search for the 50 4b 03 04 (local file header):



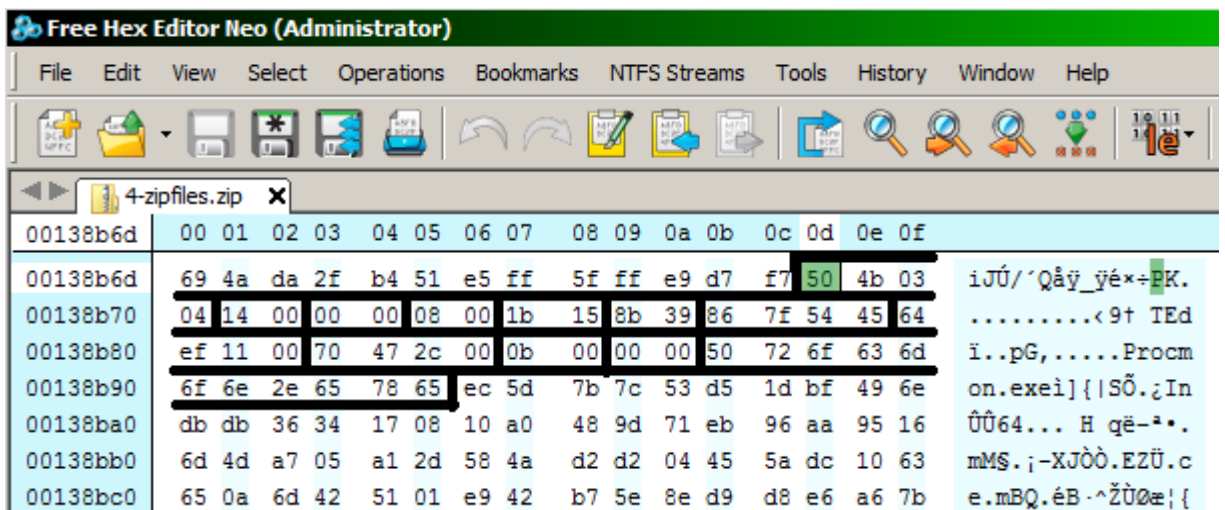
Then we find the second file entry:



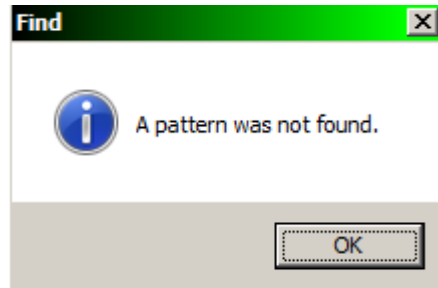
By pressing the F3 button we will find the third file entry:



Pressing again the F3 button we will find the fourth file entry:

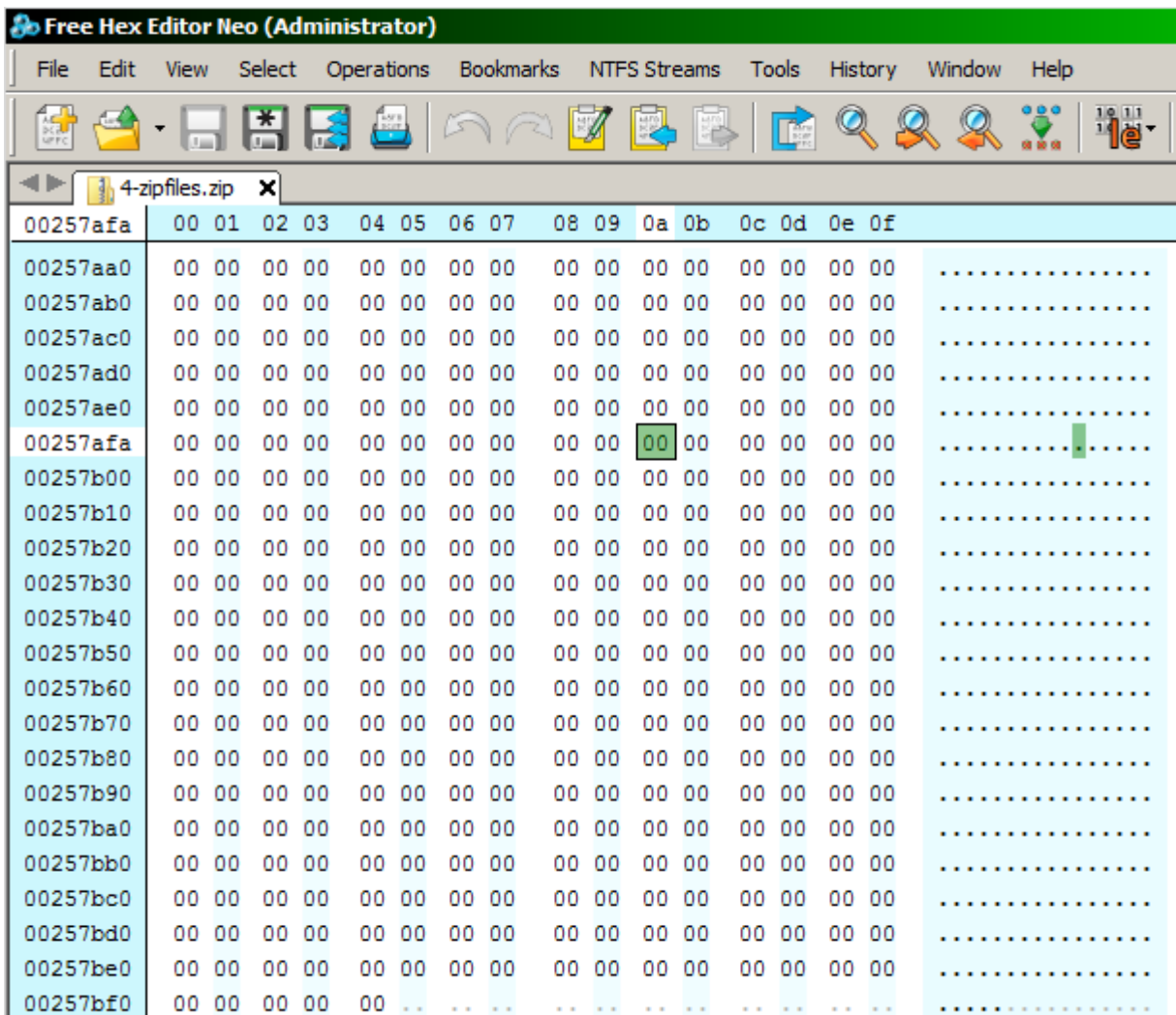


We will not find more entries:



Now we can calculate the start of the center directory structure: it will be $0x0011EF64 + 0x00138B96 = 0x00257AFA$

if we go there we will see the following:



This part of the file is zeroed out (it simulates the injured last cluster).

Now we must write back the data to here from the four local file header:

Reconstruction of central directory entries

let us compare the two headers.

Central directory:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	Central directory file header signature = 0x02014b50				Version made by		Version needed to extract (minimum)		General purpose bit flag		Compression method		File last modification time		File last modification date	
0x0010	CRC-32				Compressed size				Uncompressed size				File name length		Extra field length	
0x0020	File comment length		Disk number where file starts		Internal file attributes		External file attributes				Relative offset of local file header. This is the number of bytes between the start of the first disk on which the file occurs, and the start of the local file header				Filename	
0x0030	filename continue								Extra field							
0x0040	File Comment															

local file header:

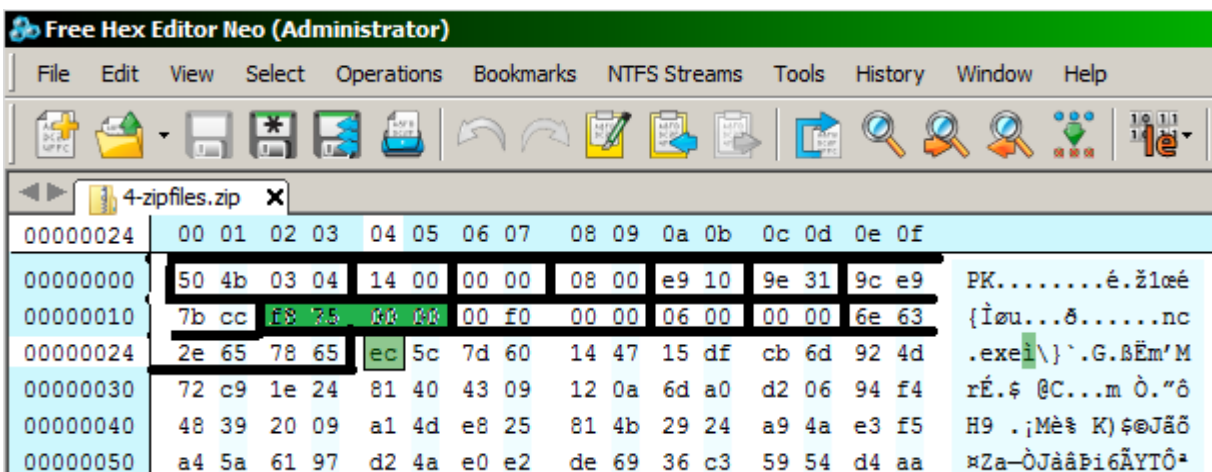
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	Local file header signature = 0x04034b50 (read as a little-endian number)				Version needed to extract (minimum)		General purpose bit flag		Compression method		File last modification time		File last modification date		CRC-	
0x0010	32		Compressed size				Uncompressed size				File name length		Extra field length		File	
0x0020	name					Extra field										

as we can see the the beginning of the two header are almost identical:

- **signature:** we know it
- **version made by:** we do not know, but if we use the same as version to extract that should be good.
- **Version need to extract:** we know
- **General purpose bit flag:** we know
- **compression method:** we know
- **file last modification time:** we know
- **file last modification date:** we know
- **CRC-32 checksum:** we know
- **compressed size:** we know
- **uncompressed size:** we know
- **file name length:** we know
- **extra field length:** we know

- **file comment lengt:** we do not know, but we create a new central directory, so we can just left the comment as empty
- **disk number where it starts:** we have one disk only, so it will be zero
- **internal file attributes:** lowest bit: one if ASCII text 0 if binary. Second lowest bit: if set, that a 4 byte variable record length control field precedes each logical record indicating the length of the record. Now all the files are binary so this field is always zero
- **external file attributes:** this is a host dependent filed. On windows system use 0x2000000 means the standard file attributes.
- **Relative offset:** address of the beginning of local file header corresponding to this entry.
- **Filename:** we know
- **Extra field:** we know
- **File comment:** we set it to zero.

