





# UEFI & EDK II Training

**EDK II Modules: Libraries, Drivers & Applications**

[tianocore.org](https://tianocore.org)

# Lesson Objective

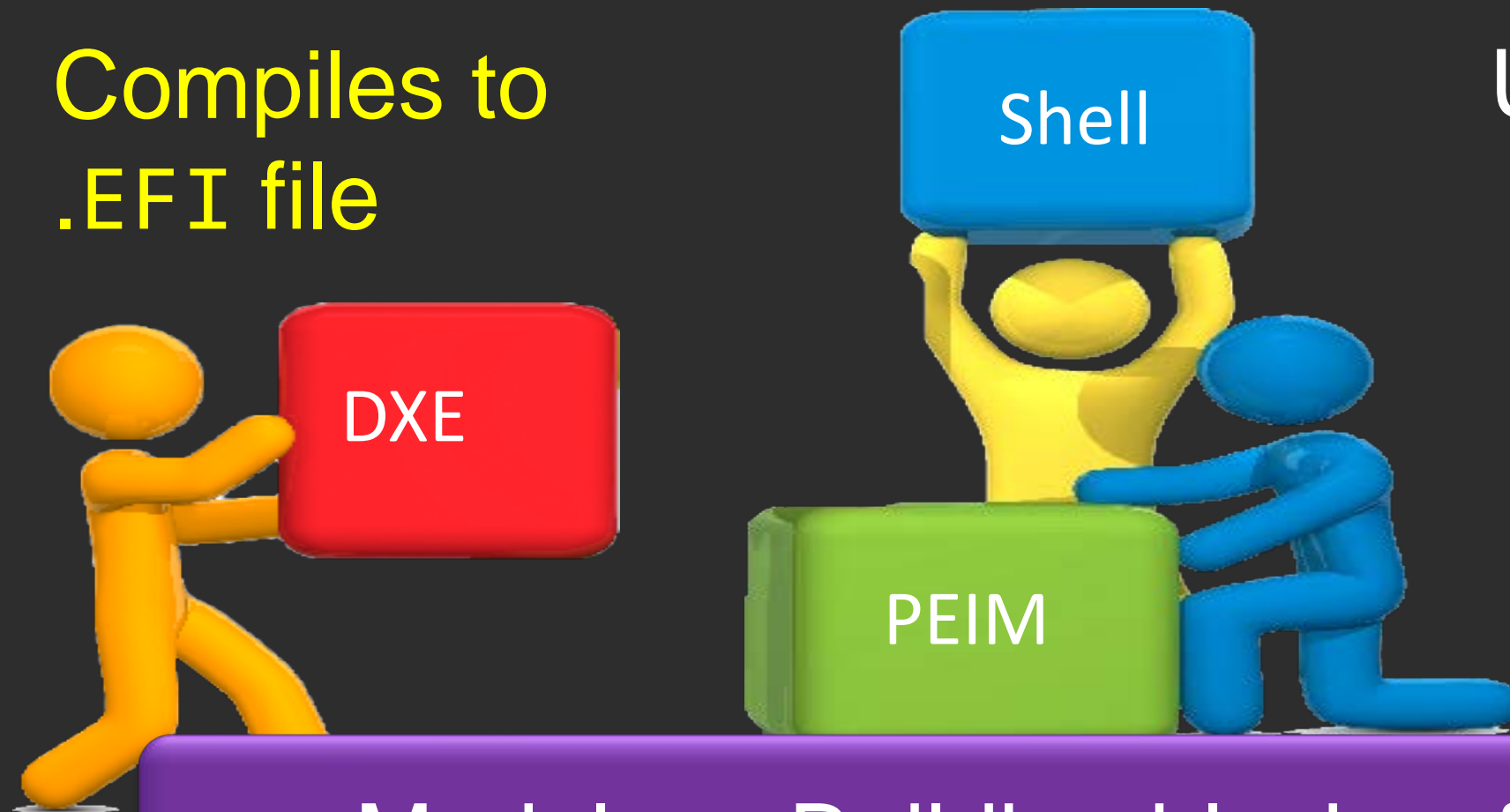
-  What is a EDK II Module
-  Use EDK II libraries to write UEFI apps/drivers
-  How to Define a UEFI application
-  Differences between UEFI App / Drivers INF file

# EDK II MODULES OVERVIEW

What are EDK II Modules

Smallest separate object compiled in EDK II

Compiles to  
.EFI file



UEFI/DXE Driver

PEIM

UEFI App. or  
Library

Modules: Building blocks of EDK II

## Most Used Module Types

PEI\_CORE

UEFI\_APPLICATION

DXE\_CORE

BASE

DXE\_RUNTIME\_DRIVER

PEIM

UEFI\_DRIVER

DXE\_DRIVER

Syntax:

```
<ModuleTypes> ::= <ModuleType> [<Space> <ModuleType>]
```

# MODULE SOURCE CONTENTS - MINIMUM FILE

| MODULE_TYPE      | Example Source files   |
|------------------|--|
| UEFI_APPLICATION | Foo.c, Foo.inf   |
| UEFI_DRIVER      | FooDriver.c, FooDriver.h, FooDriver.vfr,<br>FooDriver.uni, FooDriver.inf |

DXE\_DRIVER

FooDxe.c, FooDxe.h, FooDxe.inf

DXE, UEFI PEIM

FooLib.c, FooLib.h, FooLib.inf (w/

Library

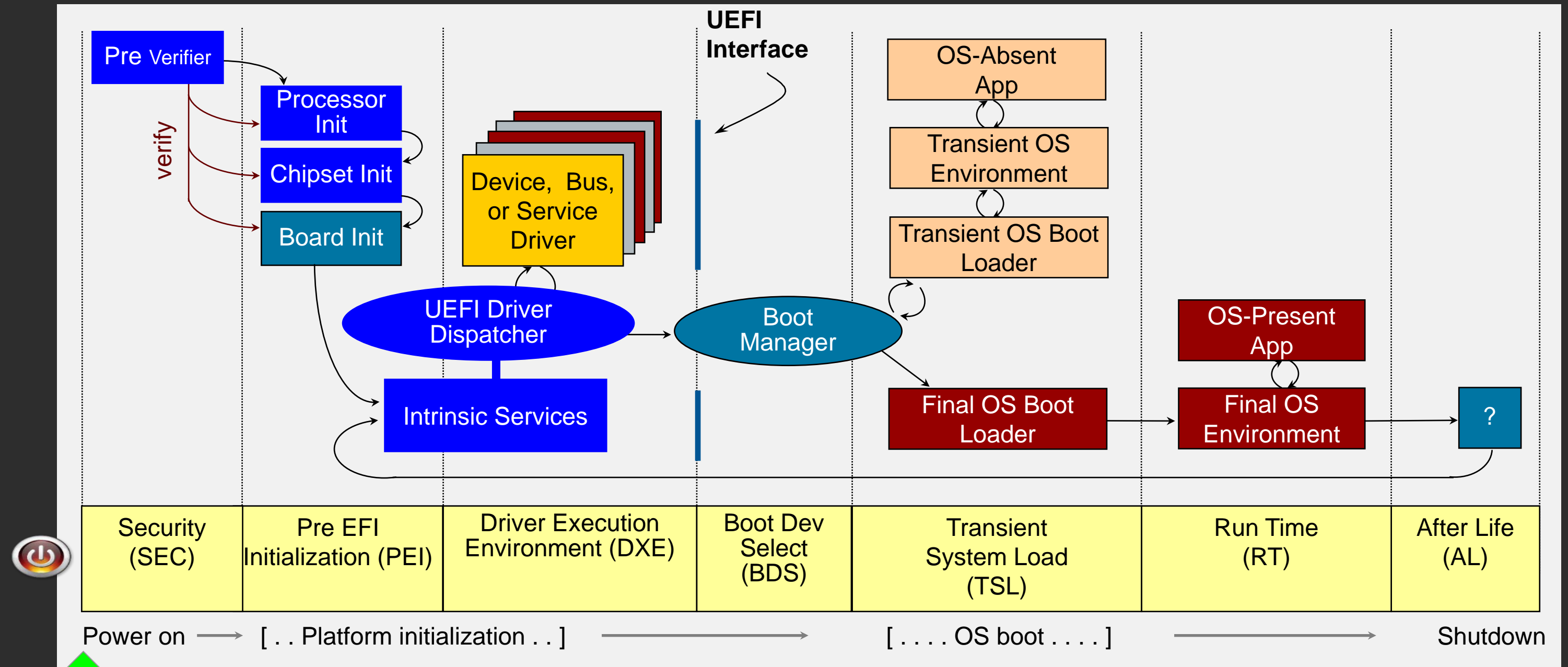
LIBRARY\_CLASS=)

