

Windows Portable Executable

> How executable files work

Portable Executable

- Format for files that can be loaded and executed
 - on all Windows flavors
 - on all supported CPU types

Goal

- Specify the expectations of an executable
 - Type (unmanaged, managed)
 - Subsystem (boot, native, GUI, console, EFI,...)
 - Binding (libraries, imports, types)
 - Sections (types, size, attributes, permissions...)
 - Resources (cursor, icon, dlg, ...)
 - Memory (stack, heap, size, alignment, TLS)
 - Boundaries (UAC, UIPI,...)
 - Security (certificate, checksum, mitigations, ...)
 - Exceptions, Relocations
 - and...the Entry-point

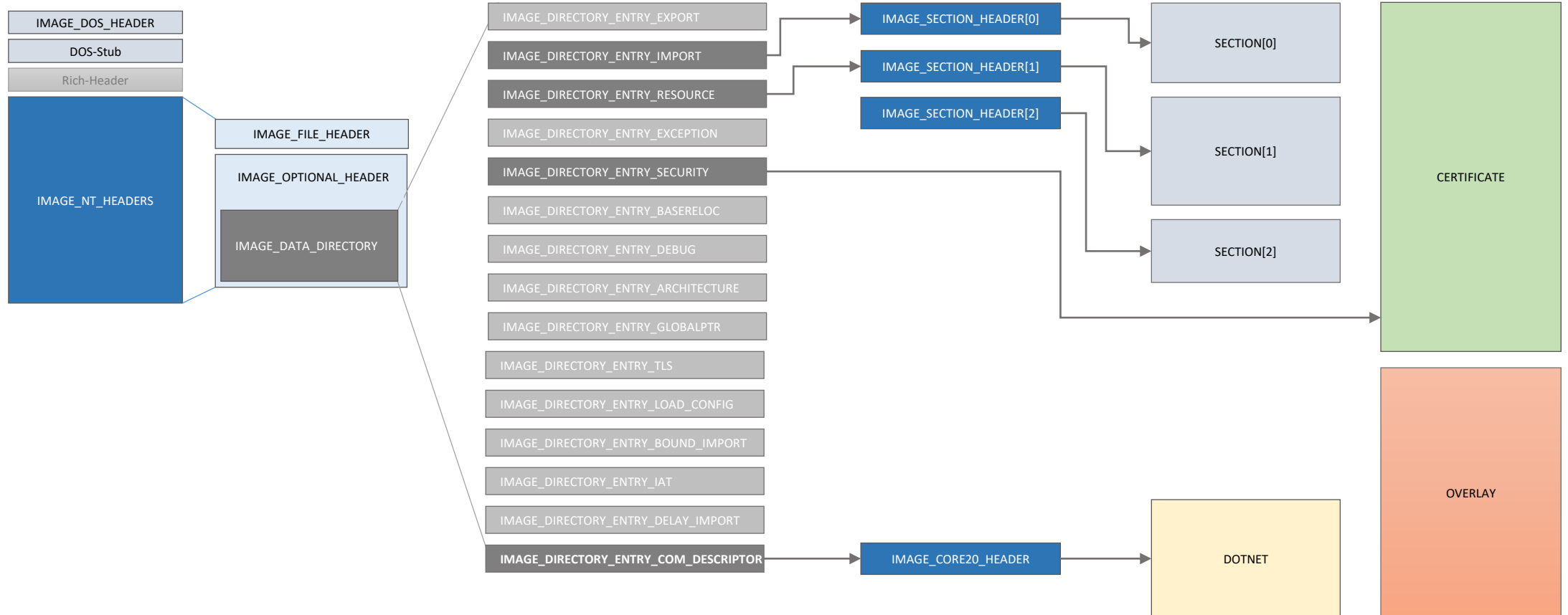
Files

type	summary
acm	audio compression manager (audio codec device driver)
ax	Direct show filter
cpl	control panel library
dll	dynamic-link library
drv	driver
efi	extensible firmware interface
exe	executable
msstyles	XP skinning
mui	multi user interface library
ocx	ActiveX control
olb	object library
rll	SQL resource library
scr	screen saver library
sys	driver
tlb	type library
xll	Excell add-in
...	

