Portable Executable

```
#define IMAGE_DOS_SIGNATURE                 0x4D5A      // MZ

typedef struct _IMAGE_DOS_HEADER {      // DOS .EXE header
    WORD   e_magic;                     // Magic number
    WORD   e_cblp;                      // Bytes on last page of file
    WORD   e_cp;                        // Pages in file
    WORD   e_crlc;                      // Relocations
    WORD   e_cparhdr;                   // Size of header in paragraphs
    WORD   e_minalloc;                  // Minimum extra paragraphs needed
    WORD   e_maxalloc;                  // Maximum extra paragraphs needed
    WORD   e_ss;                        // Initial (relative) SS value
    WORD   e_sp;                        // Initial SP value
    WORD   e_csum;                      // Checksum
    WORD   e_ip;                        // Initial IP value
    WORD   e_cs;                        // Initial (relative) CS value
    WORD   e_lfarlc;                    // File address of relocation table
    WORD   e_ovno;                      // Overlay number
    WORD   e_res[4];                    // Reserved words
    WORD   e_oemid;                     // OEM identifier (for e_oeminfo)
    WORD   e_oeminfo;                   // OEM information; e_oemid specific
    WORD   e_res2[10];                  // Reserved words
    LONG   e_lfanew;                    // File address of new exe header
  } IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

```
#define IMAGE_OS2_SIGNATURE                 0x4E45      // NE
#define IMAGE_OS2_SIGNATURE_LE              0x4C45      // LE
#define IMAGE_NT_SIGNATURE                  0x50450000  // PE00

typedef struct _IMAGE_NT_HEADERS64 {
    DWORD Signature;
    IMAGE_FILE_HEADER FileHeader;
    IMAGE_OPTIONAL_HEADER64 OptionalHeader;
} IMAGE_NT_HEADERS64, *PIMAGE_NT_HEADERS64;

typedef struct _IMAGE_NT_HEADERS {
    DWORD Signature;
    IMAGE_FILE_HEADER FileHeader;
    IMAGE_OPTIONAL_HEADER32 OptionalHeader;
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;

typedef struct _IMAGE_ROM_HEADERS {
    IMAGE_FILE_HEADER FileHeader;
    IMAGE_ROM_OPTIONAL_HEADER OptionalHeader;
} IMAGE_ROM_HEADERS, *PIMAGE_ROM_HEADERS;
```

```
//
// File header format
//
typedef struct _IMAGE_FILE_HEADER {
    WORD    Machine;
    WORD    NumberOfSections;
    DWORD   TimeDateStamp;
    DWORD   PointerToSymbolTable;
    DWORD   NumberOfSymbols;
    WORD    SizeOfOptionalHeader;
    WORD    Characteristics;
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;

#define IMAGE_SIZEOF_FILE_HEADER            20

#define IMAGE_FILE_RELOCS_STRIPPED          0x0001  // Relocation info stripped from file.
#define IMAGE_FILE_EXECUTABLE_IMAGE         0x0002  // File is executable  (i.e. no unresolved external references).
#define IMAGE_FILE_LINE_NUMS_STRIPPED       0x0004  // Line numbers stripped from file.
#define IMAGE_FILE_LOCAL_SYMS_STRIPPED      0x0008  // Local symbols stripped from file.
#define IMAGE_FILE_AGGRESIVE_WS_TRIM        0x0010  // Agressively trim working set
#define IMAGE_FILE_LARGE_ADDRESS_AWARE      0x0020  // App can handle >2gb addresses
#define IMAGE_FILE_BYTES_REVERSED_LO        0x0080  // Bytes of machine word are reversed.
#define IMAGE_FILE_32BIT_MACHINE            0x0100  // 32 bit word machine.
#define IMAGE_FILE_DEBUG_STRIPPED           0x0200  // Debugging info stripped from file in .DBG file
#define IMAGE_FILE_REMOVABLE_RUN_FROM_SWAP  0x0400  // If Image is on removable media, copy and run from the swap file.
#define IMAGE_FILE_NET_RUN_FROM_SWAP        0x0800  // If Image is on Net, copy and run from the swap file.
#define IMAGE_FILE_SYSTEM                   0x1000  // System File.
#define IMAGE_FILE_DLL                      0x2000  // File is a DLL.
#define IMAGE_FILE_UP_SYSTEM_ONLY           0x4000  // File should only be run on a UP machine
#define IMAGE_FILE_BYTES_REVERSED_HI        0x8000  // Bytes of machine word are reversed.