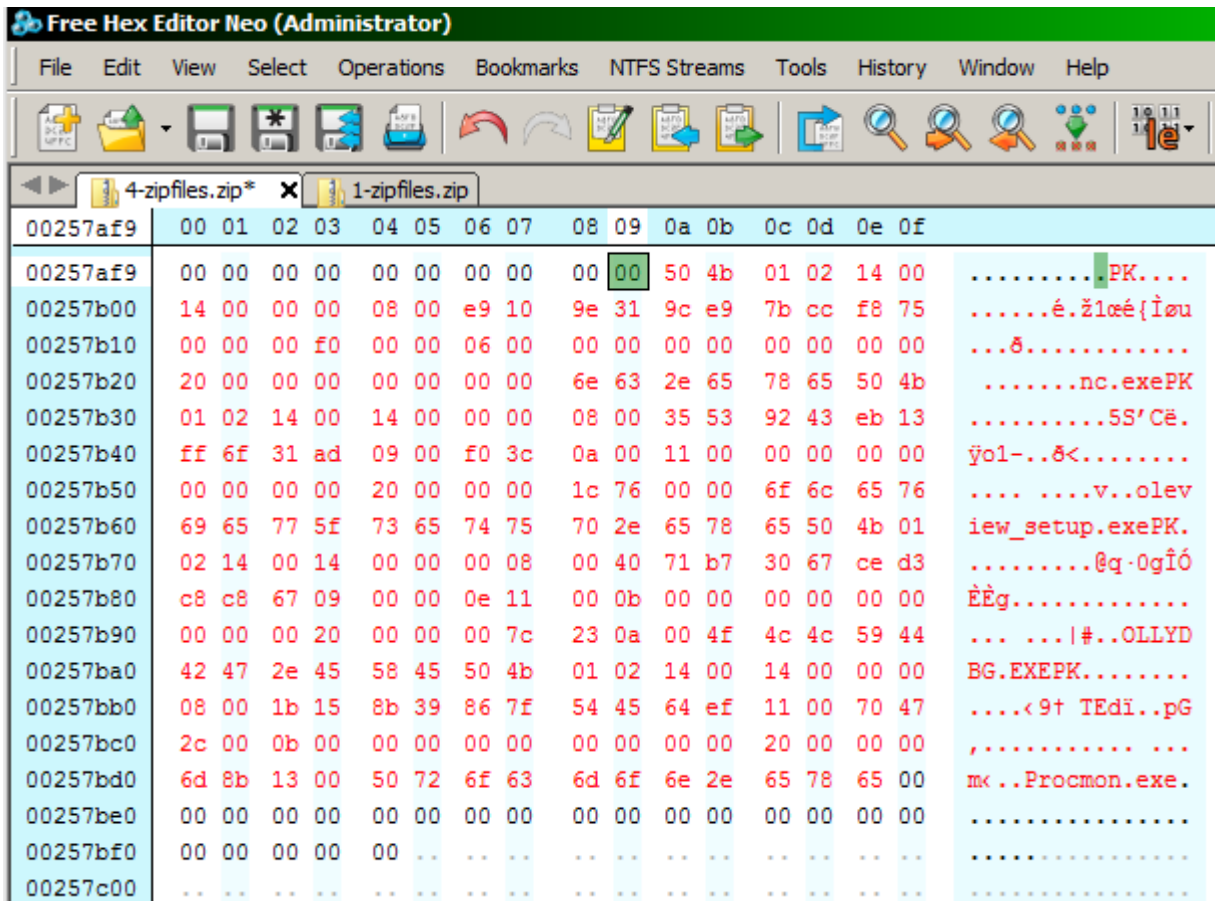
Based on these knowledge in my case we get the following central directory entry:
for the first file it is:



- **signature:** 50 4b 01 02
- **version made by:** 14 00
- **Version need to extract:** 14 00
- **General purpose bit flag:** 00 00
- **compression method:** 08 00
- **file last modification time:** E9 10
- **file last modification date:** 9E 31
- **CRC-32 checksum:** 9C E9 7B CC
- **compressed size:** F8 75 00 00
- **uncompressed size:** 00 F0 00 00
- **file name length:** 06 00
- **extra field length:** 00 00
- **file comment length:** 00 00
- **disk number where it starts:** 00 00
- **internal file attributes: lowest bit:** 00 00
- **external file attributes:** 20 00 00 00
- **Relative offset:** 00 00 00 00

- **Filename:** 6E 63 2E 65 78 65
- **Extra field:** EMPTY
- **File comment:** EMPTY

similarly we can just read and substitute the values for the other files from the local file headers corresponding to that central directory entries.



End of central directory structure

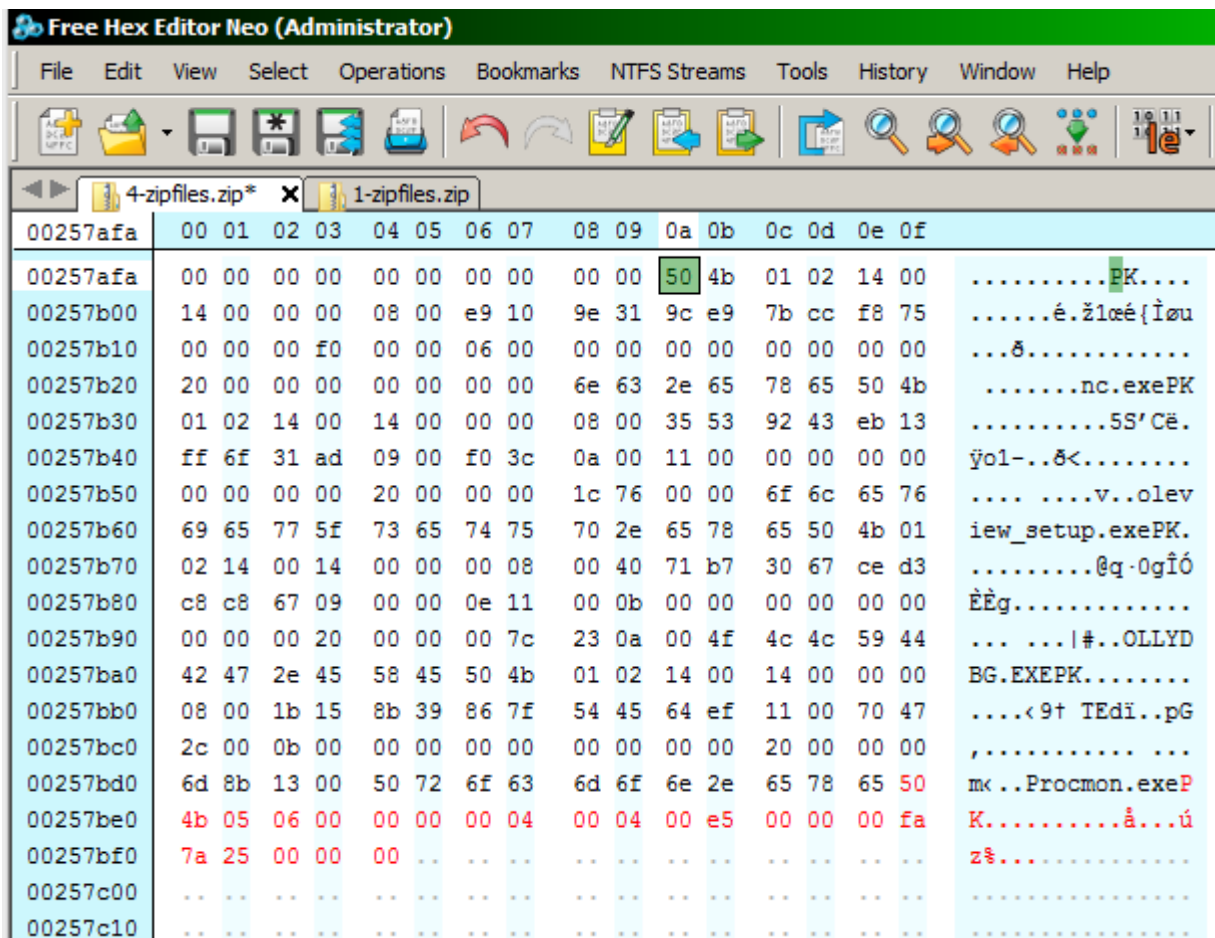
As one can see there are some bytes not filled yet after the four central directory entry. It is the "End of central directory" entry, what has the following structure:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	End of central directory signature = 0x06054b50				Number of this disk		Disk where central directory starts		Number of central directory records on this disk		Total number of central directory records		Size of central directory (bytes)			
0x0010	Offset of start of central directory, relative to start of archive				Comment length		comment									