**Complexity - Greater number of source files**

**.INF file - One file is required per module**

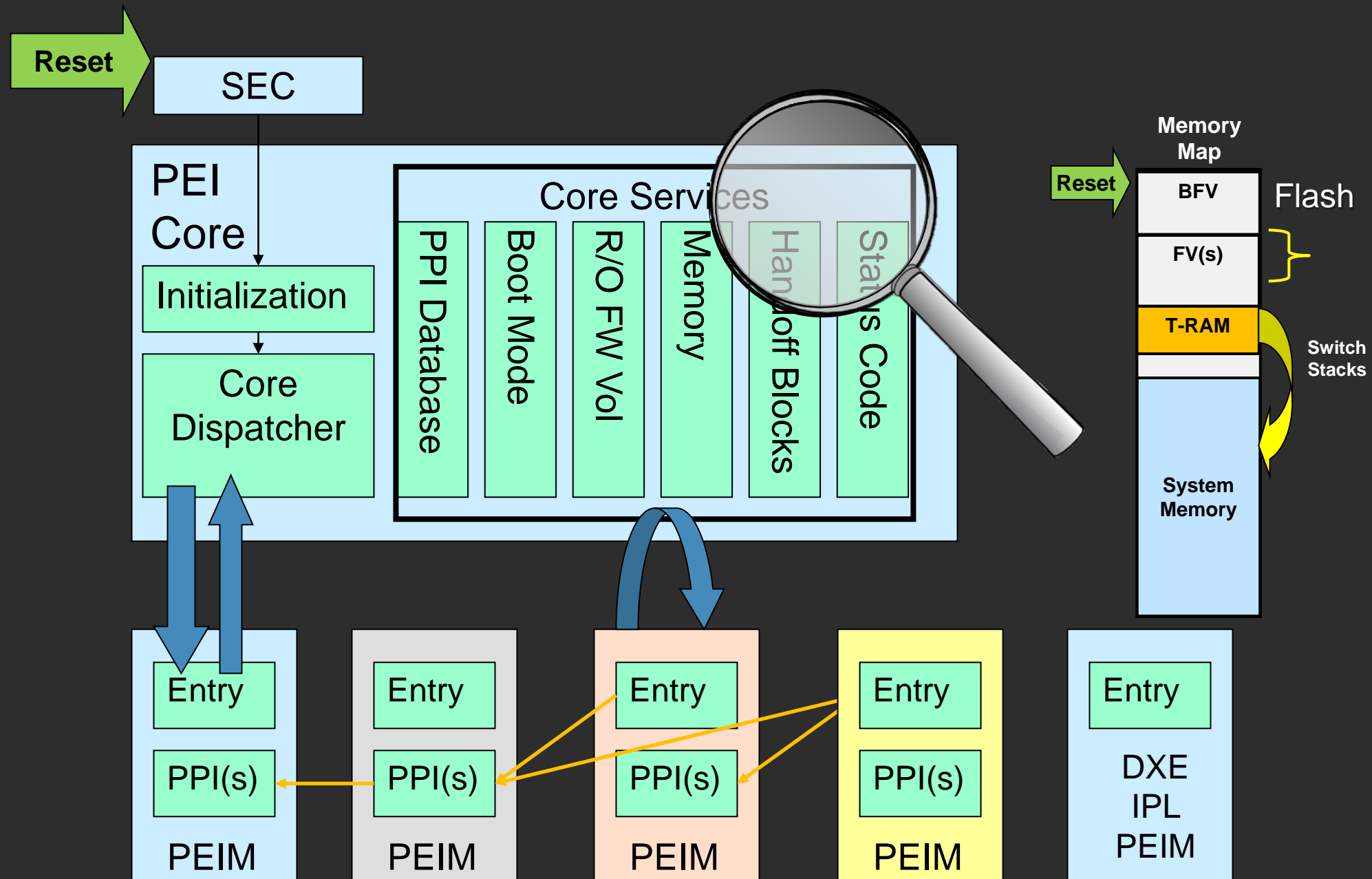
**.EFI file - Sources compiled to a single .EFI file**

# UEFI – PI & EDK II Boot Flow

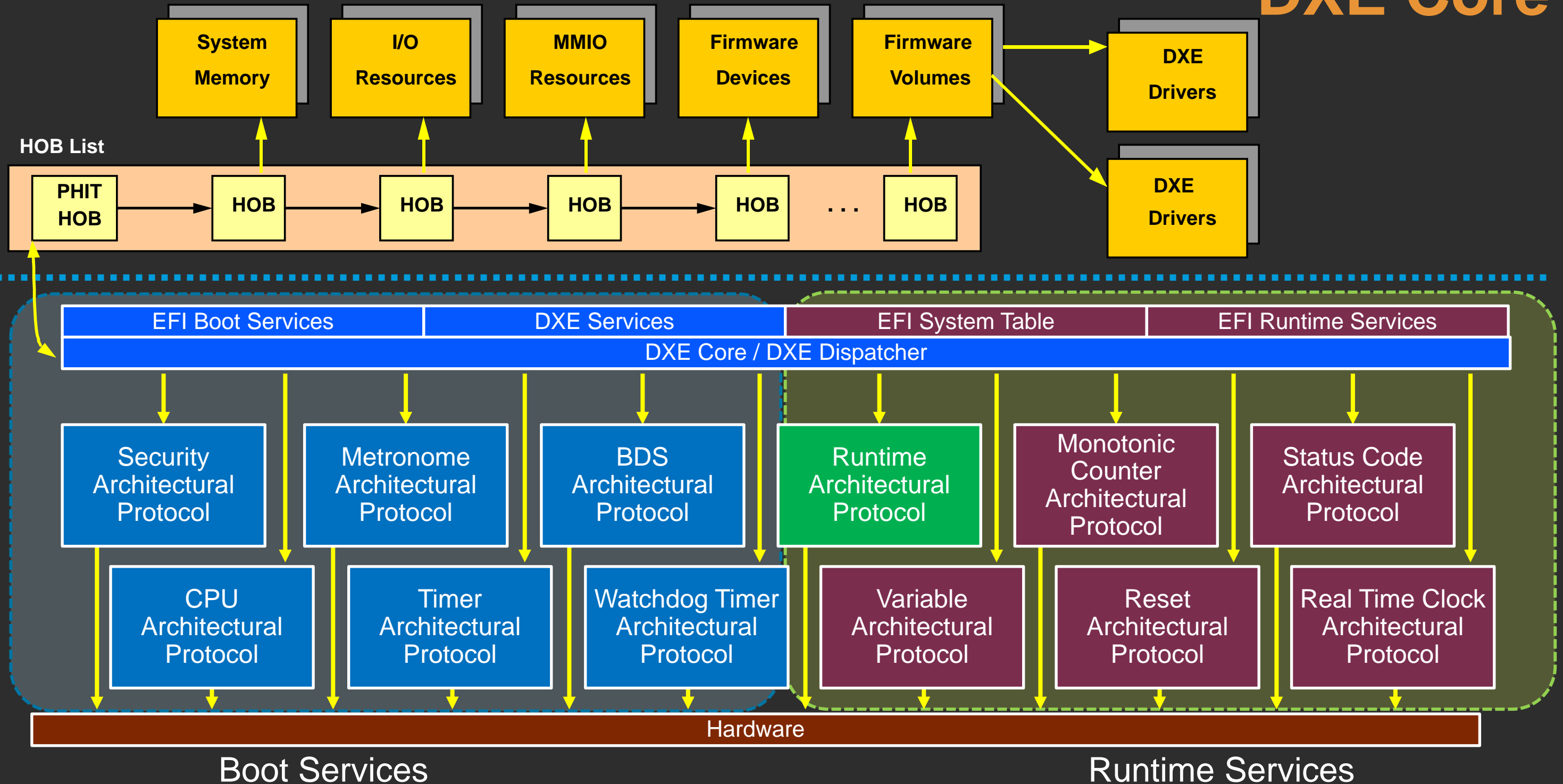


System Reset Vector  
Stage 7 on IA

# PEI Phase Flow







# EDK II LIBRARY MODULES

## *Syntax:*

```
[LibraryClasses.common]  
  <LibraryClassName> | <LibraryInstancePathToInf/Name.inf>  
  
DebugLib | MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf
```

**Name**

**Implementation<sup>3</sup>**

**Consistent set of interfaces**

**Does not describe implementation of the interfaces**

# “NULL” Library Class

## Constructors

## Special Cases

NOT “. . . LibNull” instance

### Syntax

```
Pkg/MyModule/MyModule.inf {  
  <LibraryClasses>  
    NULL|Pkg/Library/LibName/LibName.inf  
    NULL|Pkg/Library/LibName2/LibName2.inf  
}
```

## Open Source Example

DxeCrc32GuidedSectionExtractLib  
ShellPkg as used with Profiles

### UEFI Shell example:

```
ShellPkg/Application/Shell/Shell.inf {  
  <LibraryClasses>  
    NULL|ShellPkg/Library/UefiShellDriver1CommandsLib/UefiShellDriver1CommandsLib.inf  
    NULL|ShellPkg/Library/UefiShellNetwork1CommandsLib/UefiShellNetwork1CommandsLib.inf  
    . . .  
}
```

