

Lance Mueller Practical #3

Step 1: Study and understand the JFIF file header format.....	2
Step 2: File Header & Footer Location.....	2
Step 3: File Header Analysis.....	3
1 st Marker SOI – FF D8	3
2 nd Marker – FF E0	3
Offset = 2	3
Offset = 4	3
Offset = 6	3
Offset = 11	4
Offset = 13	4
Offset = 14	4
Offset = 16	5
Offset = 18	5
3 rd Marker – FF E2.....	5
Step 4 – Finding the remainder of the markers.....	6
RESULTS	7
Step 5: Putting the Header and Marker Puzzle Together.....	8
Markers	8
Subsequent Markers.....	8
Summary so Far	9
Step 6: Byte Stuffing Results	9
Step 7: Putting it all Together!.....	10
Initial Image Attempt.....	10

Step 1: Study and understand the JFIF file header format

Gain a comprehensive understanding of its structure. This was achieved by reading up on the following two websites.

1. http://en.wikipedia.org/wiki/JPEG#Syntax_and_structure
2. <http://www.obrador.com/essentialjpeg/headerinfo.htm>

Once an understanding of the makeup of a JPEG file was achieved we then started to analyze the data.

Step 2: File Header & Footer Location

We searched for and located the **File Header** with the GREP expression `\xFF\xD8\xff[\xE0\xE1\xE2\xE3\xDB\xE8\xFE]`

File Header

	Name	Bookmark Sector	Preview
<input checked="" type="checkbox"/> 1	Unallocated Clusters	427,586	ÿØÿà JFIF ÿà) IC

We searched for and located the **File Footer** with the GREP expression `\xFF\xD9`

File Footer

	Name	Bookmark Sector	Preview
<input checked="" type="checkbox"/> 1	Unallocated Clusters	427,202	X,i# \$Q U_Â ' :tçŠ ; E PQÿÙ

Note: File Footer appears before the File Header in the sector count.

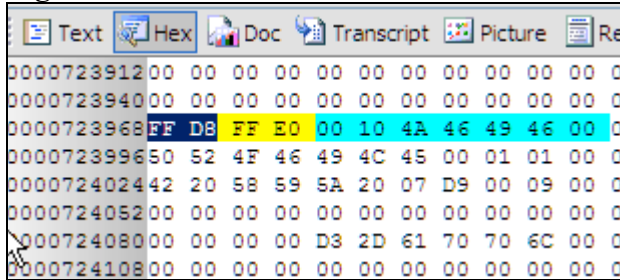
We now need analyze the File Header and subsequent markers to extract certain values to assist us in understanding the size and configuration of the file.

Step 3: File Header Analysis

1st Marker SOI – FF D8

Offset = 0 – Length = 2 Bytes (FF D8)

Fig 1



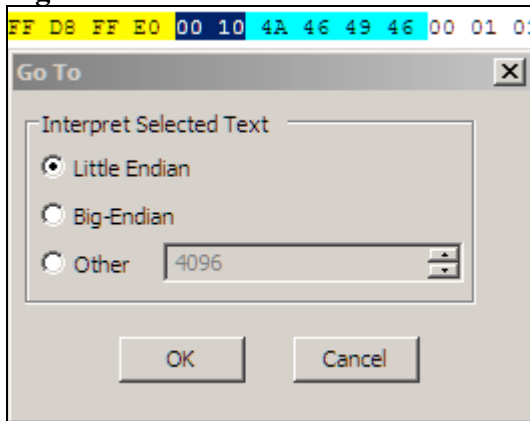
2nd Marker – FF E0

The 2nd marker has additional information following it. Because there is additional information following the 2nd marker this is referred to as a header marker. We need to decode the data following the 2nd Marker.