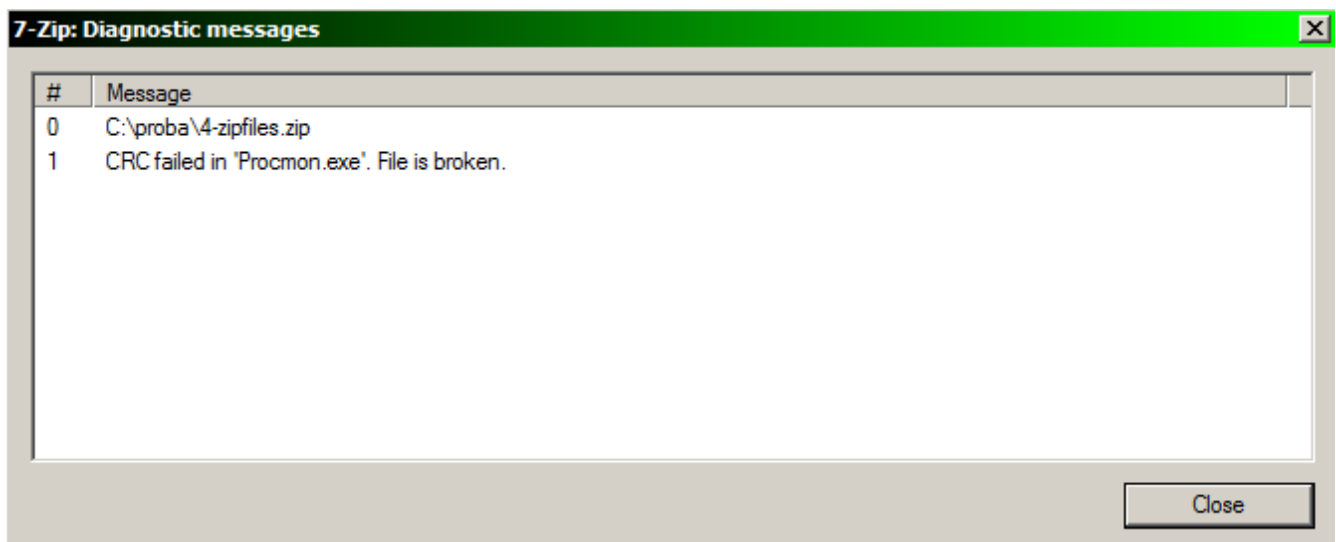
Reconstruction of "End of central directory" entry

Based on this information we can generate the "end of central directory" entry:

- **signature:** 50 4B 05 06
- **number of disks:** now we have disk so it is 00 00
- **disk where the central directory starts:** we have only one zip file so it is 00 00
- **number of central directory records on this disk:** 04 00
- **total number of central directory records:** 04 00
- **size of central directory:** E5 00 00 00
- **offset of start of the central directory we have already calculated:** FA 7A 25 00
- **comment length:** 00 00
- **comment:** EMPTY



After we added back the central directory entry we can extract the files. (some portion of the last file is also injured so obviously that file will not be correct, but the first three can be extracted successfully).



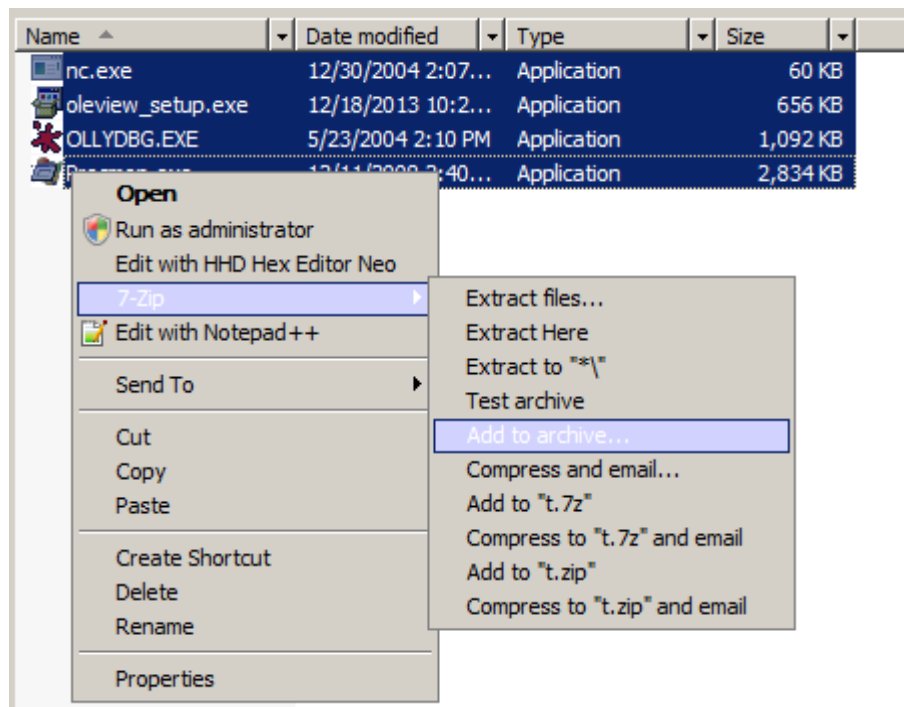
But the other files are fine.

Multipart zip file

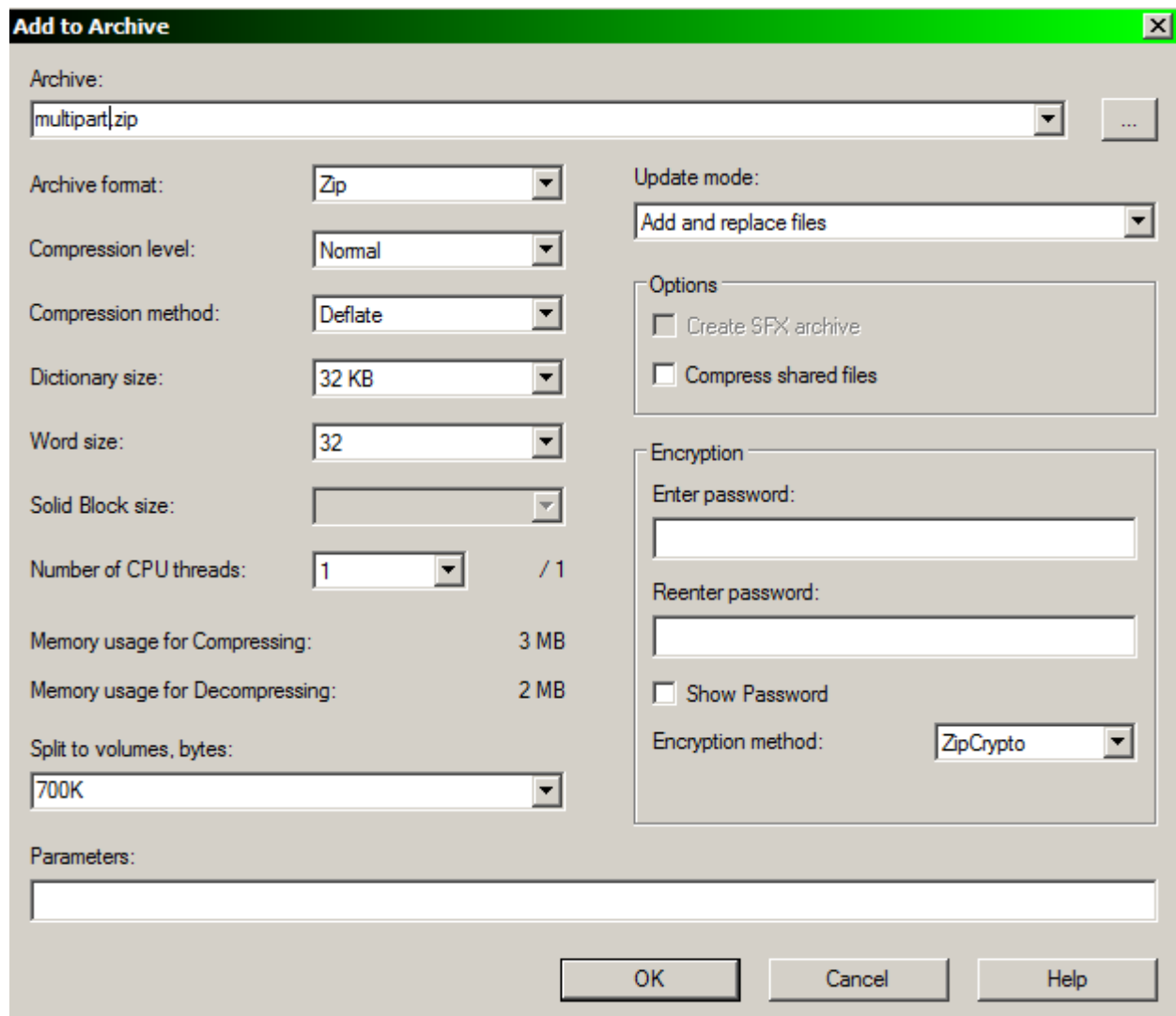
What you should remember, the disk number is not the file number. the disk number will be still 00. it will be treated like one zip is created, then cut to the necessary number of peaces.