# Locating Library Classes

Library based upon

1. Industry specs (UEFI, etc.)

**MdePkg/MdeModulePkg**

2. Features

**NetworkPkg/SecurityPkg**

Use the package help files (.CHM) to find a library or function

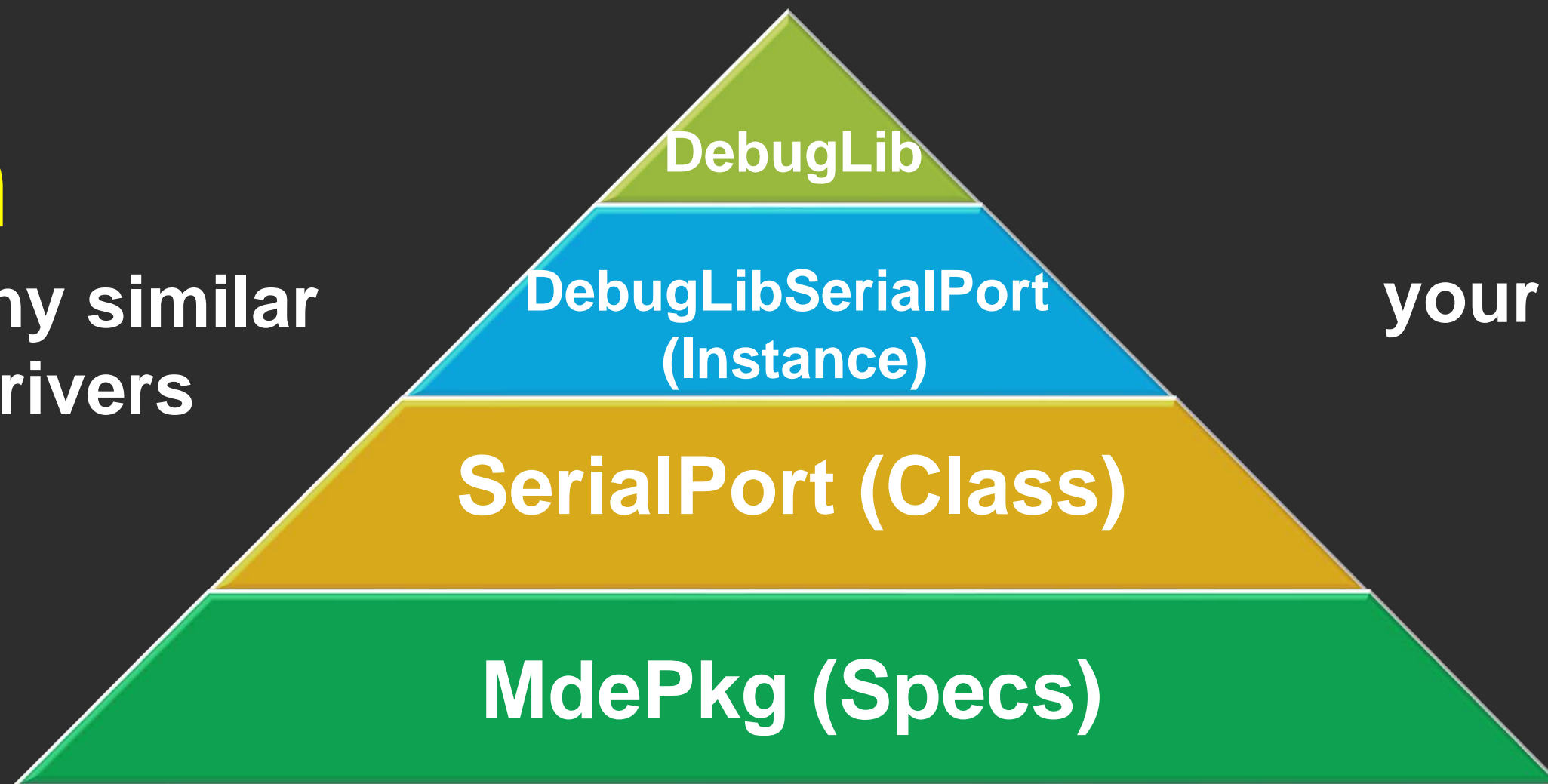
*Example: MdePkg.chm*

Search WorkSpace (.INF) "LIBRARY\_CLASS"

# Library Instance Hierarchy

## Form

a hierarchy similar  
to UEFI drivers



## Link

your module to  
another

**Build error** : Instance of Library class [*Foo...Lib*] is not found ....  
Consumed by module [*My Module.inf*]

# Commonly Used Base Library Classes

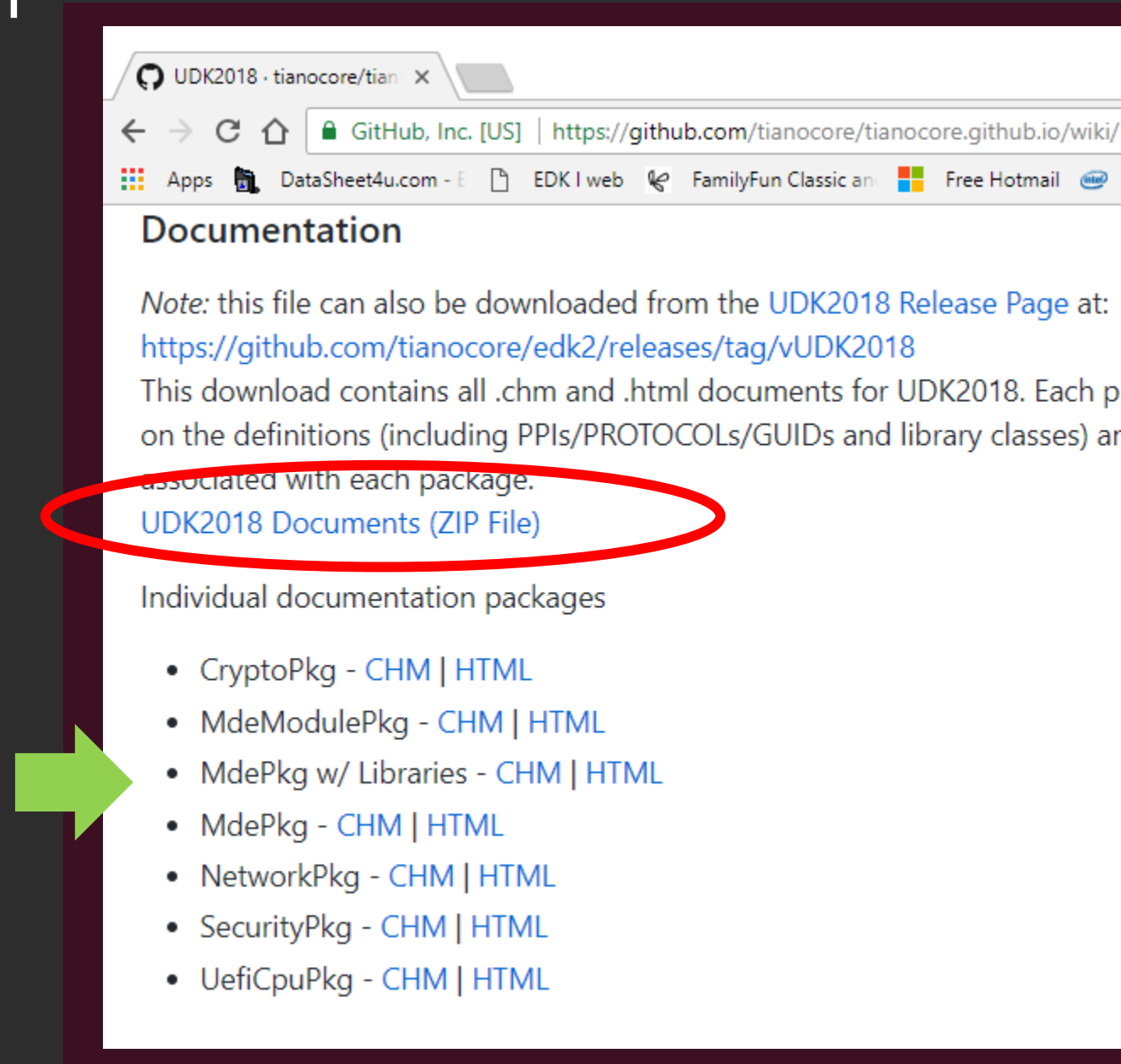
BaseLib    DebugLib    UefiDriverEntryPoint  
UefiLib    UefiBootServicesTableLib  
UefiApplicationEntryPoint    DxeCoreEntryPoint  
CpuLib    UefiUsbLib    PciLib    DevicePathLib    IoLib  
MemoryAllocationLib    PrintLib    PeimEntryPoint  
PeiCoreEntryPoint    UefiScsiLib    BaseMemoryLib  
UefiRuntimeLib    SmmMemLib  
DxeServicesLib    SynchronizationLib    PciExpressLib  
DxePcdLib    UefiRuntimeServicesTableLib  
PciSegmentLibLib    PeiServicesLib  
PeiPcdLib    DxeHobLib    UefiFileHandleLib

# MdePkg Library .CHM file Location

[tianocore.org](https://tianocore.org) UDK2018 documentation on

 [Latest UDK Release](#)

 [UDK2018](#)



UDK2018 · tianocore/tianocore.github.io/wiki/

← → ↻ 🏠 🔒 GitHub, Inc. [US] | <https://github.com/tianocore/tianocore.github.io/wiki/>

Apps 📁 DataSheet4u.com - E 📄 EDK I web 🔗 FamilyFun Classic an 🌐 Free Hotmail 🍏

## Documentation

Note: this file can also be downloaded from the [UDK2018 Release Page](#) at: <https://github.com/tianocore/edk2/releases/tag/vUDK2018>

This download contains all .chm and .html documents for UDK2018. Each package contains documentation on the definitions (including PPIs/PROTOCOLs/GUIDs and library classes) and the associated with each package.

