# Portable Executable Format Layout

```
struct _IMAGE_DOS_HEADER {
0x00  WORD e_magic;
0x02  WORD e_cblp;
0x04  WORD e_cp;
0x06  WORD e_crlc;
0x08  WORD e_cparhdr;
0x0a  WORD e_minalloc;
0x0c  WORD e_maxalloc;
0x0e  WORD e_ss;
0x10  WORD e_sp;
0x12  WORD e_csum;
0x14  WORD e_ip;
0x16  WORD e_cs;
0x18  WORD e_lfarlc;
0x1a  WORD e_ovno;
0x1c  WORD e_res[4];
0x24  WORD e_oemid;
0x26  WORD e_oeminfo;
0x28  WORD e_res2[10];
0x3c  DWORD e_lfanew;
};
```

```
struct _IMAGE_NT_HEADERS {
0x00  DWORD Signature;
0x04  _IMAGE_FILE_HEADER FileHeader;
0x18  _IMAGE_OPTIONAL_HEADER OptionalHeader;
};
```

```
struct _IMAGE_FILE_HEADER {
0x00  WORD Machine;
0x02  WORD NumberOfSections;
0x04  DWORD TimeDateStamp;
0x08  DWORD PointerToSymbolTable;
0x0c  DWORD NumberOfSymbols;
0x10  WORD SizeOfOptionalHeader;
0x12  WORD Characteristics;
};
```

```
struct _IMAGE_OPTIONAL_HEADER {
0x00  WORD Magic;
0x02  BYTE MajorLinkerVersion;
0x03  BYTE MinorLinkerVersion;
0x04  DWORD SizeOfCode;
0x08  DWORD SizeOfInitializedData;
0x0c  DWORD SizeOfUninitializedData;
0x10  DWORD AddressOfEntryPoint;
0x14  DWORD BaseOfCode;
0x18  DWORD BaseOfData;
0x1c  DWORD ImageBase;
0x20  DWORD SectionAlignment;
0x24  DWORD FileAlignment;
0x28  WORD MajorOperatingSystemVersion;
0x2a  WORD MinorOperatingSystemVersion;
0x2c  WORD MajorImageVersion;
0x2e  WORD MinorImageVersion;
0x30  WORD MajorSubsystemVersion;
0x32  WORD MinorSubsystemVersion;
0x34  DWORD Win32VersionValue;
0x38  DWORD SizeOfImage;
0x3c  DWORD SizeOfHeaders;
0x40  DWORD CheckSum;
0x44  WORD Subsystem;
0x46  WORD DllCharacteristics;
0x48  DWORD SizeOfStackReserve;
0x4c  DWORD SizeOfStackCommit;
0x50  DWORD SizeOfHeapReserve;
0x54  DWORD SizeOfHeapCommit;
0x58  DWORD LoaderFlags;
0x5c  DWORD NumberOfRvaAndSizes;
0x60  _IMAGE_DATA_DIRECTORY DataDirectory[16];
};
```

PE32+

```
struct _IMAGE_OPTIONAL_HEADER {
0x00  WORD Magic;
0x02  BYTE MajorLinkerVersion;
0x03  BYTE MinorLinkerVersion;
0x04  DWORD SizeOfCode;
0x08  DWORD SizeOfInitializedData;
0x0c  DWORD SizeOfUninitializedData;
0x10  DWORD AddressOfEntryPoint;
0x14  DWORD BaseOfCode;;
0x18  QWORD ImageBase;
0x20  DWORD SectionAlignment;
0x24  DWORD FileAlignment;
0x28  WORD MajorOperatingSystemVersion;
0x2a  WORD MinorOperatingSystemVersion;
0x2c  WORD MajorImageVersion;
0x2e  WORD MinorImageVersion;
0x30  WORD MajorSubsystemVersion;
0x32  WORD MinorSubsystemVersion;
0x34  DWORD Win32VersionValue;
0x38  DWORD SizeOfImage;
0x3c  DWORD SizeOfHeaders;
0x40  DWORD CheckSum;
0x44  WORD Subsystem;
0x46  WORD DllCharacteristics;
0x48  QWORD SizeOfStackReserve;
0x4c  QWORD SizeOfStackCommit;
0x50  QWORD SizeOfHeapReserve;
0x54  QWORD SizeOfHeapCommit;
0x58  DWORD LoaderFlags;
0x5c  DWORD NumberOfRvaAndSizes;
0x60  _IMAGE_DATA_DIRECTORY DataDirectory[16];
};
```