#define IMAGE_FILE_MACHINE_I386             0x014c  // Intel 386.
#define IMAGE_FILE_MACHINE_IA64             0x0200  // Intel 64
#define IMAGE_FILE_MACHINE_AMD64            0x8664  // AMD64 (K8)
```

```
#define IMAGE_NT_OPTIONAL_HDR32_MAGIC       0x10b
#define IMAGE_NT_OPTIONAL_HDR64_MAGIC       0x20b

//
// Optional header format
//
typedef struct _IMAGE_OPTIONAL_HEADER {
    //
    // Standard fields.
    //
    WORD    Magic;
    BYTE    MajorLinkerVersion;
    BYTE    MinorLinkerVersion;
    DWORD   SizeOfCode;
    DWORD   SizeOfInitializedData;
    DWORD   SizeOfUninitializedData;
    DWORD   AddressOfEntryPoint;
    DWORD   BaseOfCode;
    DWORD   BaseOfData;

    //
    // NT additional fields.
    //
    DWORD   ImageBase;
    DWORD   SectionAlignment;
    DWORD   FileAlignment;
    WORD    MajorOperatingSystemVersion;
    WORD    MinorOperatingSystemVersion;
    WORD    MajorImageVersion;
    WORD    MinorImageVersion;
    WORD    MajorSubsystemVersion;
    WORD    MinorSubsystemVersion;
    DWORD   Win32VersionValue;
    DWORD   SizeOfImage;
    DWORD   SizeOfHeaders;
    DWORD   CheckSum;
    WORD    Subsystem;
    WORD    DllCharacteristics;
    DWORD   SizeOfStackReserve;
    DWORD   SizeOfStackCommit;
    DWORD   SizeOfHeapReserve;
    DWORD   SizeOfHeapCommit;
    DWORD   LoaderFlags;
    DWORD   NumberOfRvaAndSizes;
    IMAGE_DATA_DIRECTORY
DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER32, *PIMAGE_OPTIONAL_HEADER32;

typedef struct _IMAGE_OPTIONAL_HEADER64 {
    WORD        Magic;
    BYTE        MajorLinkerVersion;
    BYTE        MinorLinkerVersion;
    DWORD       SizeOfCode;
    DWORD       SizeOfInitializedData;
    DWORD       SizeOfUninitializedData;
    DWORD       AddressOfEntryPoint;
    DWORD       BaseOfCode;
    ULONGLONG   ImageBase;
    DWORD       SectionAlignment;
    DWORD       FileAlignment;
    WORD        MajorOperatingSystemVersion;
    WORD        MinorOperatingSystemVersion;
    WORD        MajorImageVersion;
    WORD        MinorImageVersion;
    WORD        MajorSubsystemVersion;
    WORD        MinorSubsystemVersion;
    DWORD       Win32VersionValue;
    DWORD       SizeOfImage;
    DWORD       SizeOfHeaders;
    DWORD       CheckSum;
    WORD        Subsystem;
    WORD        DllCharacteristics;
    ULONGLONG   SizeOfStackReserve;
    ULONGLONG   SizeOfStackCommit;
    ULONGLONG   SizeOfHeapReserve;
    ULONGLONG   SizeOfHeapCommit;
    DWORD       LoaderFlags;
    DWORD       NumberOfRvaAndSizes;
    IMAGE_DATA_DIRECTORY
DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER64, *PIMAGE_OPTIONAL_HEADER64;
```

```
//
// Directory format
//
typedef struct _IMAGE_DATA_DIRECTORY {
    DWORD   VirtualAddress;
    DWORD   Size;
} IMAGE_DATA_DIRECTORY, *PIMAGE_DATA_DIRECTORY;