Create a sample file

Our first task is to create a multipart zip file. To do it select some files, and compress them with your favourite zip archiver.



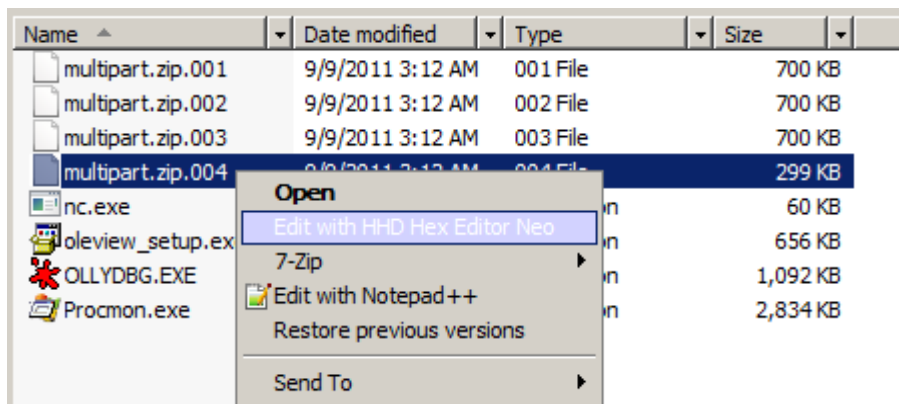
Set the split size to an appropriate value, to get about 3-5 files.



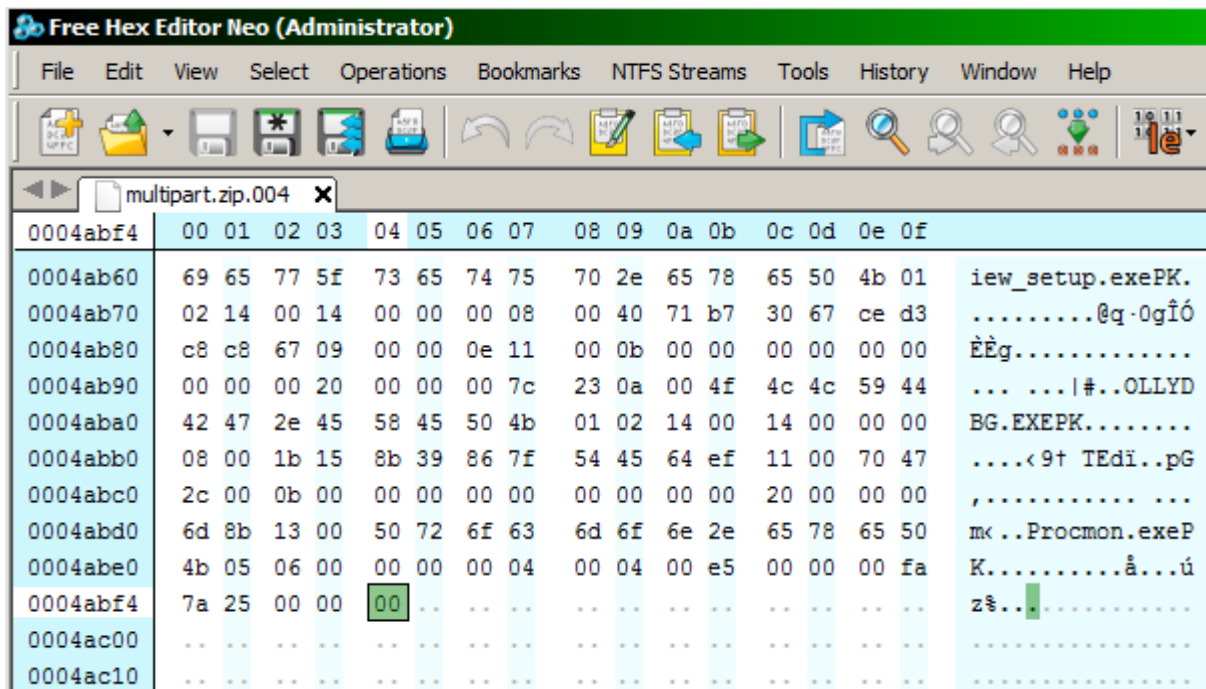
I got a four part zip file

Name	Date modified	Type	Size
multipart.zip.001	9/9/2011 3:12 AM	001 File	700 KB
multipart.zip.002	9/9/2011 3:12 AM	002 File	700 KB
multipart.zip.003	9/9/2011 3:12 AM	003 File	700 KB
multipart.zip.004	9/9/2011 3:12 AM	004 File	299 KB
nc.exe	12/30/2004 2:07...	Application	60 KB

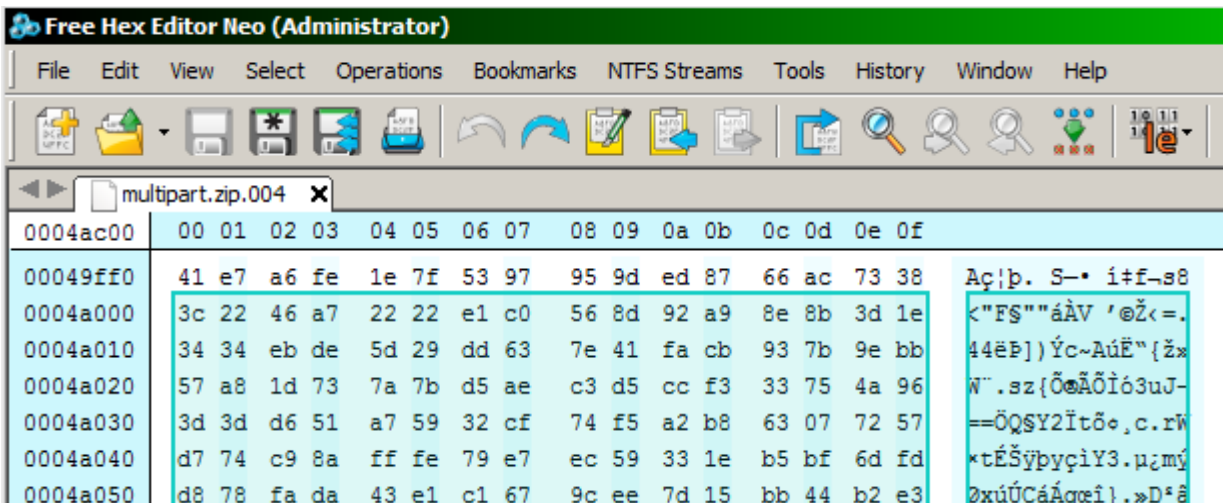
now simulate the injure of the last cluster of the last file (there is the central directory structure). Open the last zip file with your favourite hex editor



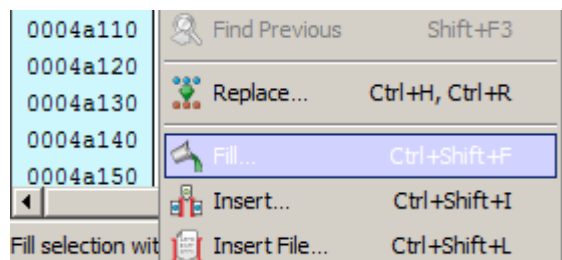
Go to the end of the file:



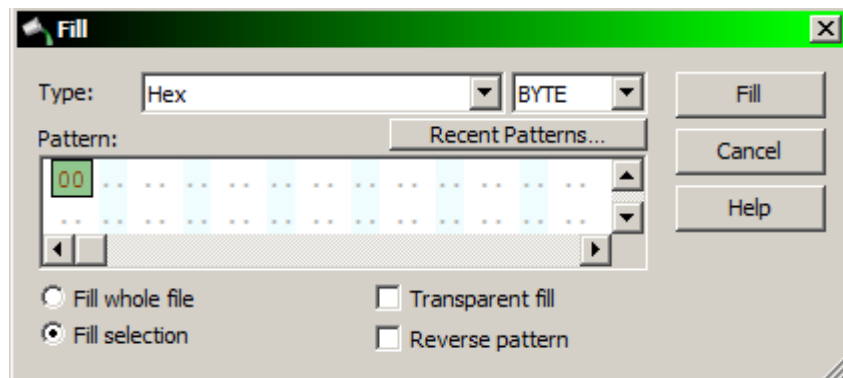
To simulate the destroyed last cluster go back from the end of the file until the first round 0x1000 value (0x1000 = 4096 the default cluster size). For me it will be the position 0x0004a000. Select this range