Windows Portable Executable

> How executable files work

Layout



Data-Directory

- Collection of predefined data structures that must be quickly located

```
#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
```

from: winnt.h

Imports

- Symbols exported by external executable files
 - implicitly Linking
 - Loader resolves (recursively) the dependencies by traversing the imports table
 - Loader invokes [LoadLibrary](#) and [GetProcAddress](#) for each function found
 - Loader updates the import-address table
 - Loader aborts the process if one dependency fails.
 - dynamically Linking
 - Developer must resolve dependencies explicitly using [LoadLibrary](#) and [GetProcAddress](#)
 - Developer is responsible (has the opportunity) to handle missing dependencies
 - delay-loaded Linking
 - Linker emits stub that is resolved only when a function is invoked
 - Loader traverses the delay-loaded table and invokes [LoadLibrary](#) and [GetProcAddress](#)

Import-name Table (INT)

- Array of arrays
 - one per imported library
 - Each array with the name of the imported library
 - Each array points an array of functions pointers (IAT)

Import-Address Table (IAT)

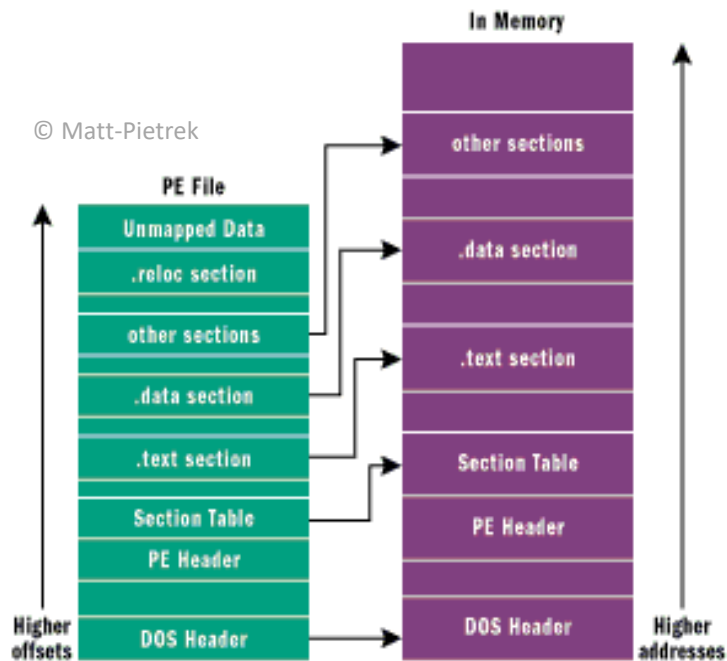
- Single place where imported functions addresses are stored
 - Once an Executable is loaded, its IAT contains the address of (all) functions that will be invoked
 - Each imported function has its own (one) spot in the IAT
 - No matter their numbers, all calls go through a single function pointer

Sections

- Content
 - code
 - data (global variable, imports, exports, resources, relocations, ...)
- Attributes
 - memory attributes (RWE, shareable, pageable)
 - name
 - is solely for the humans
 - is ignored by the OS
 - can indicate the purpose of the section (e.g. res, idata, code, text...)
 - alignments
 - file alignment
 - memory alignment

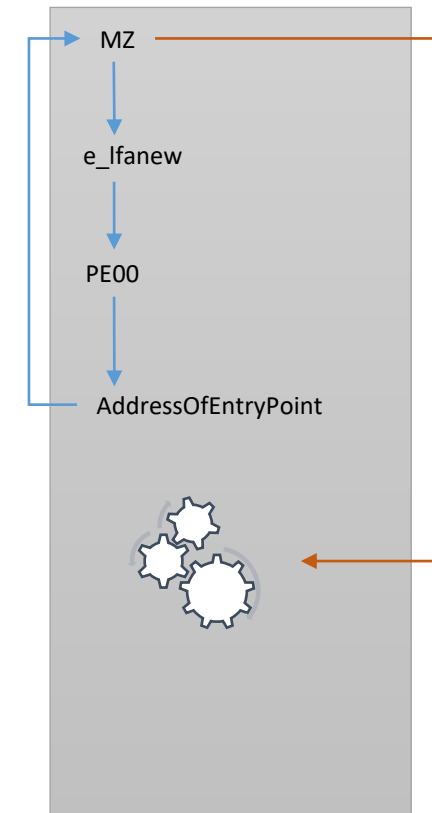
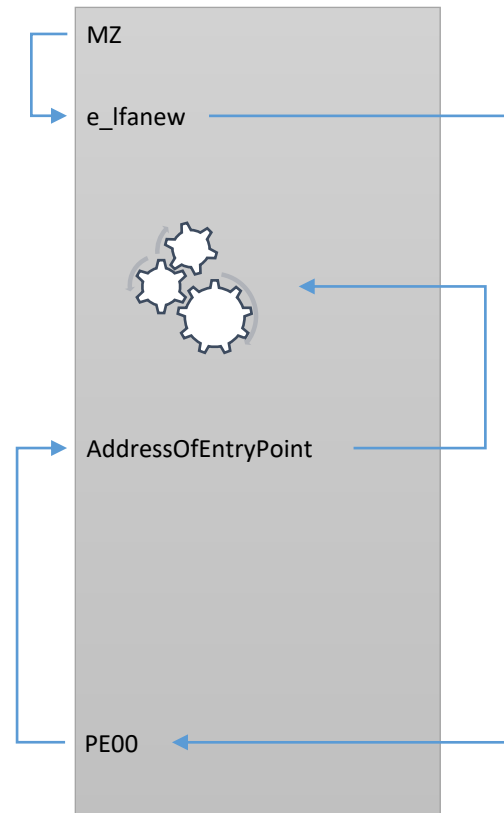
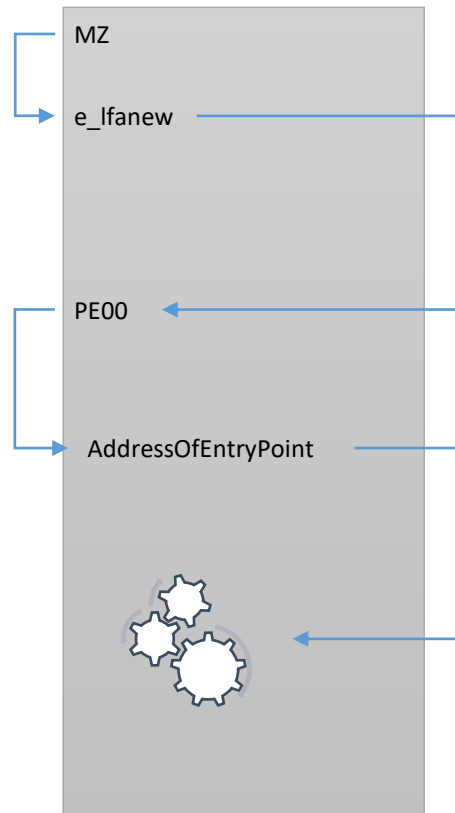
Memory mapping