#define IMAGE_NUMBEROF_DIRECTORY_ENTRIES    16

// Directory Entries
#define IMAGE_DIRECTORY_ENTRY_EXPORT         0   // Export Directory
#define IMAGE_DIRECTORY_ENTRY_IMPORT         1   // Import Directory
#define IMAGE_DIRECTORY_ENTRY_RESOURCE       2   // Resource Directory
#define IMAGE_DIRECTORY_ENTRY_EXCEPTION      3   // Exception Directory
#define IMAGE_DIRECTORY_ENTRY_SECURITY       4   // Security Directory
#define IMAGE_DIRECTORY_ENTRY_BASERELOC      5   // Base Relocation Table
#define IMAGE_DIRECTORY_ENTRY_DEBUG          6   // Debug Directory
//      IMAGE_DIRECTORY_ENTRY_COPYRIGHT      7   // (X86 usage)
#define IMAGE_DIRECTORY_ENTRY_ARCHITECTURE   7   // Architecture Specific Data
#define IMAGE_DIRECTORY_ENTRY_GLOBALPTR      8   // RVA of GP
#define IMAGE_DIRECTORY_ENTRY_TLS            9   // TLS Directory
#define IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG   10   // Load Configuration Directory
#define IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT  11   // Bound Import Directory in headers
#define IMAGE_DIRECTORY_ENTRY_IAT           12   // Import Address Table
#define IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT  13   // Delay Load Import Descriptors
#define IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR 14  // COM Runtime descriptor
```

```
//
// Subsystem Values
//
#define IMAGE_SUBSYSTEM_UNKNOWN              0   // Unknown subsystem.
#define IMAGE_SUBSYSTEM_NATIVE               1   // Image doesn't require a subsystem.
#define IMAGE_SUBSYSTEM_WINDOWS_GUI          2   // Image runs in the Windows GUI subsystem.
#define IMAGE_SUBSYSTEM_WINDOWS_CUI          3   // Image runs in the Windows character subsystem.
#define IMAGE_SUBSYSTEM_OS2_CUI              5   // image runs in the OS/2 character subsystem.
#define IMAGE_SUBSYSTEM_POSIX_CUI            7   // image runs in the Posix character subsystem.
#define IMAGE_SUBSYSTEM_NATIVE_WINDOWS       8   // image is a native Win9x driver.
#define IMAGE_SUBSYSTEM_WINDOWS_CE_GUI       9   // Image runs in the Windows CE subsystem.
#define IMAGE_SUBSYSTEM_EFI_APPLICATION     10   //
#define IMAGE_SUBSYSTEM_EFI_BOOT_SERVICE_DRIVER  11   //
#define IMAGE_SUBSYSTEM_EFI_RUNTIME_DRIVER  12   //
#define IMAGE_SUBSYSTEM_EFI_ROM             13
#define IMAGE_SUBSYSTEM_XBOX                14
#define IMAGE_SUBSYSTEM_WINDOWS_BOOT_APPLICATION 16
```

```
//
// DllCharacteristics Entries
//
//      IMAGE_LIBRARY_PROCESS_INIT          0x0001     // Reserved.
//      IMAGE_LIBRARY_PROCESS_TERM          0x0002     // Reserved.
//      IMAGE_LIBRARY_THREAD_INIT           0x0004     // Reserved.
//      IMAGE_LIBRARY_THREAD_TERM           0x0008     // Reserved.
#define IMAGE_DLLCHARACTERISTICS_DYNAMIC_BASE 0x0040     // DLL can move.
#define IMAGE_DLLCHARACTERISTICS_FORCE_INTEGRITY    0x0080     // Code Integrity Image
#define IMAGE_DLLCHARACTERISTICS_NX_COMPAT    0x0100     // Image is NX compatible
#define IMAGE_DLLCHARACTERISTICS_NO_ISOLATION 0x0200     // Image understands isolation and doesn't want it
#define IMAGE_DLLCHARACTERISTICS_NO_SEH       0x0400     // Image does not use SEH.  No SE handler may reside
                                                         // in this image
#define IMAGE_DLLCHARACTERISTICS_NO_BIND      0x0800     // Do not bind this image.
//                                    0x1000     // Reserved.
#define IMAGE_DLLCHARACTERISTICS_WDM_DRIVER   0x2000     // Driver uses WDM model
//                                    0x4000     // Reserved.
#define IMAGE_DLLCHARACTERISTICS_TERMINAL_SERVER_AWARE     0x8000
```

```
// IMAGE_FIRST_SECTION doesn't need 32/64 versions since the file header is the same either way.