Offset = 2 – Length = 2 Bytes (FF E0)

Offset = 4 – Length = 2 Bytes (00 10) File Size - In this case the file is taking up 4096 Bytes. This is 8 sectors if our FS is 512 Bytes/Sector. See **fig 2** below

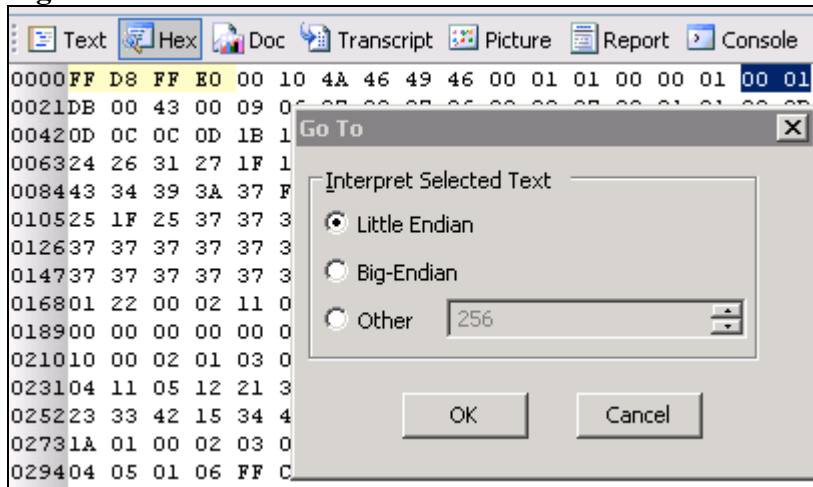
Fig 2



Offset = 6 – Length = Five Bytes 4A, 46, 49, 46, 00 (the ASCII code equivalent of a zero terminated "JFIF" string)

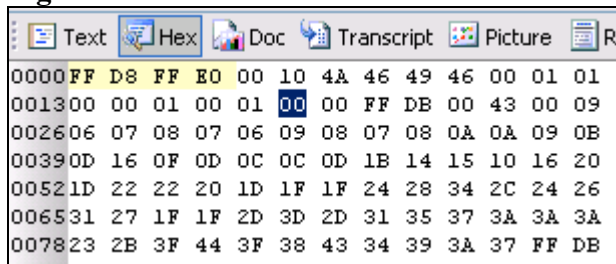
Offset = 16 – Length = 2 Bytes – Ydensity – See Fig 7 below

Fig 7



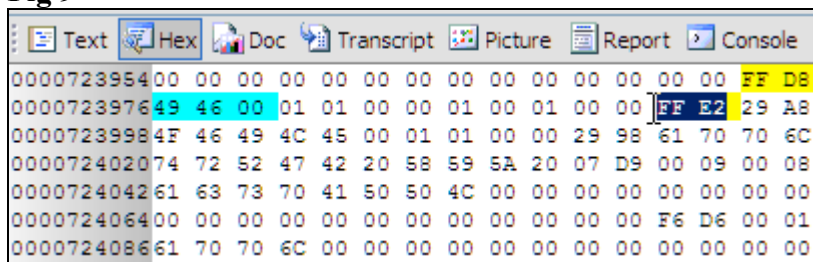
Offset = 18 – Length = 1 Byte - Xthumbnail -- one byte: 0 = no thumbnail See Fig 8 below.