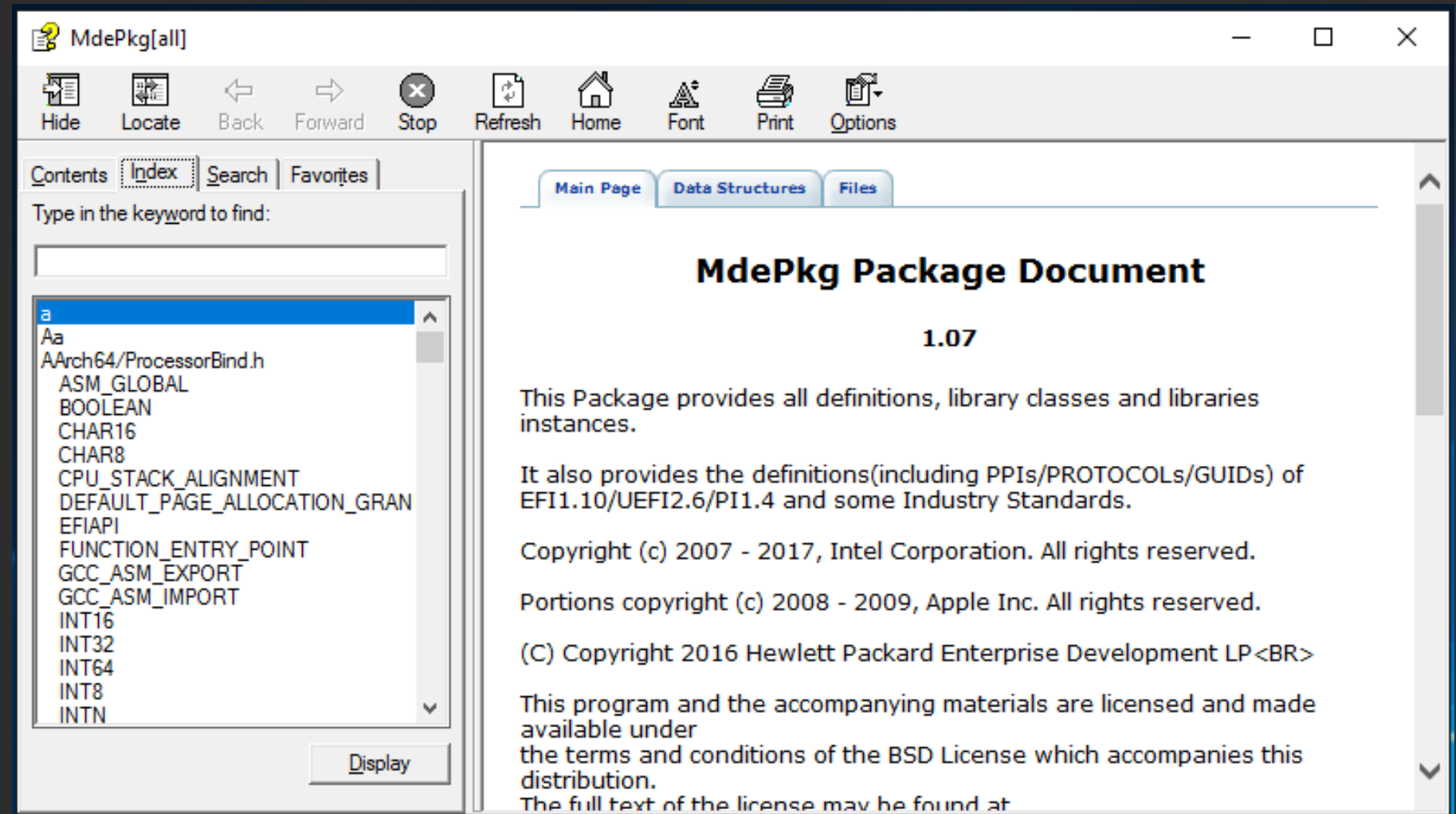
[UDK2018 Documents \(ZIP File\)](#)

### Individual documentation packages

- CryptoPkg - [CHM](#) | [HTML](#)
- MdeModulePkg - [CHM](#) | [HTML](#)
- MdePkg w/ Libraries - [CHM](#) | [HTML](#)
- MdePkg - [CHM](#) | [HTML](#)
- NetworkPkg - [CHM](#) | [HTML](#)
- SecurityPkg - [CHM](#) | [HTML](#)
- UefiCpuPkg - [CHM](#) | [HTML](#)



# Library Navigation Demonstration

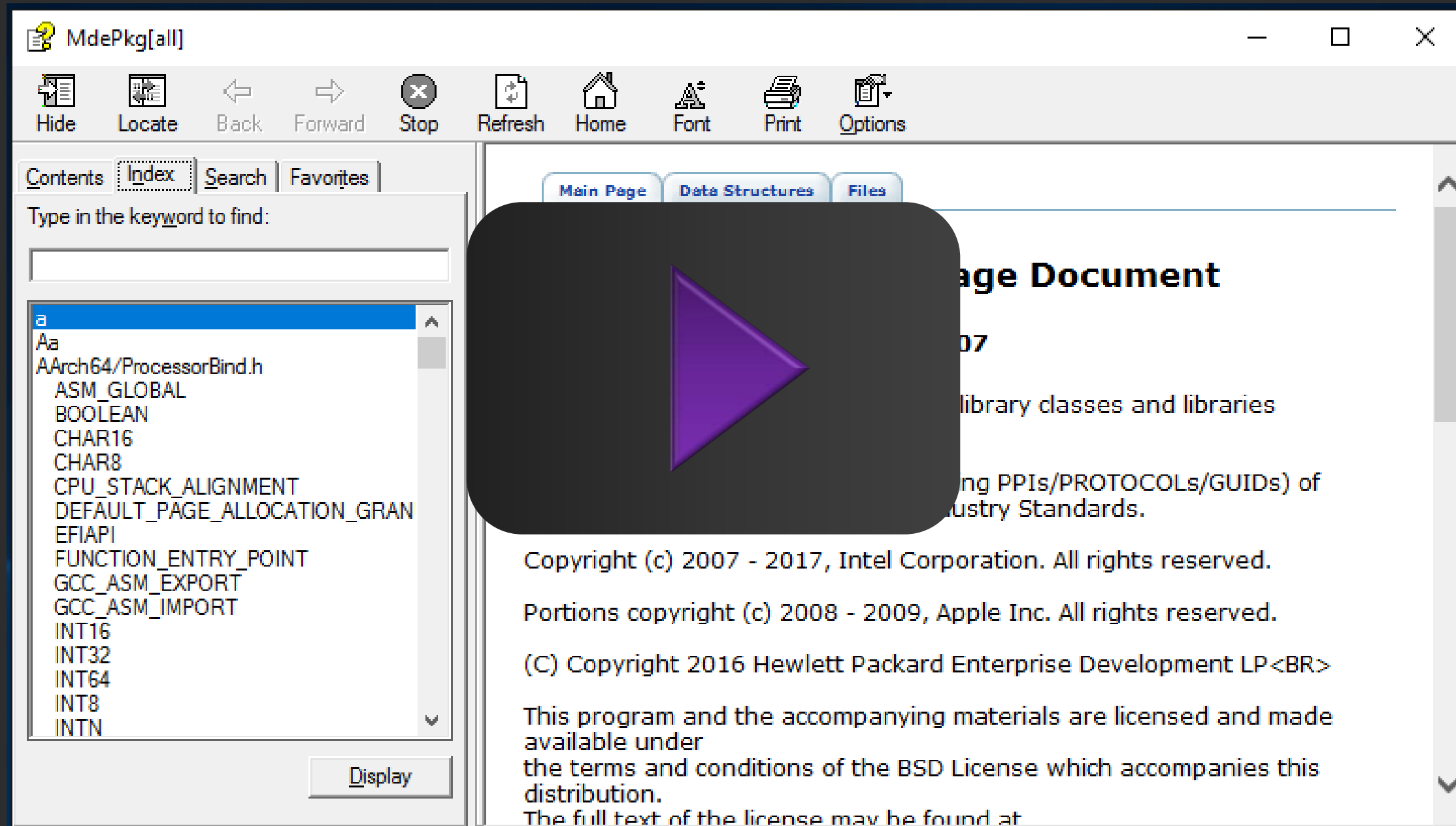


Open file: /FW/Documentation/"MdePkg Document With LibrariesMdePkg.chm"