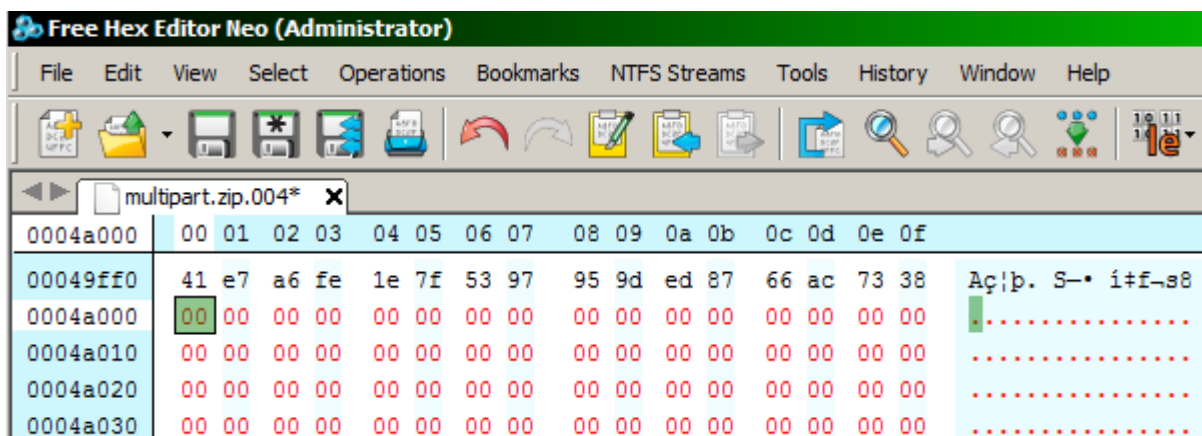
And fill the file until the end with 00 byte. To do it select Edit / fill:



For pattern use the 00 byte, and fill the previously selected area:

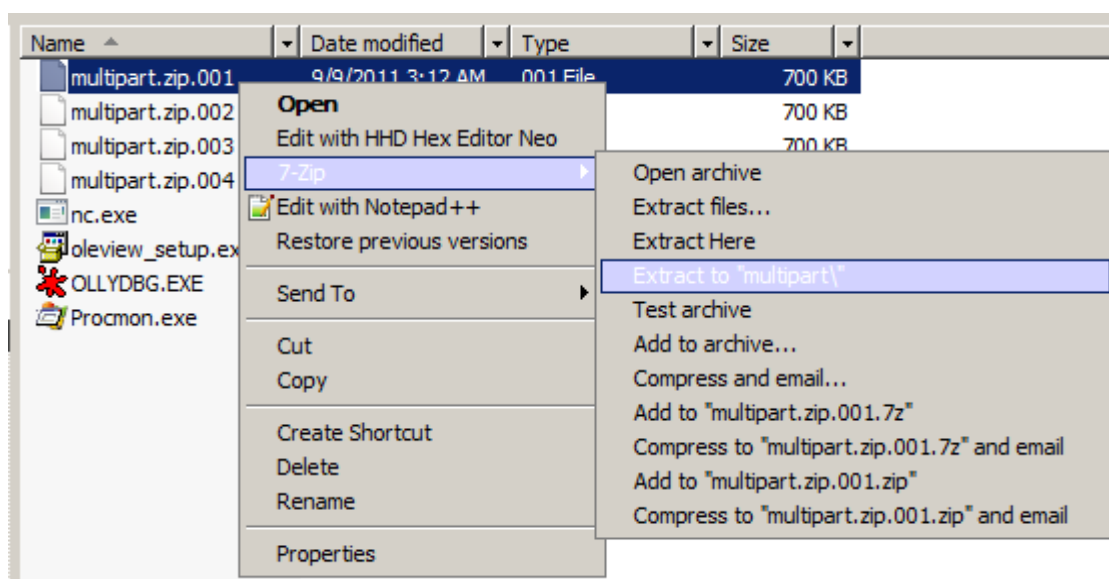


Then save the file:

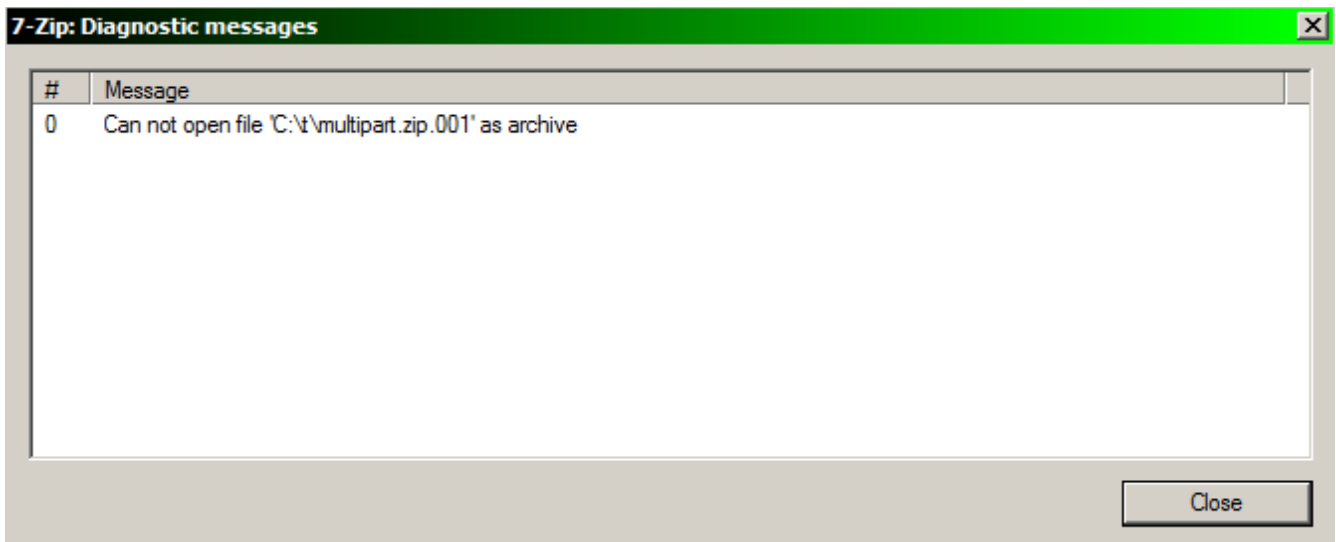


Then save the file

If you try to extract it:



Hopefully it will not work:



Restore the central directory structure

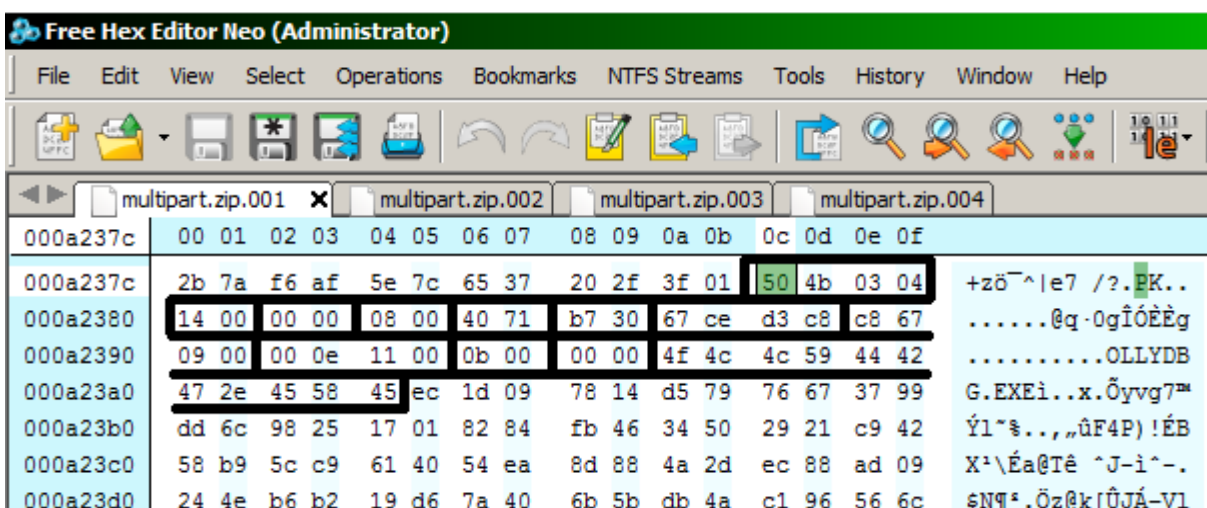
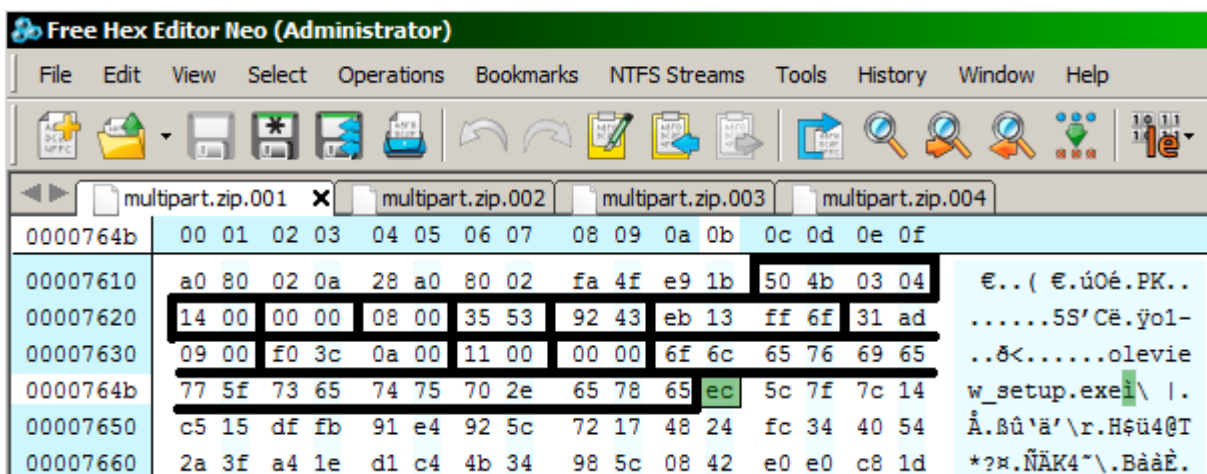
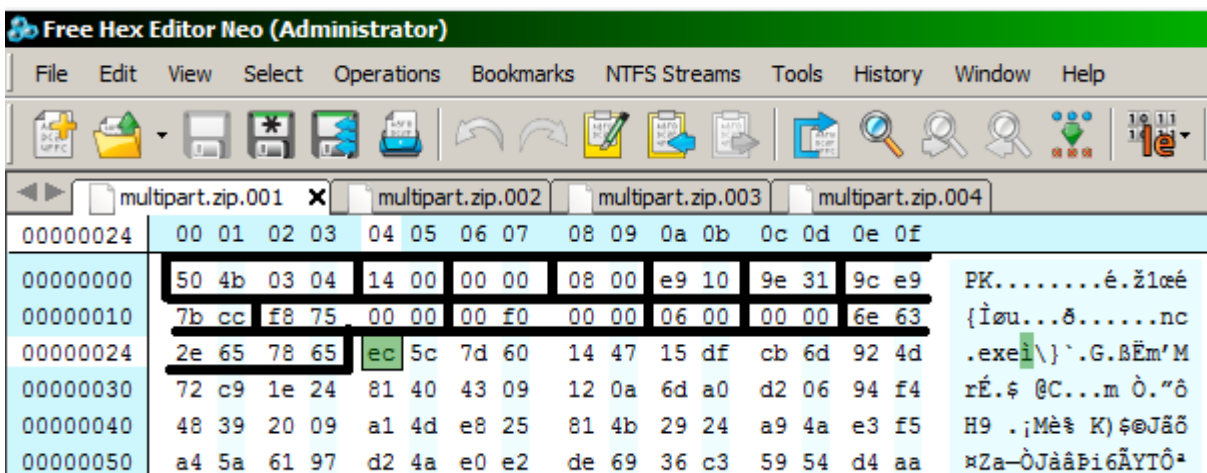
As we did in case of the single part zip file search for the local file headers in every zip file.