#define IMAGE_FIRST_SECTION( ntheader ) ((PIMAGE_SECTION_HEADER)        \
    ((ULONG_PTR)(ntheader) +                                            \
     FIELD_OFFSET( IMAGE_NT_HEADERS, OptionalHeader ) +                 \
     ((ntheader))->FileHeader.SizeOfOptionalHeader   \
    ))
```

```
//
// Section header format
//
#define IMAGE_SIZEOF_SHORT_NAME              8

typedef struct _IMAGE_SECTION_HEADER {
    BYTE    Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
            DWORD   PhysicalAddress;
            DWORD   VirtualSize;
    } Misc;
    DWORD   VirtualAddress;
    DWORD   SizeOfRawData;
    DWORD   PointerToRawData;
    DWORD   PointerToRelocations;
    DWORD   PointerToLinenumbers;
    WORD    NumberOfRelocations;
    WORD    NumberOfLinenumbers;
    DWORD   Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;

#define IMAGE_SIZEOF_SECTION_HEADER          40

//
// Section characteristics.
//
//      IMAGE_SCN_TYPE_REG                   0x00000000  // Reserved.
//      IMAGE_SCN_TYPE_DSECT                 0x00000001  // Reserved.
//      IMAGE_SCN_TYPE_NOLOAD                0x00000002  // Reserved.
//      IMAGE_SCN_TYPE_GROUP                 0x00000004  // Reserved.
#define IMAGE_SCN_TYPE_NO_PAD                0x00000008  // Reserved.
//      IMAGE_SCN_TYPE_COPY                  0x00000010  // Reserved.

#define IMAGE_SCN_CNT_CODE                   0x00000020  // Section contains code.
#define IMAGE_SCN_CNT_INITIALIZED_DATA       0x00000040  // Section contains initialized data.
#define IMAGE_SCN_CNT_UNINITIALIZED_DATA     0x00000080  // Section contains uninitialized data.

#define IMAGE_SCN_LNK_OTHER                  0x00000100  // Reserved.
#define IMAGE_SCN_LNK_INFO                   0x00000200  // Section contains comments or some other type of information.
//      IMAGE_SCN_TYPE_OVER                  0x00000400  // Reserved.
#define IMAGE_SCN_LNK_REMOVE                 0x00000800  // Section contents will not become part of image.
#define IMAGE_SCN_LNK_COMDAT                 0x00001000  // Section contents comdat.
//                                           0x00002000  // Reserved.
//      IMAGE_SCN_MEM_PROTECTED - Obsolete   0x00004000
#define IMAGE_SCN_NO_DEFER_SPEC_EXC          0x00004000  // Reset speculative exceptions handling bits in the TLB entries
                                                         // for this section.
#define IMAGE_SCN_GPREL                      0x00008000  // Section content can be accessed relative to GP
#define IMAGE_SCN_MEM_FARDATA                0x00008000
//      IMAGE_SCN_MEM_SYSHEAP  - Obsolete    0x00010000
#define IMAGE_SCN_MEM_PURGEABLE              0x00020000
#define IMAGE_SCN_MEM_16BIT                  0x00020000
#define IMAGE_SCN_MEM_LOCKED                 0x00040000
#define IMAGE_SCN_MEM_PRELOAD                0x00080000

#define IMAGE_SCN_ALIGN_1BYTES               0x00100000  //
#define IMAGE_SCN_ALIGN_2BYTES               0x00200000  //
#define IMAGE_SCN_ALIGN_4BYTES               0x00300000  //
#define IMAGE_SCN_ALIGN_8BYTES               0x00400000  //
#define IMAGE_SCN_ALIGN_16BYTES              0x00500000  // Default alignment if no others are specified.
#define IMAGE_SCN_ALIGN_32BYTES              0x00600000  //
#define IMAGE_SCN_ALIGN_64BYTES              0x00700000  //
#define IMAGE_SCN_ALIGN_128BYTES             0x00800000  //
#define IMAGE_SCN_ALIGN_256BYTES             0x00900000  //
#define IMAGE_SCN_ALIGN_512BYTES             0x00A00000  //
#define IMAGE_SCN_ALIGN_1024BYTES            0x00B00000  //
#define IMAGE_SCN_ALIGN_2048BYTES            0x00C00000  //
#define IMAGE_SCN_ALIGN_4096BYTES            0x00D00000  //
#define IMAGE_SCN_ALIGN_8192BYTES            0x00E00000  //
// Unused                                    0x00F00000
#define IMAGE_SCN_ALIGN_MASK                 0x00F00000

#define IMAGE_SCN_LNK_NRELOC_OVFL            0x01000000  // Section contains extended relocations.
#define IMAGE_SCN_MEM_DISCARDABLE            0x02000000  // Section can be discarded.
#define IMAGE_SCN_MEM_NOT_CACHED             0x04000000  // Section is not cachable.
#define IMAGE_SCN_MEM_NOT_PAGED              0x08000000  // Section is not pageable.
#define IMAGE_SCN_MEM_SHARED                 0x10000000  // Section is shareable.
#define IMAGE_SCN_MEM_EXECUTE                0x20000000  // Section is executable.
#define IMAGE_SCN_MEM_READ                   0x40000000  // Section is readable.
#define IMAGE_SCN_MEM_WRITE                  0x80000000  // Section is writeable.