**NOTE:** Install a CHM Viewer for Ubuntu

```
bash$ sudo aptitude install kchmviewer
```

# Library Navigation Demonstration



<https://youtu.be/s8Zw1w1iQS4>

# EDK II UEFI APPLICATION

# Defining a UEFI Application

## Characteristics of a UEFI Application

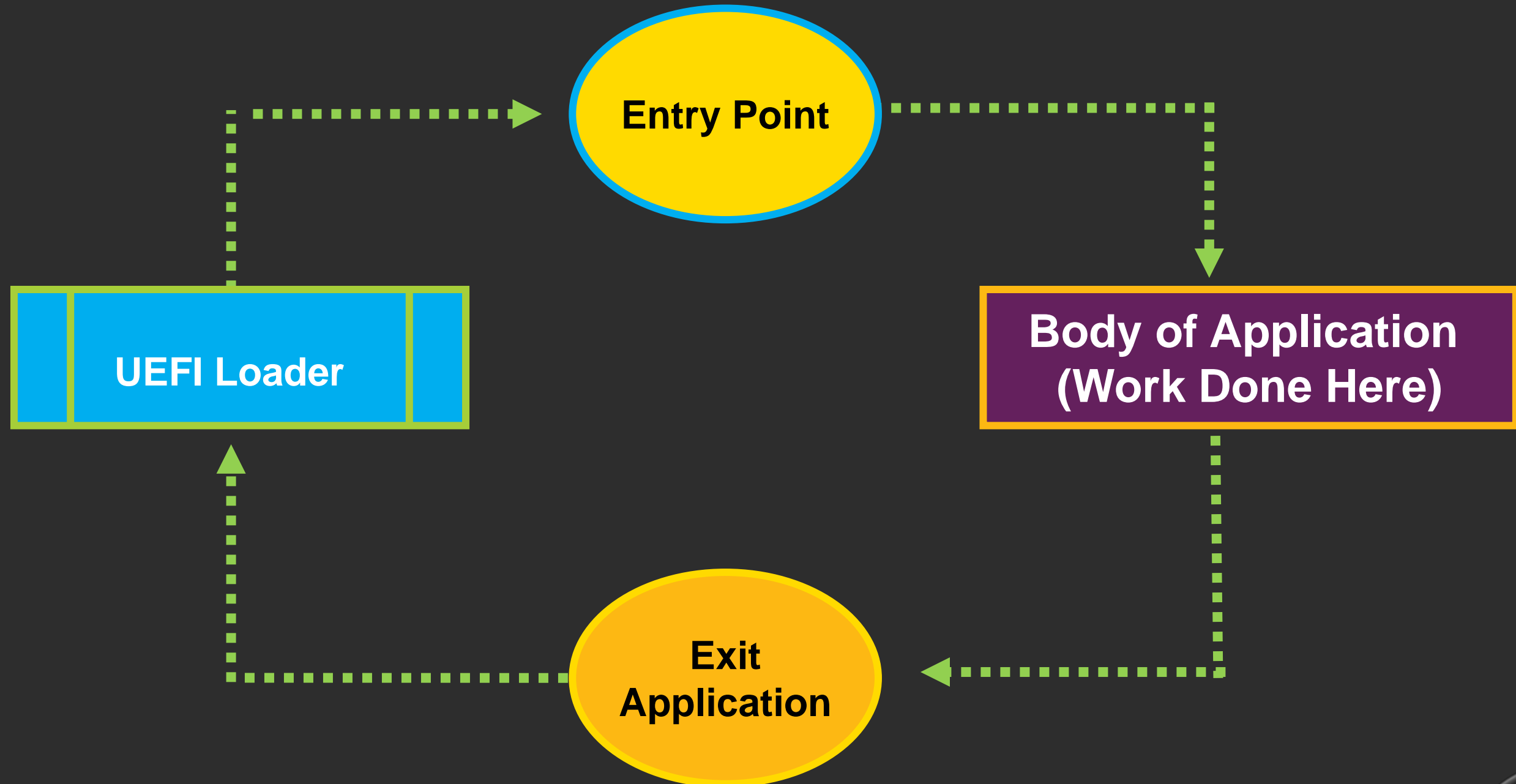
- ★ Loaded by UEFI loader, just like drivers
- ★ Does not register protocols
- ★ Consumes protocols
- ★ Typically exits when completed (user driven )
- ★ Same set of interfaces as drivers available

# Defining a UEFI Application

## UEFI Application Usages

- ✱ Platform Diagnostics
- ✱ Factory Diagnostics
- ✱ Utilities
- ✱ Driver Prototyping
- ✱ “Platform” Applications
- ✱ Portable Across Platforms (IA32, X64, ARM, Itanium, etc.)

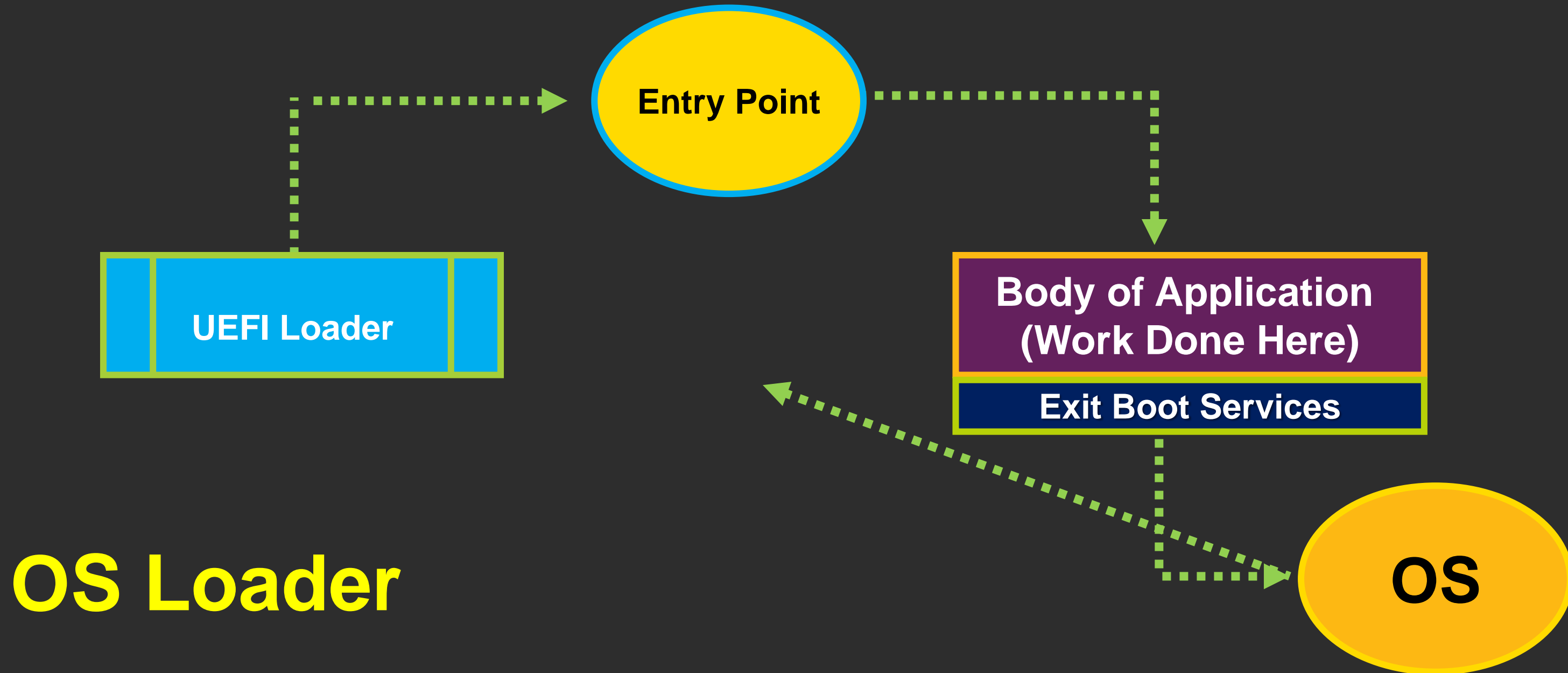
# Executing Applications



# Executing Applications



# Executing Applications





# Executing Applications



OS Loader

# Driver Vs. Application

|                       | Driver           | Application |
|-----------------------|------------------|-------------|
| Loaded by:            | UEFI Loader      | UEFI Loader |
| Interfaces available: | ALL              | ALL         |
| Consume protocols?    | YES              | YES         |
| Produce protocols?    | YES              | NO          |
| Typically driven by?  | System           | User        |
| Typical use           | Support Hardware | Any         |

# EDK II UEFI Applications

## How to Write a EDK II UEFI Application


# Application Files Placement

- Application source files can be located anywhere in the EDK II workspace including PACKAGES\_PATH
- All code and include files go under a single directory containing the driver INF
- EDK II Sample Applications can be found here:

`edk2/MdeModulePkg/Application`

- Typically, modules reside within a package:

```
MyWorkspace/  
  edk2/  
    MyPkg/  
      Application/  
        MyApp/
```



`MyApp.c`  
`MyApp.inf`

## Syntax

```
INFfile ::= [<Header>  
            <Defines>  
            [<BuildOptions>]  
            [<Sources>]  
            [<Binaries>]  
            [<Guids>]  
            [<Protocols>]  
            [<Ppis>]  
            [<Packages>]  
            [<LibraryClasses>]  
            [<Pcds>]  
            [<UserExtensions>]  
            [<Depex>]
```

PreMake

INF text file example

# Application INF Files [DEFINES]

| Field                  | Description   |
|------------------------|---|
| <b>INF_VERSION</b>     | 1.25* - Version of the INF spec.                    |
| <b>BASE_NAME</b>       | What's the name of the application                  |
| <b>FILE_GUID</b>       | Create a GUID for your module                       |
| <b>MODULE_UNI_FILE</b> | Meta-data - localization for Description & Abstract |
| <b>VERSION_STRING</b>  | Version number                                      |
| <b>ENTRY_POINT</b>     | Name of the function to call                        |
| <b>MODULE_TYPE</b>     | UEFI_APPLICATION                                    |

\* EDK II Specifications: <https://github.com/tianocore/tianocore.github.io/wiki/EDK-II-Specifications>

# Sample INF file

```
[Defines]
  INF_VERSION                = 0x00010005
  BASE_NAME                  = MyApplication
  MODULE_UNI_FILE            = MyFile.uni
  FILE_GUID                  = 10C75C00-30 . . .
  MODULE_TYPE                = UEFI_APPLICATION
  VERSION_STRING             = 1.0
  ENTRY_POINT                = UefiMain
[Sources]
  MyFile.c

[Packages]
  MdePkg/MdePkg.dec

[LibraryClasses]
  UefiApplicationEntryPoint

[Guids]

[Ppis]

[Protocols]
```

# Sample INF file

```
[Defines]
  INF_VERSION                = 0x00010005
  BASE_NAME                  = MyApplication
  MODULE_UNI_FILE            = MyFile.uni
  FILE_GUID                  = 10C75C00-30 . . .
  MODULE_TYPE                = UEFI_APPLICATION
  VERSION_STRING             = 1.0
  ENTRY_POINT                = UefiMain
```

```
[Sources]
  MyFile.c
```

```
[Packages]
  MdePkg/MdePkg.dec
```

```
[LibraryClasses]
  UefiApplicationEntryPoint
```

```
[Guids]
```

```
[Ppis]
```