Fig 8



3rd Marker – FF E2

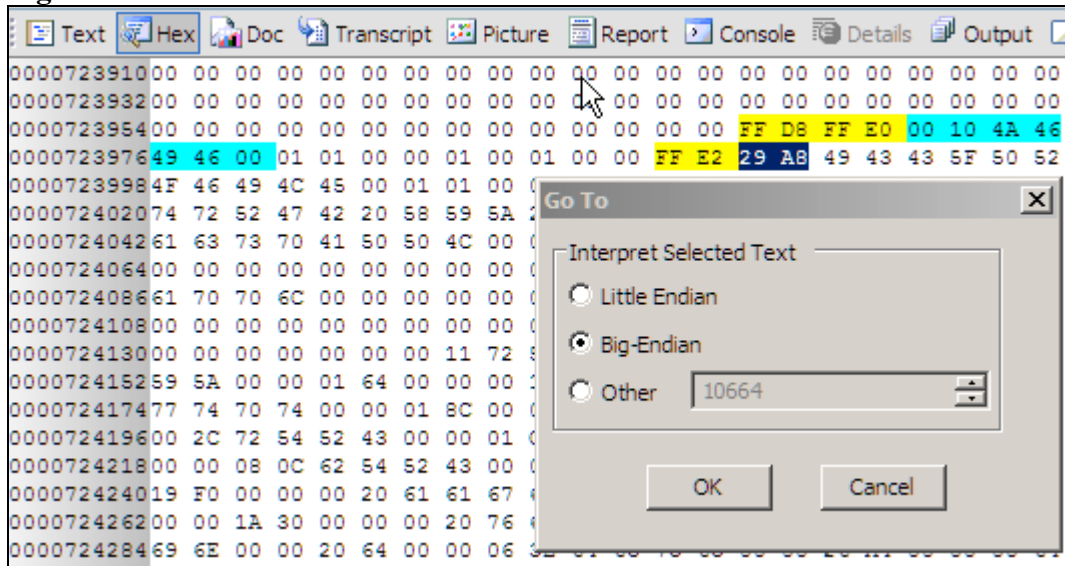
Fig 9



APPn	0xFFEn	<i>variable size</i>	Application-specific	For example, an Exif JPEG file uses an APP1 marker to store metadata, laid out in a structure based closely on TIFF .
-------------	--------	----------------------	----------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------

I can not find any useful information on this marker FF E2. Using the same format for its size I used the following 2 Bytes to gain its size. See below **Fig 10**.

Fig 10



Using the above methodology for size it reveals that this stream is 10664 Bytes in length!

Using this size we run out of data in this sector, so we need to find the sector with the next marker.

Step 4 – Finding the remainder of the markers

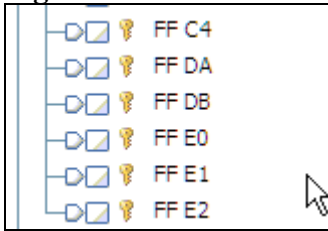
I reviewed the rest of the markers that could possibly be found (Wikipedia) and created GREP expressions for these. See below Fig 11. I already have the Header and Footer

Fig 11

	Name	
<input checked="" type="checkbox"/>	1	FF C2 \xFF\xC2
<input checked="" type="checkbox"/>	2	FF C4 \xFF\xC4
<input checked="" type="checkbox"/>	3	FF DA \xFF\xDA
<input checked="" type="checkbox"/>	4	FF DB \xFF\xDB
<input checked="" type="checkbox"/>	5	FF DD \xFF\xDD
<input checked="" type="checkbox"/>	6	FF E0 \xFF\xE0
<input checked="" type="checkbox"/>	7	FF E1 \xFF\xE1
<input checked="" type="checkbox"/>	8	FF E2 \xFF\xE2

After running the search I received 10 hits in the below expressions. See below in Fig 12.

Fig 12 - Hits



RESULTS

FF E0 (Header)

Table Report Gallery Timeline Disk Code			
	Name	Bookmark Sector	Preview
<input checked="" type="checkbox"/>	1 Unallocated Clusters	427,586	ÿØÿà JFIF ÿà) ICC

FF DB

Table Report Gallery Timeline Disk Code			
	Name	Bookmark Sector	Preview
<input checked="" type="checkbox"/>	1 Unallocated Clusters	426,442	s s ÿÛ C
<input checked="" type="checkbox"/>	2 Unallocated Clusters	426,442	ÿÛ C

FF C4

Table Report Gallery Timeline Disk Code			
	Name	Bookmark Sector	Preview
<input checked="" type="checkbox"/>	1 Unallocated Clusters	426,442	ÿÀ s s " ÿÀ
<input checked="" type="checkbox"/>	2 Unallocated Clusters	426,442	ÿÀ µ }
<input checked="" type="checkbox"/>	3 Unallocated Clusters	426,442	ØÛáääääâæçèéêëñóóóóó+óúÿÀ
<input checked="" type="checkbox"/>	4 Unallocated Clusters	426,442	ÿÀ µ w

FF DA

Table Report Gallery Timeline Disk Code			
	Name	Bookmark Sector	Preview
<input checked="" type="checkbox"/>	1 Unallocated Clusters	426,443	Ö×ØÛáääääâæçèéêëóóóóó+óúÿÛ ? ÿ øñ/<>t

FF E1

	Name	Bookmark Sector	Preview
<input type="checkbox"/> 1	Unallocated Clusters	428,686	ight Apple, Inc., 2009 yã eExif MM * 1

FF E2

	Name	Bookmark Sector	Preview
<input type="checkbox"/> 1	Unallocated Clusters	427,586	yÏyà JFIF yà ICC_PROFILE)~app

FF D9 - Footer

	Name	Bookmark Sector	File Offset	Preview
<input type="checkbox"/> 1	Unallocated Clusters	427,202	527396	X,i# \$Q U_Â 'tçŠ ; E PQyU

At this point we have possibly found all the markers. Now we have to put them together to complete the first part of this puzzle.