//
// TLS Chacteristic Flags
//
#define IMAGE_SCN_SCALE_INDEX                0x00000001  // Tls index is scaled
```

```
//
// Thread Local Storage
//
typedef VOID
(NTAPI *PIMAGE_TLS_CALLBACK) (
    PVOID DllHandle,
    DWORD Reason,
    PVOID Reserved
    );

typedef struct _IMAGE_TLS_DIRECTORY64 {
    ULONGLONG   StartAddressOfRawData;
    ULONGLONG   EndAddressOfRawData;
    ULONGLONG   AddressOfIndex;         // PDWORD
    ULONGLONG   AddressOfCallBacks;     // PIMAGE_TLS_CALLBACK *;
    DWORD   SizeOfZeroFill;
    DWORD   Characteristics;
} IMAGE_TLS_DIRECTORY64;
typedef IMAGE_TLS_DIRECTORY64 * PIMAGE_TLS_DIRECTORY64;

typedef struct _IMAGE_TLS_DIRECTORY32 {
    DWORD   StartAddressOfRawData;
    DWORD   EndAddressOfRawData;
    DWORD   AddressOfIndex;             // PDWORD
    DWORD   AddressOfCallBacks;         // PIMAGE_TLS_CALLBACK *
    DWORD   SizeOfZeroFill;
    DWORD   Characteristics;
} IMAGE_TLS_DIRECTORY32;
typedef IMAGE_TLS_DIRECTORY32 * PIMAGE_TLS_DIRECTORY32;
```

```
//
// Load Configuration Directory Entry
//
typedef struct {
    DWORD   Size;
    DWORD   TimeDateStamp;
    WORD    MajorVersion;
    WORD    MinorVersion;
    DWORD   GlobalFlagsClear;
    DWORD   GlobalFlagsSet;
    DWORD   CriticalSectionDefaultTimeout;
    DWORD   DeCommitFreeBlockThreshold;
    DWORD   DeCommitTotalFreeThreshold;
    DWORD   LockPrefixTable;            // VA
    DWORD   MaximumAllocationSize;
    DWORD   VirtualMemoryThreshold;
    DWORD   ProcessHeapFlags;
    DWORD   ProcessAffinityMask;
    WORD    CSDVersion;
    WORD    Reserved1;
    DWORD   EditList;                   // VA
    DWORD   SecurityCookie;             // VA
    DWORD   SEHandlerTable;             // VA
    DWORD   SEHandlerCount;
} IMAGE_LOAD_CONFIG_DIRECTORY32, *PIMAGE_LOAD_CONFIG_DIRECTORY32;

typedef struct {
    DWORD       Size;
    DWORD       TimeDateStamp;
    WORD        MajorVersion;
    WORD        MinorVersion;
    DWORD       GlobalFlagsClear;
    DWORD       GlobalFlagsSet;
    DWORD       CriticalSectionDefaultTimeout;
    ULONGLONG   DeCommitFreeBlockThreshold;
    ULONGLONG   DeCommitTotalFreeThreshold;
    ULONGLONG   LockPrefixTable;        // VA
    ULONGLONG   MaximumAllocationSize;
    ULONGLONG   VirtualMemoryThreshold;
    ULONGLONG   ProcessAffinityMask;
    DWORD       ProcessHeapFlags;
    WORD        CSDVersion;
    WORD        Reserved1;
    ULONGLONG   EditList;               // VA
    ULONGLONG   SecurityCookie;         // VA
    ULONGLONG   SEHandlerTable;         // VA
    ULONGLONG   SEHandlerCount;
} IMAGE_LOAD_CONFIG_DIRECTORY64, *PIMAGE_LOAD_CONFIG_DIRECTORY64;
```

```
//
// Import Format
//
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD   Characteristics;            // 0 for terminating null import descriptor
        DWORD   OriginalFirstThunk;         // RVA to original unbound IAT (PIMAGE_THUNK_DATA)
    } DUMMYUNIONNAME;
    DWORD   TimeDateStamp;                  // 0 if not bound,
                                            // -1 if bound, and real date\time stamp
                                            //     in IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT (new BIND)
                                            // O.W. date/time stamp of DLL bound to (Old BIND)