# Building an Application

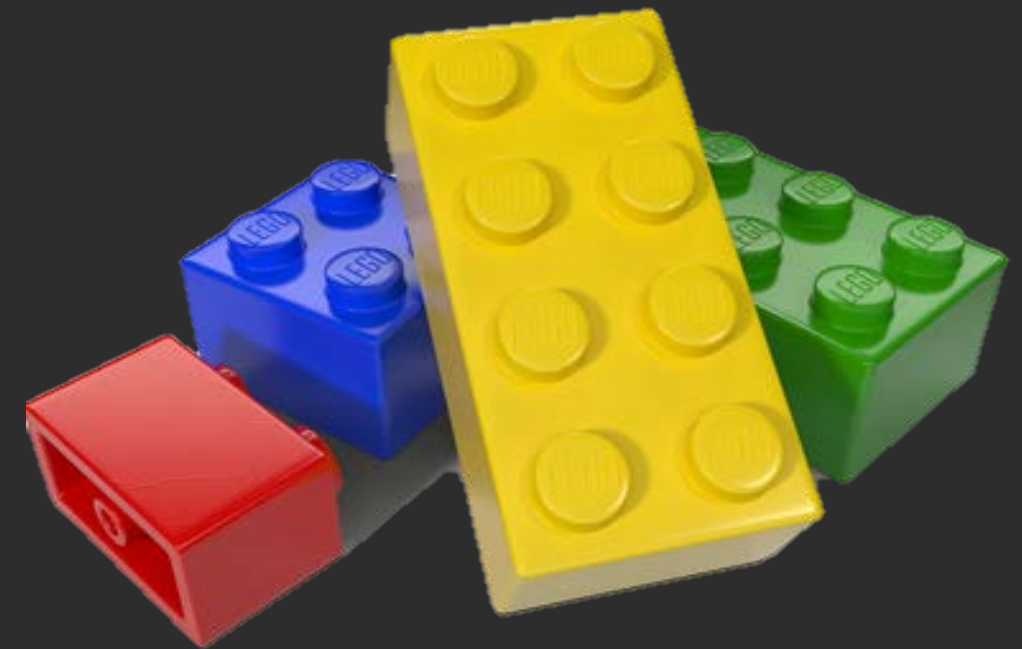
Platform .DSC references .INF

Runs:

“Build” for the entire platform

OR

“Build” in the application’s directory



# Sample Application 'C' file

```
#include <Uefi.h>
#include <Library/UefiApplicationEntryPoint.h>

EFI_STATUS
EFIAPI
UefiMain (
    IN EFI_HANDLE        ImageHandle,
    IN EFI_SYSTEM_TABLE  *SystemTable
)
{
    return EFI_SUCCESS;
}
```

# Sample Application 'C' file

```
#include <Uefi.h>
#include <Library/UefiApplicationEntryPoint.h>
```

```
EFI_STATUS
```

```
EFIAPI
```

```
UefiMain (
```

```
    IN EFI_HANDLE
```

```
    IN EFI_SYSTEM_TABLE
```

```
)
```

```
{
```

```
    return EFI_SUCCESS;
```

```
}
```

```
    ImageHandle,  
    *SystemTable
```

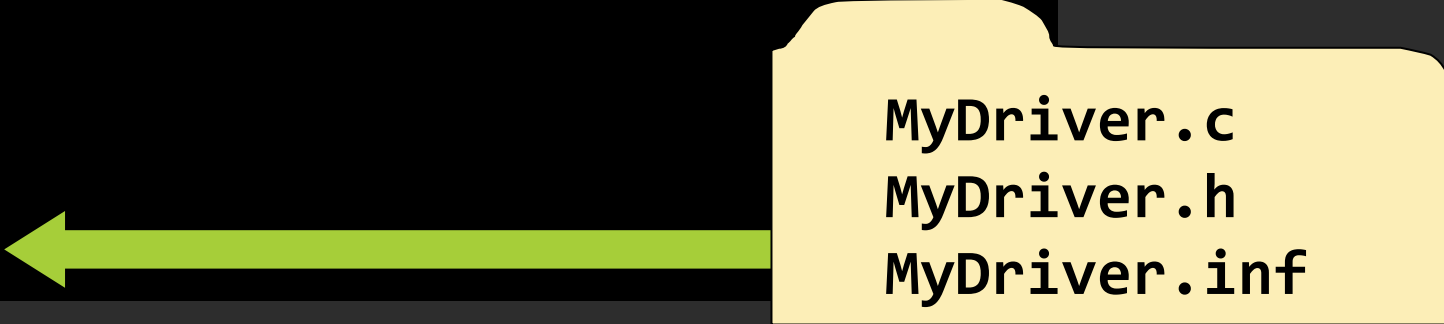
# EDK II UEFI DRIVERS

How to Write DXE Drivers, PEIM, Etc.

# Driver Files Placement

- ★ Driver source code can go anywhere in the EDK II workspace
- ★ All code and include files go under a single directory containing an INF
- ★ Good example of UEFI Drivers can be found here:  
`edk2/MdeModulePkg/Bus/ScsiDiskDxe`
- ★ Typically, Driver modules reside within a package:

```
MyWorkspace/  
  edk2/  
    MyPkg/  
      Include/  
        MyDriver/
```



```
MyDriver.c  
MyDriver.h  
MyDriver.inf
```

# Driver INF Files: [DEFINES]

| Field                  | Description   |
|------------------------|---|
| <b>INF_VERSION</b>     | 1.25* - Version of the INF spec.                    |
| <b>BASE_NAME</b>       | What's the name of the driver                       |
| <b>FILE_GUID</b>       | Create a GUID for your module                       |
| <b>MODULE_UNI_FILE</b> | Meta-data - localization for Description & Abstract |
| <b>VERSION_STRING</b>  | Version number                                      |
| <b>ENTRY_POINT</b>     | Name of the function to call                        |
| <b>MODULE_TYPE</b>     | UEFI_DRIVER, DXE_DRIVER, PEIM, or others            |

# Changes for a UEFI Driver Module

Applications can be converted to a driver

**But** ...It remains in memory after it runs

UEFI Driver Module requirements:

- Driver Binding Protocol
- Component Name2 Protocol (recommended

DXE/PEIM/other Driver requirements



# Sample Driver INF file

```
[Defines]
  INF_VERSION              = 0x00010005
  BASE_NAME                = MyDriver
  FILE_GUID                = 10C75C00-30...
  MODULE_TYPE              = UEFI_DRIVER
  VERSION_STRING           = 1.0
  ENTRY_POINT              = UefiMain
```

```
[Sources]
  MyDriverFile.c
```

```
[Packages]
  MdePkg/MdePkg.dec
```

```
[LibraryClasses]
  UefiDriverEntryPoint
  . . .
```

```
[Guids]
  . . .
[Protocols]
  . . .
```



# INF Usage Fields – DIST files

## *Optional* UEFI Spec – Package Distribution

### Usage Key Word

## UNDEFINED

## CONSUMES

## SOMETIMES\_CONSUMES

## PRODUCES

## SOMETIMES\_PRODUCES

## TO\_START

## BY\_START

## NOTIFY

Usage Fields used by Build tools for creating the .Dist files for binary modules

[GUID]

[PCD]

[PROTOCOL]

[PPIS]

1 Usage Block – “##” After the entry

*n* Usage Blocks – “##” Precede the entry



} UEFI Protocol

# INF File Usage Block examples

[Guids]

```
## SOMETIMES_PRODUCES ## Variable:L"ConInDev"  
## SOMETIMES_CONSUMES ## Variable:L"ConInDev"  
## SOMETIMES_PRODUCES ## Variable:L"ConOutDev"  
## SOMETIMES_CONSUMES ## Variable:L"ConOutDev"  
## SOMETIMES_PRODUCES ## Variable:L"ErrOutDev"  
## SOMETIMES_CONSUMES ## Variable:L"ErrOutDev"
```

gEfiGlobalVariableGuid

```
gEfiVTUTF8Guid ## SOMETIMES_CONSUMES ## GUID # used with a Vendor-Defined  
gEfiVT100Guid ## SOMETIMES_CONSUMES ## GUID # used with a Vendor-Defined  
gEfiVT100PlusGuid ## SOMETIMES_CONSUMES ## GUID # used with a Vendor-Defined  
gEfiPcAnsiGuid ## SOMETIMES_CONSUMES ## GUID # used with a Vendor-Defined  
gEfiTtyTermGuid ## SOMETIMES_CONSUMES ## GUID # used with a Vendor-Defined  
gEdkiiStatusCodeDataTypeVariableGuid ## SOMETIMES_CONSUMES ## GUID
```

Example: [TerminalDxe.inf](#)

# INF File Usage Block examples

[Protocols]

gEfiSerialIoProtocolGuid                      ## TO\_START

## BY\_START

## TO\_START

gEfiDevicePathProtocolGuid

gEfiSimpleTextInProtocolGuid                ## BY\_START

gEfiSimpleTextInputExProtocolGuid          ## BY\_START

gEfiSimpleTextOutProtocolGuid               ## BY\_START

[Pcd]

gEfiMdePkgTokenSpaceGuid.PcdDefaultTerminalType                ## SOMETIMES\_CONSUMES