only for remaind the local file header structure looks like as follow:

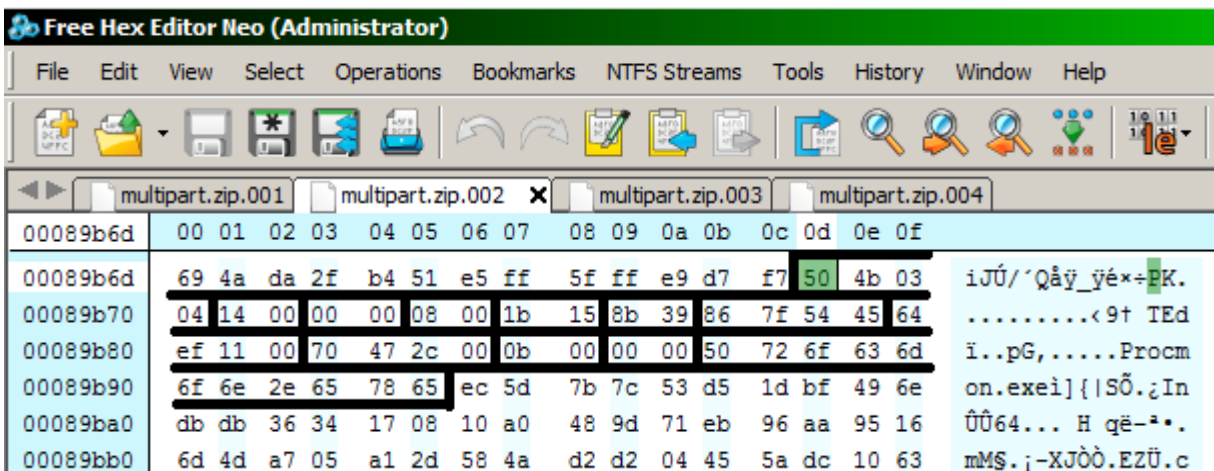
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0x0000	Local file header signature = 0x04034b50 (read as a little-endian number)				Version needed to extract (minimum)		General purpose bit flag		Compression method		File last modification time		File last modification date		CRC-		
0x0010	32		Compressed size				Uncompressed size				File name length		Extra field length		File		
0x0020	name					Extra field											

So open every zip file, and search for the signature 50 4b 03 04 in them, I got the following:

in the first file:



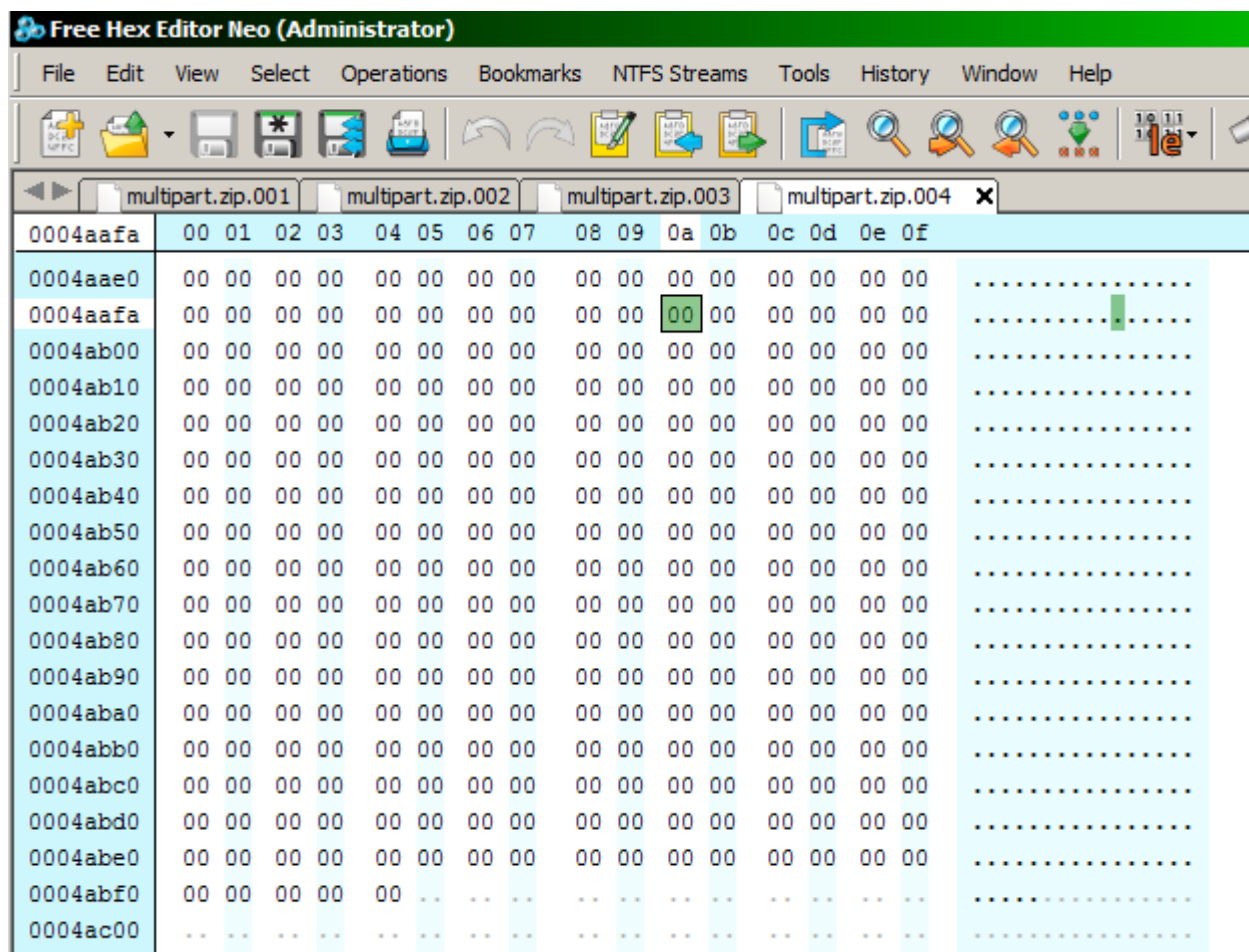
in the second file:



There are no more local file entry obviously, because I added four files.

Now let us calculate the position of the central directory structure. The compressed size of the last entry is 0x0011EF64 and it starts at position 0x00089B96. So the central directory structure starts at: $0x0011EF64 + 0x00089B96 = 0x001A8AFA$. But this calculation is totally wrong, because, it does not encounter the file size limit. The file size is 0x000AF000. Let us consider the filesize limit. In the 002 file we have $0x000AF000 - 0x00089B95 = 0x0002546B$ bytes space left. If we subtract it from the required size: $0x0011EF64 - 0x0002546B = 0x000F9AF9$. The length limit of the file 003 is 0x000AF000 smaller than the required size. So the central directory will be in the fourth file. At position $0x000F9AF9 - 0x000AF000 = 0x0004AAF9$.