_IMAGE_FILE_HEADER_NumberOfSections

```
typedef struct _IMAGE_SECTION_HEADER {
0x00  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
      union {
0x08          DWORD PhysicalAddress;
0x08          DWORD VirtualSize;
      } Misc;
0x0c  DWORD VirtualAddress;
0x10  DWORD SizeOfRawData;
0x14  DWORD PointerToRawData;
0x18  DWORD PointerToRelocations;
0x1c  DWORD PointerToLinenumbers;
0x20  WORD  NumberOfRelocations;
0x22  WORD  NumberOfLinenumbers;
0x24  DWORD Characteristics;
};
```

```
typedef struct _IMAGE_SECTION_HEADER {
0x00  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
      union {
0x08          DWORD PhysicalAddress;
0x08          DWORD VirtualSize;
      } Misc;
0x0c  DWORD VirtualAddress;
0x10  DWORD SizeOfRawData;
0x14  DWORD PointerToRawData;
0x18  DWORD PointerToRelocations;
0x1c  DWORD PointerToLinenumbers;
0x20  WORD  NumberOfRelocations;
0x22  WORD  NumberOfLinenumbers;
0x24  DWORD Characteristics;
};
```

```
typedef struct _IMAGE_SECTION_HEADER {
0x00  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
      union {
0x08          DWORD PhysicalAddress;
0x08          DWORD VirtualSize;
      } Misc;
0x0c  DWORD VirtualAddress;
0x10  DWORD SizeOfRawData;
0x14  DWORD PointerToRawData;
0x18  DWORD PointerToRelocations;
0x1c  DWORD PointerToLinenumbers;
0x20  WORD  NumberOfRelocations;
0x22  WORD  NumberOfLinenumbers;
0x24  DWORD Characteristics;
};
```

```
IMAGE_DIRECTORY_ENTRY_EXPORT

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_IMPORT

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_RESOURCE

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_EXCEPTION

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_SECURITY

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_BASERELOC

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_DEBUG

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_COPYRIGHT

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_GLOBALPTR

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_TLS

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_IAT

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR

struct _IMAGE_DATA_DIRECTORY {
0x00  DWORD VirtualAddress;
0x04  DWORD Size;
};
```

```
struct _IMAGE_IMPORT_DESCRIPTOR {
0x00  union {
            /* 0 for terminating null import descriptor  */
0x00        DWORD       Characteristics;
            /* RVA to original unbound IAT */
0x00        PIMAGE_THUNK_DATA OriginalFirstThunk;
      } u;
0x04  DWORD       TimeDateStamp;      /* 0 if not bound,
                                       * -1 if bound, and real date\time stamp
                                       *   in IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT
                                       * (new BIND)
                                       * otherwise date/time stamp of DLL bound to
                                       * (Old BIND)
                                       */
0x08  DWORD       ForwarderChain;     /* -1 if no forwarders */
0x0c  DWORD       Name;
            /* RVA to IAT (if bound this IAT has actual addresses) */
0x10  PIMAGE_THUNK_DATA FirstThunk;
};
```

```
struct _IMAGE_RESOURCE_DIRECTORY {
0x00  DWORD       Characteristics;
0x04  DWORD       TimeDateStamp;
0x08  WORD        MajorVersion;
0x0a  WORD        MinorVersion;
0x0c  WORD        NumberOfNamedEntries;
0x0e  WORD        NumberOfIdEntries;
};
```

```
struct _IMAGE_DEBUG_DIRECTORY {
0x00  DWORD Characteristics;
0x04  DWORD TimeDateStamp;
0x08  WORD  MajorVersion;
0x0a  WORD  MinorVersion;
0x0c  DWORD Type;
0x10  DWORD SizeOfData;
0x14  DWORD AddressOfRawData;
0x18  DWORD PointerToRawData;
};
```

```
struct _IMAGE_TLS_DIRECTORY {
0x00  DWORD       StartAddressOfRawData;
0x04  DWORD       EndAddressOfRawData;
0x08  DWORD       AddressOfIndex;
0x0c  DWORD       AddressOfCallBacks;
0x10  DWORD       SizeOfZeroFill;
0x14  DWORD       Characteristics;
};
```

