



YATES` PE NOTES
=====

- 1.....Header Details
- 2.....Section Details
- 3.....Full PEHeader listing
- 4.....Import details
- 5.....Export Details
- 6.....Reloc Details

01/FEB/04

;------ .COMMON.HEADER.-----

@ 0x3c a DD which states the file offset of the PE Header,
this should point to 'PE'

0x06 bytes into the PE is the amount of sections, 1 word
0x28 bytes into the PE is the entry point (- imagebase) "OEP"
0x34 bytes into the PE is the imagebase
0x50 bytes into the PE is the size of image
0xf8 bytes into the PE section names start

;------ .COMMON.SECTION.-----

Start of section
=====

0x00 is the section name
0x08 is the section virtual size
0x0C is the section virtual address
0x10 is the section raw size
0x14 is the section raw offset
0x24 is the section characteristics
0x28 new section starts



;------.DETAILED.HEADER.-----;

All detailed offsets
=====

starting @ PE

the rest of the pe structure..

example info taken from rpp.exe, packed with UPX. first three numbers are offsets from 'PE'

```

000 pe signature          50450000      ; PE,0,0
    image_file_header {
004 MACHINE              4C01          ; 014c...intel 386...
006 NumberOfSections     0300          ; 3 sections, upx0,upx1,upx2
008 TimeDateStamp        A626967C      ; allegedly Thursday 27th march 2036
00c PointerToSymbolTable 00000000
010 NumberOfSymbols      00000000
014 SizeOfOptionalHeader E000          ; 00e0
016 Characteristics      8E81          ; yah, executable image, bytes reversed lo &
hi, 32bit machine.....
    }
_IMAGE_OPTIONAL_HEADER {
018 Magic                0B01
01a MajorLinkerVersion   02
01b MinorLinkerVersion   19            ; linker version 2.25 (decimal...)
01c SizeOfCode           00060000      ; code size 00000600
020 SizeOfInitializedData 00CC0000      ; initialized data size 0000cc00
024 SizeOfUninitializedData 00000000
028 AddressOfEntryPoint  37200100      ; entry point 00012037
02c BaseOfCode           00100000      ; base of code 00001000 (relative offset in
memory)
030 BaseOfData           00200000      ; base of data 00002000  "
"
034 ImageBase            00004000      ; image base 00400000 (preffered base address
to map the image to)
038 SectionAlignment     00100000      ; 00001000, section alignment in memory...
03c FileAlignment        00020000      ; 00000200, section alignment in file, zero
padded...
040 MajorOperatingSystemVersion 0100
042 MinorOperatingSystemVersion 0000      ; os version 1.0
044 MajorImageVersion    0000
046 MinorImageVersion    0000
048 MajorSubsystemVersion 0300
04a MinorSubsystemVersion 0A00          ; 3.10 decimal, for windoze NT v3.10
04c Reserved1            00000000
050 SizeOfImage          2E2B0100      ; 00012b2e, how much address space to reserve
in the address space for the loaded executable image
054 SizeOfHeaders        00040000      ; 00000400, how much space is used by all

```



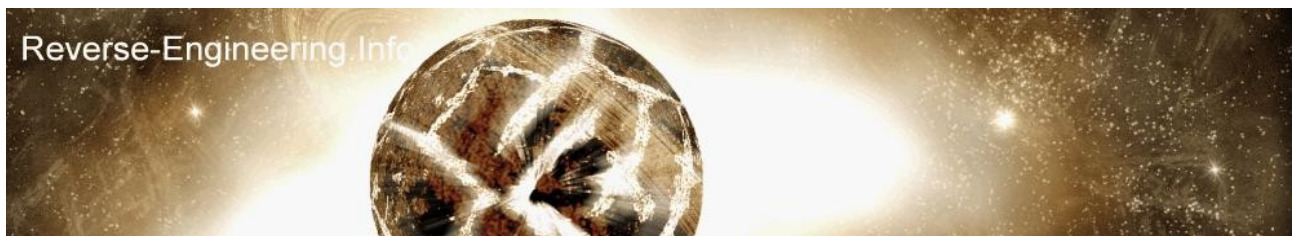
```

file headers, ms-dos header, pe-file header, pe-optional header, pe-section headers
058 CheckSum          00000000
05c Subsystem        0200          ; 0002, windows gui...
05e DllCharacteristics 0000
060 SizeOfStackReserve 00001000
064 SizeOfStackCommit 00200000
068 SizeOfHeapReserve 00001000
06c SizeOfHeapCommit  00100000
070 LoaderFlags      00000000
074 NumberOfRvaAndSizes 10000000      ; 0x00000010, this field identifies the
length of the datadirectory array
    IMAGE_DATA_DIRECTORY DataDirectory[
078 IMAGE_DIRECTORY_ENTRY_EXPORT 00000000 00000000      ; 11 entrys being used, out
of a possible 16...
080 IMAGE_DIRECTORY_ENTRY_IMPORT 00100100 90000000
088 IMAGE_DIRECTORY_ENTRY_RESOURCE 00000000 00000000
090 IMAGE_DIRECTORY_ENTRY_EXCEPTION 00000000 00000000
098 IMAGE_DIRECTORY_ENTRY_SECURITY 00000000 00000000
0a0 IMAGE_DIRECTORY_ENTRY_BASERELOC 00000000 00000000
0a8 IMAGE_DIRECTORY_ENTRY_DEBUG 00000000 00000000
0b0 IMAGE_DIRECTORY_ENTRY_COPYRIGHT 00000000 00000000
0b8 IMAGE_DIRECTORY_ENTRY_GLOBALPTR 00000000 00000000
0c0 IMAGE_DIRECTORY_ENTRY_TLS 00000000 00000000
0c8 IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG 00000000 00000000
0d0
0d8
0e0
0e8
0f0
    ]
} end of optional header...

section_header{
Name[IMAGE_SIZEOF_SHORT_NAME] 55505830 00000000      ; 'UPX0',0,0,0,0
PhysicalAddress / VirtualSize 00000100      ; 00010000, apprantly unused
VirtualAddress 00100000      ; 00001000, virtual address to load
this section into
SizeOfRawData 00000000
PointerToRawData 00040000      ; 00000400, pointer to the data in
the file
PointerToRelocations 00000000
PointerToLinenumbers 00000000
NumberOfRelocations 0000
NumberOfLinenumbers 0000
Characteristics 400000E0      ; e0000040, initialized data, section
is executable/readable/writable
} end of section header...