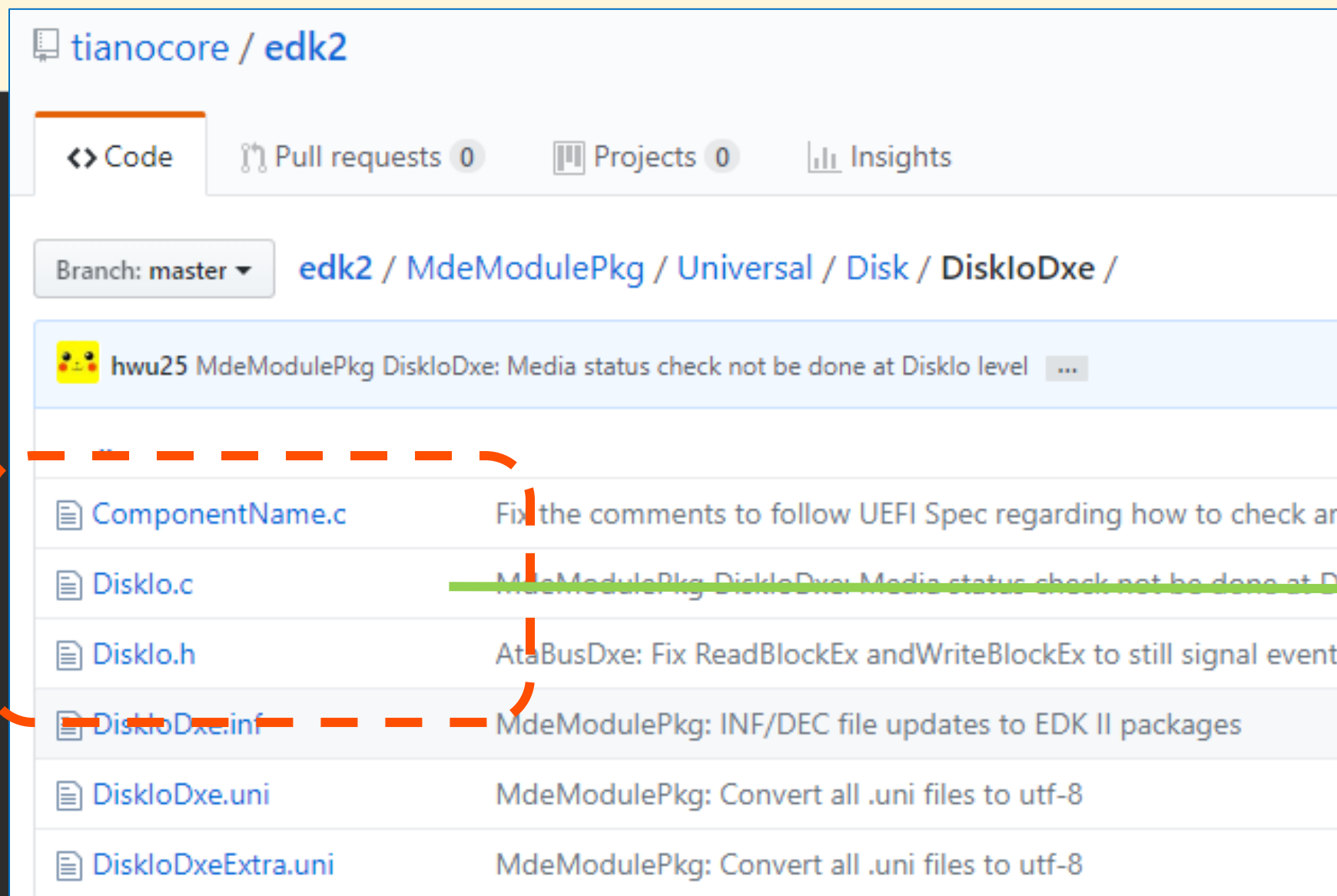
gEfiMdeModulePkgTokenSpaceGuid.PcdErrorCodeSetVariable        ## CONSUMES

Example: TerminalDxe.inf

# UEFI Driver Example – Disk I/O



<https://github.com/tianocore/edk2/MdeModulePkg/Universal/Disk/DiskIoDxe>



tianocore / edk2

Code Pull requests 0 Projects 0 Insights

Branch: master edk2 / MdeModulePkg / Universal / Disk / DiskIoDxe /

hwu25 MdeModulePkg DiskIoDxe: Media status check not be done at DiskIo level ...

|                    |  |
|--------------------|--|
| ComponentName.c    | Fix the comments to follow UEFI Spec regarding how to check an   |
| DiskIo.c           | MdeModulePkg DiskIoDxe: Media status check not be done at Di     |
| DiskIo.h           | AtaBusDxe: Fix ReadBlockEx andWriteBlockEx to still signal event |
| DiskIoDxe.inf      | MdeModulePkg: INF/DEC file updates to EDK II packages            |
| DiskIoDxe.uni      | MdeModulePkg: Convert all .uni files to utf-8                    |
| DiskIoDxeExtra.uni | MdeModulePkg: Convert all .uni files to utf-8                    |

Driver Binding  
Supported  
Start  
Stop

# UEFI Driver Example – Disk I/O

 <https://github.com/tianocore/edk2/.../Disk/DiskIoDxe>

## Entry Point

### “C” File

```
EFI_STATUS
EFIAPR
InitializeDiskIo (
    IN EFI_HANDLE      ImageHandle,
    IN EFI_SYSTEM_TABLE *SystemTable
)
{
    // ...
    Status = EfiLibInstallDriverBindingComponentName2 (
        ImageHandle,
        SystemTable,
        &gDiskIoDriverBinding,
        ImageHandle,
        &gDiskIoComponentName,
        &gDiskIoComponentName2
    );
    ASSERT_EFI_ERROR (Status);
    return Status;
}
```

### INF File

[Defines]

```
ENTRY_POINT = InitializeDiskIo
```

# UEFI Driver Example – Disk I/O

 <https://github.com/tianocore/edk2/.../Disk/DiskIoDxe>

## Supported

### “C” File

```
EFI_STATUS
EFI_API
DiskIoDriverBindingSupported (
    IN EFI_DRIVER_BINDING_PROTOCOL *This,
    IN EFI_HANDLE ControllerHandle,
    IN EFI_DEVICE_PATH_PROTOCOL *RemainingDevicePath,
    OPTIONAL
    )
{
    Status = gBS->OpenProtocol (
        ControllerHandle,
        &gEfiBlockIoProtocolGuid,
        (VOID **) &BlockIo,
        This->DriverBindingHandle,
        ControllerHandle,
        EFI_OPEN_PROTOCOL_BY_DRIVER
    );
}
```

### INF File

```
[Protocols]
gEfiBlockIoProtocolGuid ## TO_START
```

# UEFI Driver Example – Disk I/O

 <https://github.com/tianocore/edk2/.../Disk/DiskIoDxe>

## Start

### “C” File

```
EFI_STATUS
EFI_API
DiskIoDriverBindingStart (
    IN EFI_DRIVER_BINDING_PROTOCOL *This,
    IN EFI_HANDLE                  ControllerHandle,
    IN EFI_DEVICE_PATH_PROTOCOL   *RemainingDevicePath
    OPTIONAL
)
{
    if (Instance->BlockIo2 != NULL) {
        Status = gBS->InstallMultipleProtocolInterfaces (
            &ControllerHandle,
            &gEfiDiskIoProtocolGuid, &Instance->DiskIo,
            &gEfiDiskIo2ProtocolGuid, &Instance->DiskIo2,
            NULL
        );
    }
}
```

### INF File

[Protocols]

```
gEfiDiskIoProtocolGuid  ## BY_START
gEfiDiskIo2ProtocolGuid ## BY_START
```

# DXE Driver Example - PlatformInfoDxe



<https://github.com/tianocore/edk2-platforms/. . ./Vlv2TbItDevicePkg/PlatformInfoDxe>

tianocore / edk2-platforms

Code Pull requests 0 Projects 0 Security Insights

Branch: master edk2-platforms / Platform / Intel / Vlv2TbItDevicePkg / PlatformInfoDxe /

mdkinney Platform/Vlv2TbItDevicePkg: Import Vlv2TbItDevicePkg from edk2

|                     |  |
|---------------------|--|
| PlatformInfoDxe.c   | Platform/Vlv2TbItDevicePkg: Import Vlv2TbItDevicePkg from edk2 |
| PlatformInfoDxe.h   | Platform/Vlv2TbItDevicePkg: Import Vlv2TbItDevicePkg from edk2 |
| PlatformInfoDxe.inf | Platform/Vlv2TbItDevicePkg: Import Vlv2TbItDevicePkg from edk2 |



# DXE Driver Example – PlatformInfoDxe

 <https://github.com/tianocore/edk2-platforms/PlatformInfoDxe>

## Entry Point

### “C” File

```
#include "PlatformInfoDxe.h"
...
EFI_STATUS
EFIAPI
PlatformInfoInit (
    IN EFI_HANDLE      ImageHandle,
    IN EFI_SYSTEM_TABLE *SystemTable
)
/*++
{
// ...
return Status;
}
```