    DWORD   ForwarderChain;                 // -1 if no forwarders
    DWORD   Name;
    DWORD   FirstThunk;                     // RVA to IAT (if bound this IAT has actual addresses)
} IMAGE_IMPORT_DESCRIPTOR;
typedef IMAGE_IMPORT_DESCRIPTOR UNALIGNED *PIMAGE_IMPORT_DESCRIPTOR;

//
// New format import descriptors pointed to by DataDirectory[ IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT ]
//
typedef struct _IMAGE_BOUND_IMPORT_DESCRIPTOR {
    DWORD   TimeDateStamp;
    WORD    OffsetModuleName;
    WORD    NumberOfModuleForwarderRefs;
// Array of zero or more IMAGE_BOUND_FORWARDER_REF follows
} IMAGE_BOUND_IMPORT_DESCRIPTOR,  *PIMAGE_BOUND_IMPORT_DESCRIPTOR;

typedef struct _IMAGE_BOUND_FORWARDER_REF {
    DWORD   TimeDateStamp;
    WORD    OffsetModuleName;
    WORD    Reserved;
} IMAGE_BOUND_FORWARDER_REF, *PIMAGE_BOUND_FORWARDER_REF;

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

#include "pshpack8.h"                       // Use align 8 for the 64-bit IAT.

typedef struct _IMAGE_THUNK_DATA64 {
    union {
        ULONGLONG ForwarderString;  // PBYTE
        ULONGLONG Function;         // PDWORD
        ULONGLONG Ordinal;
        ULONGLONG AddressOfData;    // PIMAGE_IMPORT_BY_NAME
    } u1;
} IMAGE_THUNK_DATA64;
typedef IMAGE_THUNK_DATA64 * PIMAGE_THUNK_DATA64;

#include "poppack.h"                        // Back to 4 byte packing

typedef struct _IMAGE_THUNK_DATA32 {
    union {
        DWORD ForwarderString;      // PBYTE
        DWORD Function;             // PDWORD
        DWORD Ordinal;
        DWORD AddressOfData;        // PIMAGE_IMPORT_BY_NAME
    } u1;
} IMAGE_THUNK_DATA32;
typedef IMAGE_THUNK_DATA32 * PIMAGE_THUNK_DATA32;

#define IMAGE_ORDINAL_FLAG64 0x8000000000000000
#define IMAGE_ORDINAL_FLAG32 0x80000000
```

```
//
// Resource Format
//

//
// Resource directory consists of two counts, following by a variable length
// array of directory entries.  The first count is the number of entries at
// beginning of the array that have actual names associated with each entry.
// The entries are in ascending order, case insensitive strings.  The second
// count is the number of entries that immediately follow the named entries.
// This second count identifies the number of entries that have 16-bit integer
// Ids as their name.  These entries are also sorted in ascending order.
//
// This structure allows fast lookup by either name or number, but for any
// given resource entry only one form of lookup is supported, not both.
// This is consistant with the syntax of the .RC file and the .RES file.
//
typedef struct _IMAGE_RESOURCE_DIRECTORY {
    DWORD   Characteristics;
    DWORD   TimeDateStamp;
    WORD    MajorVersion;
    WORD    MinorVersion;
    WORD    NumberOfNamedEntries;
    WORD    NumberOfIdEntries;
//  IMAGE_RESOURCE_DIRECTORY_ENTRY DirectoryEntries[];
} IMAGE_RESOURCE_DIRECTORY, *PIMAGE_RESOURCE_DIRECTORY;

#define IMAGE_RESOURCE_NAME_IS_STRING        0x80000000
#define IMAGE_RESOURCE_DATA_IS_DIRECTORY     0x80000000

//
// Each directory contains the 32-bit Name of the entry and an offset,
// relative to the beginning of the resource directory of the data associated
// with this directory entry.  If the name of the entry is an actual text
// string instead of an integer Id, then the high order bit of the name field
// is set to one and the low order 31-bits are an offset, relative to the
// beginning of the resource directory of the string, which is of type
// IMAGE_RESOURCE_DIRECTORY_STRING.  Otherwise the high bit is clear and the
// low-order 16-bits are the integer Id that identify this resource directory
// entry. If the high bit of the directory entry is yet another resource directory (i.e. a
// subdirectory), then the high order bit of the offset field will be
// set to indicate this.  Otherwise the high bit is clear and the offset
// field points to a resource data entry.
//
typedef struct _IMAGE_RESOURCE_DIRECTORY_ENTRY {
    union {
        struct {
            DWORD NameOffset:31;
            DWORD NameIsString:1;
        } DUMMYSTRUCTNAME;
        DWORD   Name;
        WORD    Id;
    } DUMMYUNIONNAME;
    union {
        DWORD   OffsetToData;
        struct {
            DWORD   OffsetToDirectory:31;
            DWORD   DataIsDirectory:1;
        } DUMMYSTRUCTNAME2;
    } DUMMYUNIONNAME2;
} IMAGE_RESOURCE_DIRECTORY_ENTRY, *PIMAGE_RESOURCE_DIRECTORY_ENTRY;