- A file is not mapped as continuous blocks
 - Loader decides the items to map
 - Loader decides the address
- Some items are not mapped
 - Relocation (as appropriate...)
 - Certificate
 - Debug
 - Overlay



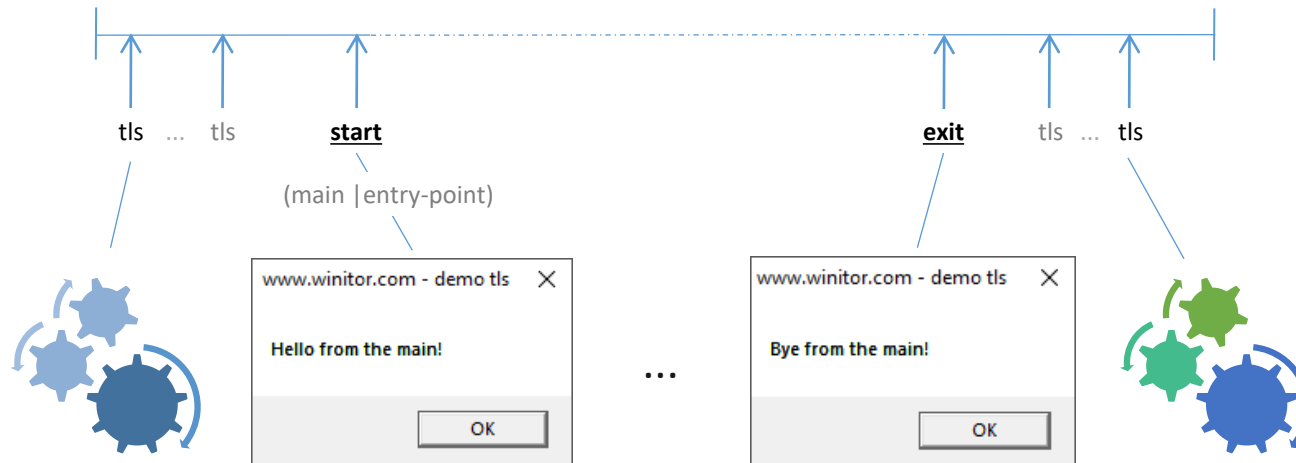
Process start flow

- Typical > Specific



Thread Local Storage

- Action
 - before the entry-point is reached
 - after the last thread has finished!



References

Peering Inside the PE: A Tour of the Win32 Portable Executable File Format ⁽¹⁾

http://bytepointer.com/resources/pietrek_peering_inside_pe.htm

An In-Depth Look into the Win32 Portable Executable File Format ⁽²⁾

<https://msdn.microsoft.com/en-us/magazine/bb985992.aspx>

Backup slides...

Windows Portable Executable

> How executable files work