Step 5: Putting the Header and Marker Puzzle Together

Markers

Marker 1: FF D8 – 1st Part of Header

Marker 2: FF E0 – 2nd Part of Header

Subsequent Markers...

***Marker 3: FF E2 - Application-specific**

Located in sector 427586 (same as FF D8 FF E0). Could not locate any info on this marker so I think we will remove it and the data run.

Marker 4: FF DB - Define Quantization Table(s)

Located in sector 426442

Marker 5: FF C4 - Define Huffman Table(s)

Located in sector 426442

Marker 6: FF DA – SOS – Start of Scan

Located in sector 426443

***Marker 7: FF E1 - Application-specific**

Located in sector 424686

Marker 8: FF D9 - Footer

Located in sector 427202 for a LE of 38 bytes

* **Note on FF E1 & FF E2 - Application-specific** - For example, an Exif JPEG file uses an APP1 marker to store metadata, laid out in a structure based closely on TIFF. **I'm going to remove Marker 2 and 6.** We don't need them and this will clean things up and make it easier when putting the sectors together. So we are left with...

1. FF D8 FF E0
 2. FF DB
 3. FF C4
 4. FF DA – Start of Image
 5. FF D9 – Image End
- Theoretically you could remove all the markers up to **FF DB (Define Quantization Table(s))** and just go with the header, **FF DB FF E0**, and all the remaining data until the Footer **FF D9**. Let's try that and see if it works.
 - a. **I tried this on a TEST IMAGE. I exported the header up to (and not including) the first marker FF E2, the last byte before FF E2. Then I went to the next marker FF DB (Define Quantization Table(s)) and exported all the data (including the marker) all the way to the Footer FF D9 (EOF, end of file). The image exported perfectly in EnCase.**
 - Next I'll start looking for byte stuffing FF 00 for sectors that may hold image data.

Summary so Far

We have found the header and the markers after the header. Now we need to place the headers in order and find all the other sectors that have the actual data that comprises the image. We need to figure out where the sectors are and then in what order they go.

Step 6: Byte Stuffing Results

A search for FF 00 yielded 71 hits in UC. We will review each hit and remove any duplicates.

I removed all the initial hits that resided in the same sectors and then removed any additional duplicates or false positives when reviewed a second time. This gave me 12 remaining sectors with FF 00.

Next I then looked for the invalid sectors where byte stuffing had occurred. This is where after a FF 00 there was a 00. I found 3 such sectors and removed them from my analysis. Below are the remaining sectors.

	Name	Bookmark Sector	File Offset
<input type="checkbox"/>	Unallocated Clusters	426,172	40
<input type="checkbox"/>	Unallocated Clusters	426,188	8289
<input type="checkbox"/>	Unallocated Clusters	426,208	18459
<input type="checkbox"/>	Unallocated Clusters	426,228	28701
<input type="checkbox"/>	Unallocated Clusters	426,246	38195
<input type="checkbox"/>	Unallocated Clusters	426,718	279637
<input type="checkbox"/>	Unallocated Clusters	426,858	351316
<input type="checkbox"/>	Unallocated Clusters	426,879	362442
<input type="checkbox"/>	Unallocated Clusters	428,457	1169933

7 of the data runs covered 512 byte counts
 2 of the data runs covered 1024 byte counts

Step 7: Putting it all Together!

We have now exported the header **1) FF D8 FF E0**, and markers **2) FF DB**, **3) FF C4** and **4) FF DA**. The Footer **5) FF D9** will be last after we export all the data runs.

I then exported the sectors by bookmark sector order just to see if all of the above work would actually yield anything that resembled a JPEG. To my delight I got something. See below.

Initial Image Attempt



As we can see we have some bytes out of order. We will play around to see if we can clean this up.