//
// For resource directory entries that have actual string names, the Name
// field of the directory entry points to an object of the following type.
// All of these string objects are stored together after the last resource
// directory entry and before the first resource data object.  This minimizes
// the impact of these variable length objects on the alignment of the fixed
// size directory entry objects.
//
typedef struct _IMAGE_RESOURCE_DIRECTORY_STRING {
    WORD    Length;
    CHAR    NameString[ 1 ];
} IMAGE_RESOURCE_DIRECTORY_STRING, *PIMAGE_RESOURCE_DIRECTORY_STRING;

typedef struct _IMAGE_RESOURCE_DIR_STRING_U {
    WORD    Length;
    WCHAR   NameString[ 1 ];
} IMAGE_RESOURCE_DIR_STRING_U, *PIMAGE_RESOURCE_DIR_STRING_U;

//
// Each resource data entry describes a leaf node in the resource directory
// tree.  It contains an offset, relative to the beginning of the resource
// directory of the data for the resource, a size field that gives the number
// of bytes of data at that offset, a CodePage that should be used when
// decoding code point values within the resource data.  Typically for new
// applications the code page would be the unicode code page.
//
typedef struct _IMAGE_RESOURCE_DATA_ENTRY {
    DWORD   OffsetToData;
    DWORD   Size;
    DWORD   CodePage;
    DWORD   Reserved;
} IMAGE_RESOURCE_DATA_ENTRY, *PIMAGE_RESOURCE_DATA_ENTRY;
```

```
//
// Delay Import Format
//
typedef struct _IMAGE_DELAY_IMPORT_DESCRIPTOR {
    DWORD Characteristics;          // Attributes (addressing mode 1 VA/0 RVA)
    DWORD Name;                     // DLL name
    DWORD ModHandle;                // Module handle
    DWORD pIAT;                     // Delayed Import Address Table
    DWORD pNames;                   // Delayed Import Name Table
    DWORD pBoundIAT;                // Bound Delayed Import Address Table
    DWORD pUnloadIAT;               // Unload Delayed Import Table
    DWORD TimeDateStamp;            // Time stamp
} IMAGE_DELAY_IMPORT_DESCRIPTOR, *PIMAGE_DELAY_IMPORT_DESCRIPTOR;
```

```
//
// Export Format
//
typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD   Characteristics;
    DWORD   TimeDateStamp;
    WORD    MajorVersion;
    WORD    MinorVersion;
    DWORD   Name;
    DWORD   Base;
    DWORD   NumberOfFunctions;
    DWORD   NumberOfNames;
    DWORD   AddressOfFunctions;     // RVA from base of image
    DWORD   AddressOfNames;         // RVA from base of image
    DWORD   AddressOfNameOrdinals;  // RVA from base of image
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;
```

```
//
// Based relocation format
//
typedef struct _IMAGE_BASE_RELOCATION {
    DWORD   VirtualAddress;
    DWORD   SizeOfBlock;
//  WORD    TypeOffset[1];
} IMAGE_BASE_RELOCATION;
typedef IMAGE_BASE_RELOCATION UNALIGNED * PIMAGE_BASE_RELOCATION;

//
// Based relocation types
//
#define IMAGE_REL_BASED_ABSOLUTE              0
#define IMAGE_REL_BASED_HIGH                  1
#define IMAGE_REL_BASED_LOW                   2
#define IMAGE_REL_BASED_HIGHLOW               3
#define IMAGE_REL_BASED_HIGHADJ               4
#define IMAGE_REL_BASED_MIPS_JMPADDR          5
#define IMAGE_REL_BASED_MIPS_JMPADDR16        9
#define IMAGE_REL_BASED_IA64_IMM64            9
#define IMAGE_REL_BASED_DIR64                 10
```