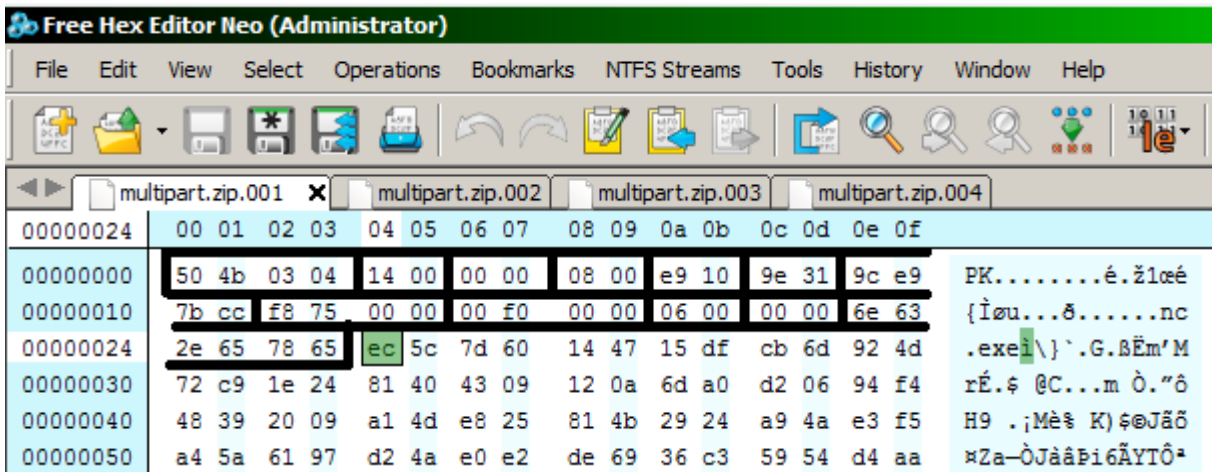
Let us jump to that position:



And start to substitute back the information from the local file headers to the central directory structures looks like as follows:

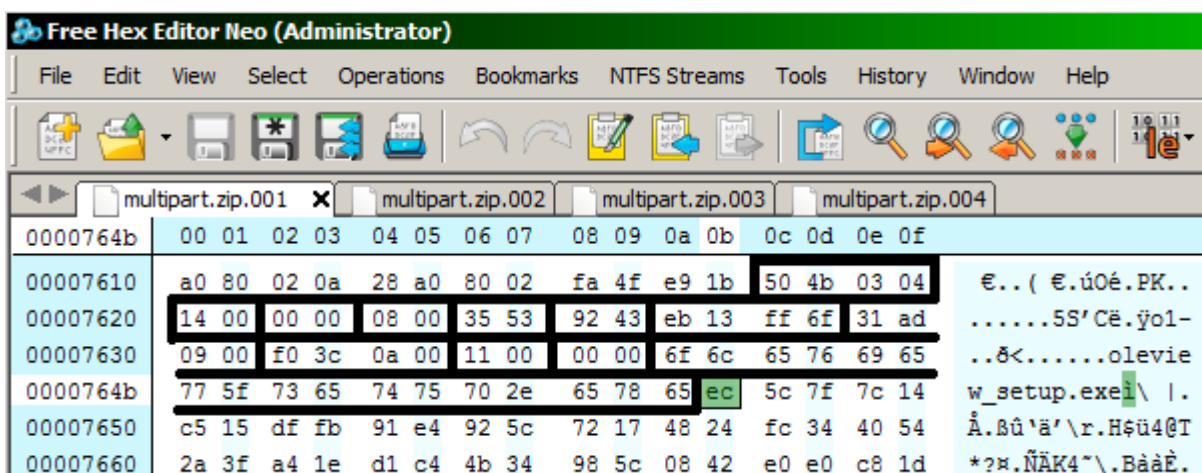
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	Central directory file header signature = 0x02014b50				Version made by		Version needed to extract (minimum)		General purpose bit flag		Compression method		File last modification time		File last modification date	
0x0010	CRC-32				Compressed size				Uncompressed size				File name length		Extra field length	
0x0020	File comment length		Disk number where file starts		Internal file attributes		External file attributes				Relative offset of local file header. This is the number of bytes between the start of the first disk on which the file occurs, and the start of the local file header				Filename	
0x0030	filename continue								Extra field							
0x0040	File Comment															

The first one will be this:

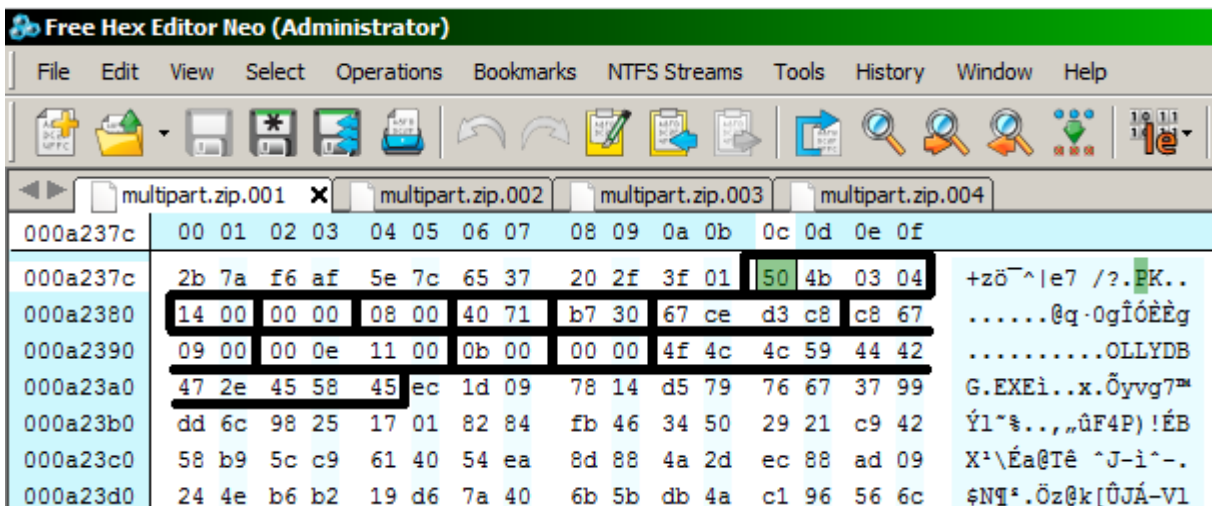


- **signature:** 50 4b 01 02
- **version made by:** 14 00
- **Version need to extract:** 14 00
- **General purpose bit flag:** 00 00
- **compression method:** 08 00
- **file last modification time:** E9 10
- **file last modification date:** 9E 31
- **CRC-32 checksum:** 9C E9 7B CC
- **compressed size:** F8 75 00 00
- **uncompressed size:** 00 F0 00 00
- **file name length:** 06 00
- **extra field length:** 00 00
- **file comment length:** 00 00
- **disk number where it starts:** 00 00
- **internal file attributes: lowest bit:** 00 00
- **external file attributes:** 20 00 00 00
- **Relative offset:** 00 00 00 00
- **Filename:** 6E 63 2E 65 78 65
- **Extra field:** EMPTY
- **File comment:** EMPTY

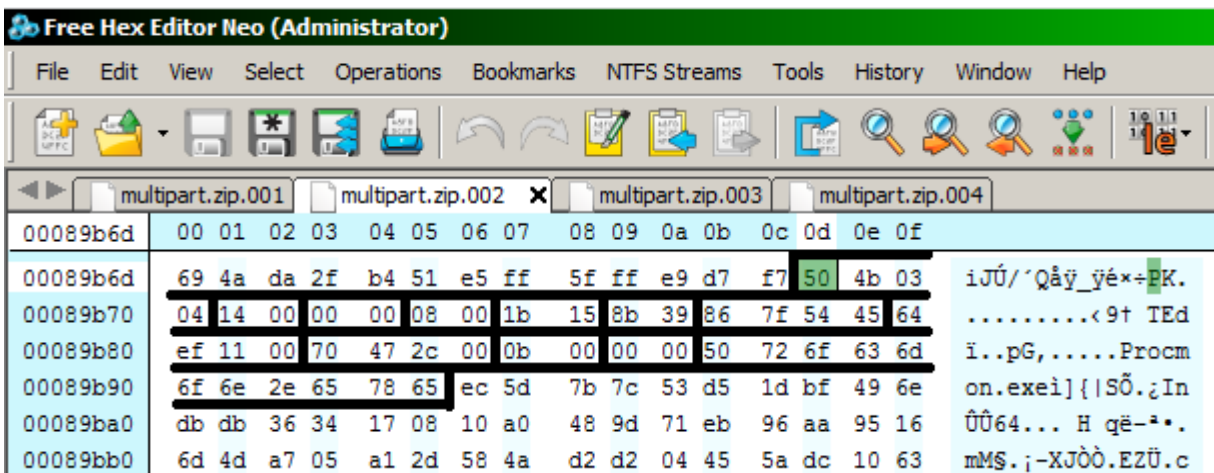
The second entry will be:



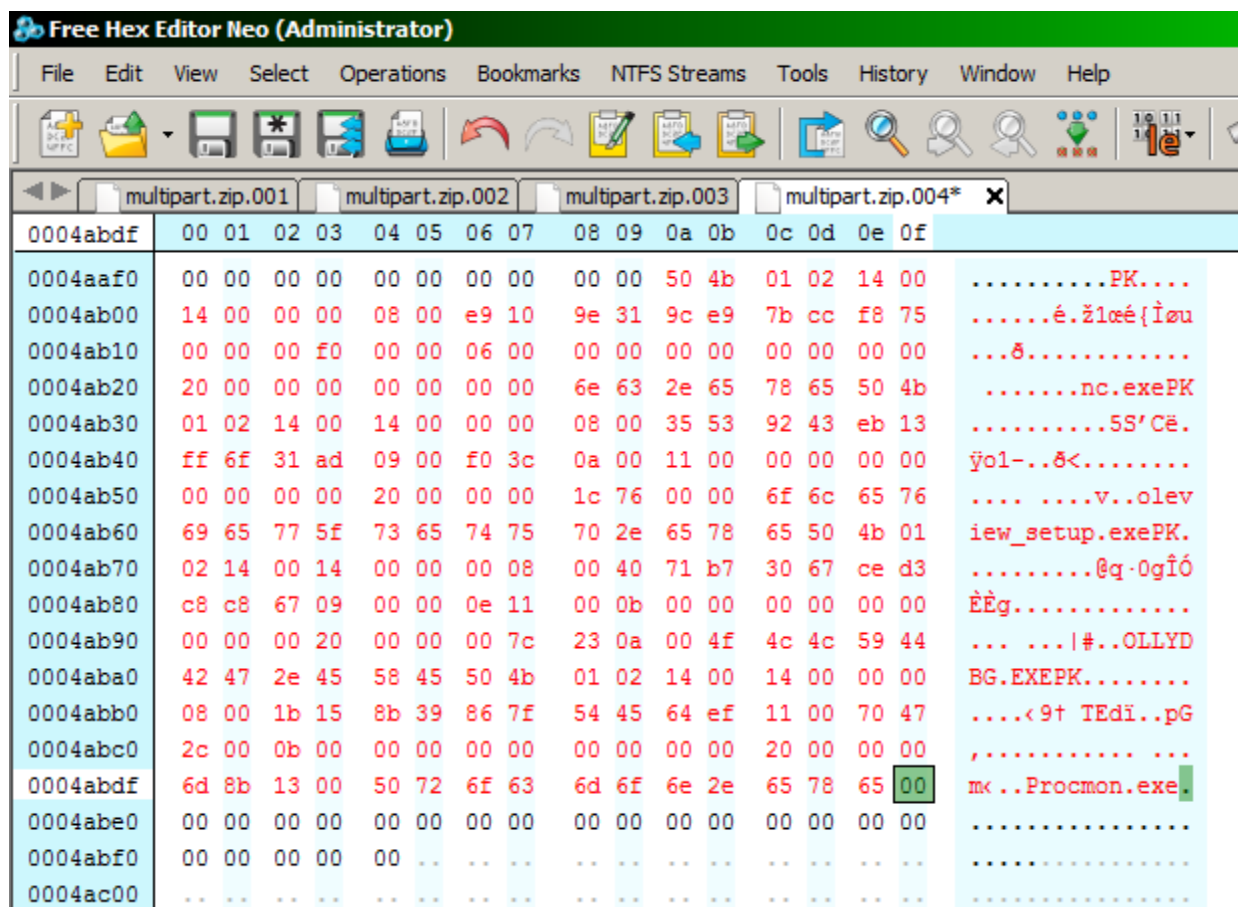
- **signature:** 50 4b 01 02
- **version made by:** 14 00
- **Version need to extract:** 14 00
- **General purpose bit flag:** 00 00
- **compression method:** 08 00
- **file last modification time:** 35 53
- **file last modification date:** 92 43
- **CRC-32 checksum:** EB 13 FF 6F
- **compressed size:** 31 AD 09 00
- **uncompressed size:** F0 3C 0A 00
- **file name length:** 11 00
- **extra field length:** 00 00
- **file comment length:** 00 00
- **disk number where it starts:** 00 00
- **internal file attributes: lowest bit:** 00 00
- **external file attributes:** 20 00 00 00
- **Relative offset:** 1C 76 00 00
- **Filename:** 6f 6C 65 76 69 65 77 5F 73 65 74 75 70 2E 65 78 65
- **Extra field:** EMPTY
- **File comment:** EMPTY



- **signature:** 50 4b 01 02
- **version made by:** 14 00
- **Version need to extract:** 14 00
- **General purpose bit flag:** 00 00
- **compression method:** 08 00
- **file last modification time:** 40 71
- **file last modification date:** B7 30
- **CRC-32 checksum:** 67 CE D3 C8
- **compressed size:** C8 67 09 00
- **uncompressed size:** 00 0E 11 00
- **file name length:** 0B 00
- **extra field length:** 00 00
- **file comment length:** 00 00
- **disk number where it starts:** 00 00
- **internal file attributes: lowest bit:** 00 00
- **external file attributes:** 20 00 00 00
- **Relative offset:** 7C 23 0A 00
- **Filename:** 4f 4C 4C 59 44 42 47 2E 45 58 45
- **Extra field:** EMPTY
- **File comment:** EMPTY



- **signature:** 50 4b 01 02
- **version made by:** 14 00
- **Version need to extract:** 14 00
- **General purpose bit flag:** 00 00
- **compression method:** 08 00
- **file last modification time:** 1B 15
- **file last modification date:** 8B 39
- **CRC-32 checksum:** 86 7F 54 45
- **compressed size:** 64 EF 11 00
- **uncompressed size:** 70 47 2C 00
- **file name length:** 0B 00
- **extra field length:** 00 00
- **file comment length:** 00 00
- **disk number where it starts:** 00 00
- **internal file attributes: lowest bit:** 00 00
- **external file attributes:** 20 00 00 00
- **Relative offset:** 6D 9B 08 00
- **Filename:** 50 72 6F 63 6D 6F 6E 2E 65 78 65
- **Extra field:** EMPTY
- **File comment:** EMPTY



As we can see the end of central directory structure is missing, so generate it.

End of central directory structure

As one can see there are some bytes not filled yet after the four central directory entry. It is the "End of central directory" entry, what has the following structure:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
0x0000	End of central directory signature = 0x06054b50				Number of this disk		Disk where central directory starts		Number of central directory records on this disk		Total number of central directory records		Size of central directory (bytes)					
0x0010	Offset of start of central directory, relative to start of archive				Comment length		comment											

Reconstruction of "End of central directory" entry

Based on this information we can generate the "end of central directory" entry:

- **signature:** 50 4B 05 06
- **number of disks:** 00 00
- **disk where the central directory starts:** the central directory is on the last disk: 00 00
- **number of central directory records on this disk:** 04 00
- **total number of central directory records:** 04 00
- **size of central directory:** E5 00 00 00
- **offset of start of the central directory:** FA 7A 25 00 (recognize, it is measured from the beginning of the first file, not from the beginning of this file, like it were only one huge zip file)
- **comment length:** 00 00
- **comment:** EMPTY

The reconstructed central directory, and end of central directory looks like as:

Offset	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	ASCII
0004abf5																	
0004aaf0	00	00	00	00	00	00	00	00	00	00	50	4b	01	02	14	00PK....
0004ab00	14	00	00	00	08	00	e9	10	9e	31	9c	e9	7b	cc	f8	75é.žlœé{ïœu
0004ab10	00	00	00	f0	00	00	06	00	00	00	00	00	00	00	00	00	...đ.....
0004ab20	20	00	00	00	00	00	00	00	6e	63	2e	65	78	65	50	4bnc.exePK
0004ab30	01	02	14	00	14	00	00	00	08	00	35	53	92	43	eb	135S' Cē.
0004ab40	ff	6f	31	ad	09	00	f0	3c	0a	00	11	00	00	00	00	00	ÿo1-..đ<.....
0004ab50	00	00	00	00	20	00	00	00	1c	76	00	00	6f	6c	65	76v..olev
0004ab60	69	65	77	5f	73	65	74	75	70	2e	65	78	65	50	4b	01	iew_setup.exePK.
0004ab70	02	14	00	14	00	00	00	08	00	40	71	b7	30	67	ce	d3@q·0gîÓ
0004ab80	c8	c8	67	09	00	00	0e	11	00	0b	00	00	00	00	00	00	ÈÈg.....
0004ab90	00	00	00	20	00	00	00	7c	23	0a	00	4f	4c	4c	59	44 #..OLLYD
0004aba0	42	47	2e	45	58	45	50	4b	01	02	14	00	14	00	00	00	BG.EXEPK.....
0004abb0	08	00	1b	15	8b	39	86	7f	54	45	64	ef	11	00	70	47<9† TEđi..pG
0004abc0	2c	00	0b	00	00	00	00	00	00	00	00	00	20	00	00	00	,..... ..
0004abd0	6d	8b	13	00	50	72	6f	63	6d	6f	6e	2e	65	78	65	50	m< ..Procmon.exeP
0004abe0	4b	05	06	00	00	00	00	04	00	04	00	e5	00	00	00	fa	K.....â...ú
0004abf5	7a	25	00	00	00	zš... ..
0004ac00

We can extract the files. The last one will be injured because some part of it is overwritten too, but the other three are saved.

7-Zip: Diagnostic messages



#	Message
0	C:\t\multipart.zip.001
1	CRC failed in 'Procmon.exe'. File is broken.

Close