PE32+

```
struct _IMAGE_TLS_DIRECTORY {
0x00  QWORD       StartAddressOfRawData;
0x04  QWORD       EndAddressOfRawData;
0x08  QWORD       AddressOfIndex;
0x0c  QWORD       AddressOfCallBacks;
0x10  DWORD       SizeOfZeroFill;
0x14  DWORD       Characteristics;
};
```

```
struct _IMAGE_DELAY_IMPORT_DESCRIPTOR {
0x00  DWORD grAttrs;
0x04  DWORD szName;
0x08  DWORD phmod;
0x0c  DWORD pIAT;
0x10  DWORD pINT;
0x14  DWORD pBoundIAT;
0x18  DWORD pUnloadIAT;
0x1c  DWORD dwTimeStamp;
};
```

```
struct _IMAGE_EXPORT_DIRECTORY {
0x00  DWORD       Characteristics;
0x04  DWORD       TimeDateStamp;
0x08  WORD        MajorVersion;
0x0a  WORD        MinorVersion;
0x0c  DWORD       Name;
0x10  DWORD       Base;
0x14  DWORD       NumberOfFunctions;
0x18  DWORD       NumberOfNames;
0x1c  DWORD       AddressOfFunctions;
0x20  DWORD       AddressOfNames;
0x24  DWORD       AddressOfNameOrdinals;
};
```

```
Indexed by Ordinals

address_of_function[0]
address_of_function[1]
address_of_function[2]
.
.
.
address_of_function[NumberOfFunctions]
```

Code/Data
Code/Data
Code/Data
Code/Data
Code/Data

```
Array of WORDs

name_ordinal[0]
name_ordinal[1]
name_ordinal[2]
.
.
.
name_ordinal[NumberOfNames]
```

```
Pointers to strings

address_of_name[0]
address_of_name[1]
address_of_name[2]
.
.
.
address_of_name[NumberOfNames]
```

If a symbol N is exported by ordinal and name then:
-Its name will be located at **AddressOfNames[N]**
-Its ordinal at **AddressOfNameOrdinals[N]**
-And its address* will be
**AddressOfFunctions[*AddressOfNameOrdinals[N]*]**

The function might be *forwarded*, in that case the last pointer will refer to an address within the exports pointing to the forwarder string, which will contain information on the symbol and the module where to find it.

```
typedef struct _IMAGE_THUNK_DATA {
      union {
0x00        LPBYTE    ForwarderString;
0x00        PDWORD    Function;
0x00        DWORD     Ordinal;
0x00        PIMAGE_IMPORT_BY_NAME      AddressOfData;
      } u1;
} IMAGE_THUNK_DATA,*PIMAGE_THUNK_DATA;
```

```
typedef struct _IMAGE_IMPORT_BY_NAME {
0x00  WORD          Hint;
0x02  BYTE  Name[1];
} IMAGE_IMPORT_BY_NAME,*PIMAGE_IMPORT_BY_NAME;
```

```
typedef struct _IMAGE_THUNK_DATA {
      union {
0x00        LPBYTE    ForwarderString;
0x00        PDWORD    Function;
0x00        DWORD     Ordinal;
0x00        PIMAGE_IMPORT_BY_NAME      AddressOfData;
      } u1;
} IMAGE_THUNK_DATA,*PIMAGE_THUNK_DATA;
```

```
typedef struct _IMAGE_IMPORT_BY_NAME {
0x00  WORD          Hint;
0x02  BYTE  Name[1];
} IMAGE_IMPORT_BY_NAME,*PIMAGE_IMPORT_BY_NAME;
```

```
typedef struct _IMAGE_THUNK_DATA {
      union {
0x00        LPBYTE    ForwarderString;
0x00        PDWORD    Function;
0x00        DWORD     Ordinal;
0x00        PIMAGE_IMPORT_BY_NAME      AddressOfData;
      } u1;
} IMAGE_THUNK_DATA,*PIMAGE_THUNK_DATA;
```

```
typedef struct _IMAGE_IMPORT_BY_NAME {
0x00  WORD          Hint;
0x02  BYTE  Name[1];
} IMAGE_IMPORT_BY_NAME,*PIMAGE_IMPORT_BY_NAME;
```

```
typedef struct _IMAGE_THUNK_DATA {
      union {
0x00        LPBYTE    ForwarderString;
0x00        PDWORD    Function;
0x00        DWORD     Ordinal;
0x00        PIMAGE_IMPORT_BY_NAME      AddressOfData;
      } u1;
} IMAGE_THUNK_DATA,*PIMAGE_THUNK_DATA;
```

```
typedef struct _IMAGE_IMPORT_BY_NAME {
0x00  WORD          Hint;
0x02  BYTE  Name[1];
} IMAGE_IMPORT_BY_NAME,*PIMAGE_IMPORT_BY_NAME;
```

## The PE32 and PE32+ Format

Display of the main headers describing the basic information in a Portable Executable file.

More details about the Portable Executable specification can be found at:

http://en.wikipedia.org/wiki/Portable_Executable

Structure contained within parent

Structure pointed to by the parent

© 2007 Ero Carrera
http://dkbza.org
http://blog.dkbza.org

OpenRCE.org