```
//
// Debug Format
//
typedef struct _IMAGE_DEBUG_DIRECTORY {
    DWORD   Characteristics;
    DWORD   TimeDateStamp;
    WORD    MajorVersion;
    WORD    MinorVersion;
    DWORD   Type;
    DWORD   SizeOfData;
    DWORD   AddressOfRawData;
    DWORD   PointerToRawData;
} IMAGE_DEBUG_DIRECTORY, *PIMAGE_DEBUG_DIRECTORY;
```

```
//
// CLR 2.0 header structure
//
typedef struct IMAGE_COR20_HEADER
{
    // Header versioning
    DWORD                   cb;
    WORD                    MajorRuntimeVersion;
    WORD                    MinorRuntimeVersion;

    // Symbol table and startup information
    IMAGE_DATA_DIRECTORY    MetaData;
    DWORD                   Flags;

    // If COMIMAGE_FLAGS_NATIVE_ENTRYPOINT is not set, EntryPointToken represents a managed entrypoint.
    // If COMIMAGE_FLAGS_NATIVE_ENTRYPOINT is set, EntryPointRVA represents an RVA to a native entrypoint.
    union {
        DWORD               EntryPointToken;
        DWORD               EntryPointRVA;
    } DUMMYUNIONNAME;

    // Binding information
    IMAGE_DATA_DIRECTORY    Resources;
    IMAGE_DATA_DIRECTORY    StrongNameSignature;

    // Regular fixup and binding information
    IMAGE_DATA_DIRECTORY    CodeManagerTable;
    IMAGE_DATA_DIRECTORY    VTableFixups;
    IMAGE_DATA_DIRECTORY    ExportAddressTableJumps;

    // Precompiled image info (internal use only - set to zero)
    IMAGE_DATA_DIRECTORY    ManagedNativeHeader;
} IMAGE_COR20_HEADER, *PIMAGE_COR20_HEADER;

typedef enum ReplacesCorHdrNumericDefines
{
    // COM+ Header entry point flags.
    COMIMAGE_FLAGS_ILONLY               = 0x00000001,
    COMIMAGE_FLAGS_32BITREQUIRED        = 0x00000002,
    COMIMAGE_FLAGS_IL_LIBRARY           = 0x00000004,
    COMIMAGE_FLAGS_STRONGNAMESIGNED     = 0x00000008,
    COMIMAGE_FLAGS_NATIVE_ENTRYPOINT    = 0x00000010,
    COMIMAGE_FLAGS_TRACKDEBUGDATA       = 0x00010000,

    // Version flags for image.
    COR_VERSION_MAJOR_V2                = 2,
    COR_VERSION_MAJOR                   = COR_VERSION_MAJOR_V2,
    COR_VERSION_MINOR                   = 0,
    COR_DELETED_NAME_LENGTH             = 8,
    COR_VTABLEGAP_NAME_LENGTH           = 8,

    // Maximum size of a NativeType descriptor.
    NATIVE_TYPE_MAX_CB                  = 1,
    COR_ILMETHOD_SECT_SMALL_MAX_DATASIZE= 0xFF,

    // #defines for the MIH FLAGS
    IMAGE_COR_MIH_METHODRVA             = 0x01,
    IMAGE_COR_MIH_EHRVA                 = 0x02,
    IMAGE_COR_MIH_BASICBLOCK            = 0x08,

    // V-table constants
    COR_VTABLE_32BIT                    = 0x01,     // V-table slots are 32-bits in size.
    COR_VTABLE_64BIT                    = 0x02,     // V-table slots are 64-bits in size.
    COR_VTABLE_FROM_UNMANAGED           = 0x04,     // If set, transition from unmanaged.
    COR_VTABLE_FROM_UNMANAGED_RETAIN_APPDOMAIN  = 0x08, // If set, transition from unmanaged with keeping
                                                        // the current appdomain.
    COR_VTABLE_CALL_MOST_DERIVED        = 0x10,     // Call most derived method described by

    // EATJ constants
    IMAGE_COR_EATJ_THUNK_SIZE           = 32,       // Size of a jump thunk reserved range.

    // Max name lengths
    //@todo: Change to unlimited name lengths.
    MAX_CLASS_NAME                      = 1024,
    MAX_PACKAGE_NAME                    = 1024,
} ReplacesCorHdrNumericDefines;
```

The Portable Executable (PE/PE32+) file format structures from Microsoft Windows SDK v7.0A (WinNT.h)

Created by **Bartosz Wójcik**

Visit us at http://www.pelock.com

Contact me at support@pelock.com

Last updated on **14.02.2012**