```

Import Table Info



The import directory is an array of `IMAGE_IMPORT_DESCRIPTORs` (iids) 0x80 of the PE there is an rva which points to the image directory

there is an iid for each dll used

The list is terminated by a `IMAGE_IMPORT_DESCRIPTOR` that's entirely filled with 0-bytes.

each iid is five dwords

(iid) `IMAGE_IMPORT_DESCRIPTOR` structure..

`OriginalFirstThunk, TimeDateStamp, ForwarderChain, Name, FirstThunk`

`OriginalFirstThunk`

An RVA (32 bit) pointing to a 0-terminated array of RVAs to `IMAGE_THUNK_DATAs`, each describing one imported function. The array will never change.

`TimeDateStamp`

A 32-bit-timestamp that has several purposes. Let's pretend that the timestamp is 0, and handle the advanced cases later.

`ForwarderChain`

The 32-bit-index of the first forwarder in the list of imported functions. Forwarders are also advanced stuff; set to all-bits-1 for beginners.

`Name`

A 32-bit-RVA to the name (a 0-terminated ASCII string) of the DLL.

`FirstThunk`

An RVA (32 bit) to a 0-terminated array of RVAs to `IMAGE_THUNK_DATAs`, each describing one imported function. The array is part of the import address table and will change.

0x00 OFT
0x04 Time
0x08 FC
0x0c lib
0x10 FT
0x14 next iid



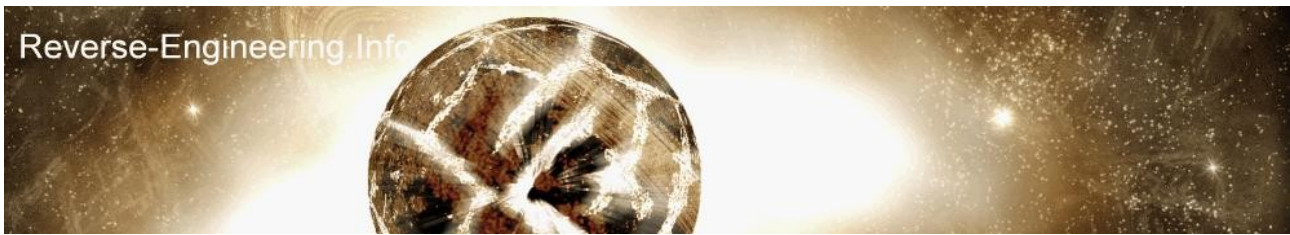
Export Table Info

+0C Dll Name Ptr
+10 Ordinal Base
+14 Number of functions
+18 Number of names
+1C Address of functions
+20 Address of names
+24 Address of Ordinals

EXPORT TABLE LINKAGE

FUNC (dwords)	INDEX (words)	NAME (dwords)
FUNC1 [77E8F321]	ORD1 (04) <->	NAMEptr1
FUNC2 [77E8E319] <-	ORD2 (54) <->	NAMEptr2
FUNC3 [77E8A329] \	ORD3 (01) <->	NAMEptr3
FUNC4 [77E72DAB] \->	ORD4 (02) <->	NAMEptr4->MessageBoxA
FUNC5 [77E8A311] -->	ORD5 (07) <->	NAMEptr5
FUNC7 [77E45A28] /	ORD6 (23) <->	NAMEptr6
FUNC8 [77E45A28] <-	ORD7 (03) <->	NAMEptr7->GetOpenFileName
FUNC9 [77E45125]	ORD8 (21) <->	NAMEptr8

77E8E319 = MessageBoxA
77E45125 = 8th func, no ord 08, 00000008 + ordbase



```
RELOC
```

```
-----
```

```
relocation records
```

```
DD VA_BASE, DD SIZE, DW, DW, DW, DW, DW, DW, ... 0000 term  
DD VA_BASE, DD SIZE, DW, DW, DW, DW, DW, DW, ... 0000 term  
DD VA_BASE, DD SIZE, DW, DW, DW, DW, DW, DW, ... 0000 term  
00000
```

the words are the entries and have a format of,

```
1111,111111111111  
TYPE  OFFSET
```

TYPE - 3, subtract pe,imagebase and add current base

example

```
DD 00100000, DD xxxxxxxx, 0030, 1630, 2830, 4030, 4C30, 5C30
```

e.g imagebase 400000

Base = 00401000

```
first record = 3000 | TYPE = 3000 >> 0x0C = 3 | offset = (3000 << 4) (3000 >> 4) = 0  
401000+0 = 401000 reloc dd @ [401000]  
second record = 3016 | TYPE = 3016 >> 0x0C = 3 | offset = (3016 << 4) (3016 >> 4) = 16  
401000+16 = 401016 reloc dd @ [401016]  
.. etc
```

```
-----
```

```
--
```