### INF File

```
[Defines]
...
MODULE_TYPE           = DXE_DRIVER
VERSION_STRING        = 1.0
ENTRY_POINT           = PlatformInfoInit
...

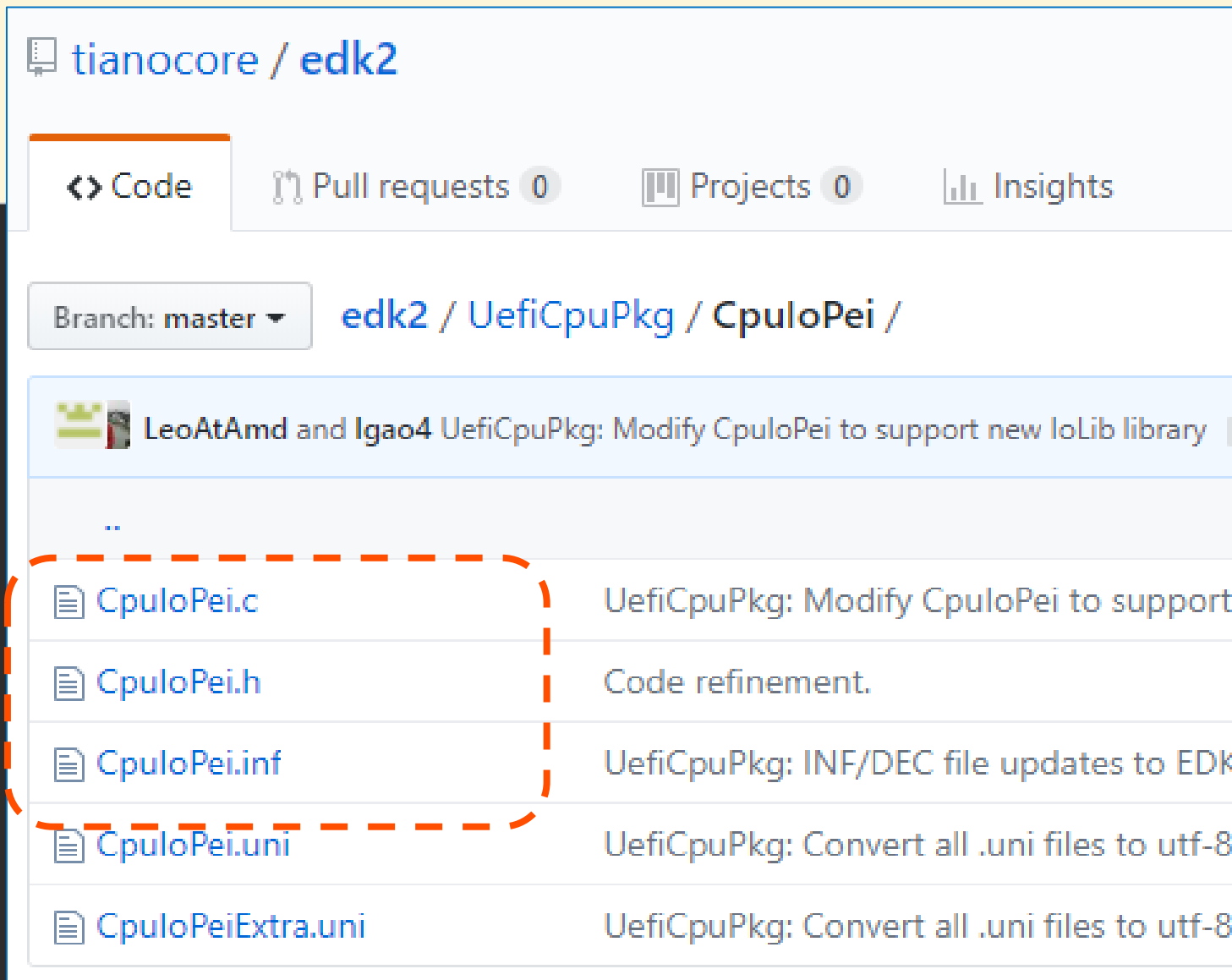
[Depex]
gEfiVariableArchProtocolGuid AND
gEfiVariableWriteArchProtocolGuid
```

Notice the MODULE TYPE, C function Entry point and the [Depex] differences in the INF file

# PEI Driver (PEIM) Example - CpuloPei



<https://github.com/tianocore/edk2/UefiCpuPkg/CpuloPei>



tianocore / edk2

Code Pull requests 0 Projects 0 Insights

Branch: master edk2 / UefiCpuPkg / CpuloPei /

LeoAtAmd and Igao4 UefiCpuPkg: Modify CpuloPei to support new IoLib library

..

|                   |   |
|-------------------|---|
| CpuloPei.c        | UefiCpuPkg: Modify CpuloPei to support      |
| CpuloPei.h        | Code refinement.                            |
| CpuloPei.inf      | UefiCpuPkg: INF/DEC file updates to EDK     |
| CpuloPei.uni      | UefiCpuPkg: Convert all .uni files to utf-8 |
| CpuloPeiExtra.uni | UefiCpuPkg: Convert all .uni files to utf-8 |

# PEI Driver (PEIM) Example – CpuIoPei

## Entry Point

 <https://github.com/tianocore/edk2/UefiCpuPkg/CpuIoPei>

### “C” File

```
#include "CpuIoPei.h"
// . . .
EFI_STATUS
EFIAPI
CpuIoInitialize (
    IN EFI_PEI_FILE_HANDLE FileHandle,
    IN CONST EFI_PEI_SERVICES **PeiServices
)
{
    EFI_STATUS Status;
    // . . .
    return EFI_SUCCESS;
}
```

### INF File

[Defines]

. . .

|                |                   |
|----------------|-------------------|
| MODULE_TYPE    | = PEIM            |
| VERSION_STRING | = 1.0             |
| ENTRY_POINT    | = CpuIoInitialize |

. . .

[Depex]  
TRUE

# SUMMARY

- What is a EDK II Module
- Use EDK II libraries to write UEFI apps/drivers
- How to Define a UEFI application
- Differences between UEFI App / Drivers INF file

# Questions?



# Return to Main Training Page



Return to Training Table of contents for next presentation [link](#)





# ACKNOWLEDGEMENTS

Redistribution and use in source (original document form) and 'compiled' forms (converted to PDF, epub, HTML and other formats) with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code (original document form) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.

Redistributions in compiled form (transformed to other DTDs, converted to PDF, epub, HTML and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS DOCUMENTATION IS PROVIDED BY TIANOCORE PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL TIANOCORE PROJECT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 2021-2022, Intel Corporation. All rights reserved.

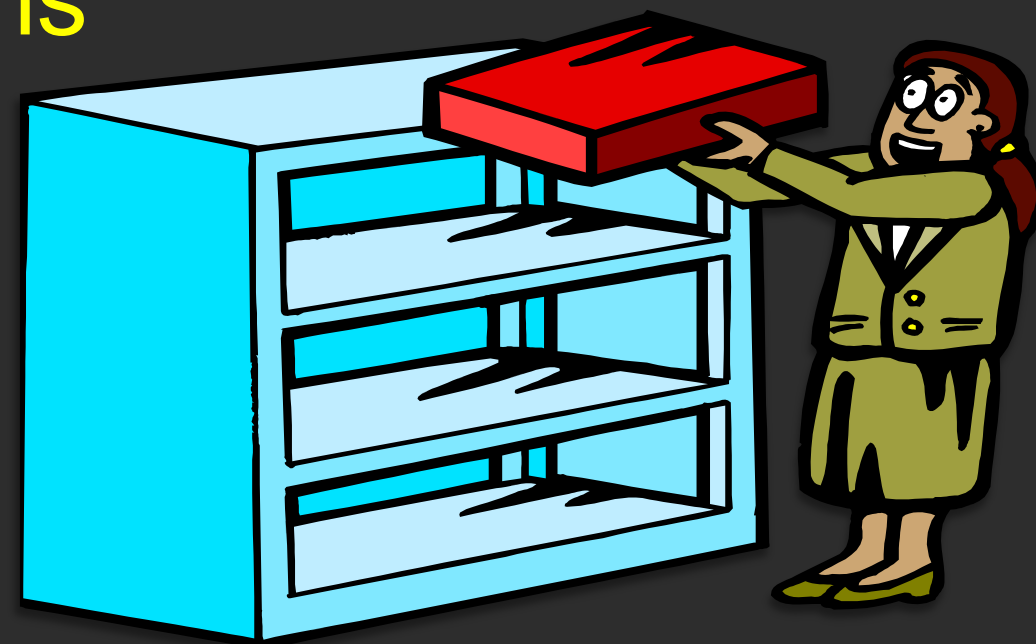


**BACK UP**

# UEFI Application Vs. EADK Application

EDK II Application Development  
Kit includes the Standard C  
Libraries in UEFI Shell Applications

*Off the shelf* “C” application  
Converted to UEFI application



# Sample INF file using EDK II EADK

```
[Defines]
  INF_VERSION           = 0x00010005
  BASE_NAME             = MyApplication
  FILE_GUID             = 10C75C00-30 . . .
  MODULE_TYPE           = UEFI_APPLICATION
  VERSION_STRING        = 1.0
  ENTRY_POINT           = ShellCEntryLib
[Sources]
  MyFile.c

[Packages]
  StdLib/StdLib.dec
  ShellPkg/ShellPkg.dec
  MdePkg/MdePkg.dec

[LibraryClasses]
  LibC
  LibStdio
```

# Sample INF file using EDK II EADK

```
[Defines]
  INF_VERSION           = 0x00010005
  BASE_NAME             = MyApplication
  FILE_GUID             = 10C75C00-30 . . .
  MODULE_TYPE           = UEFI_APPLICATION
  VERSION_STRING        = 1.0
  ENTRY_POINT           = ShellCEntryLib
```

```
[Sources]
  MyFile.c
```

```
[Packages]
  StdLib/StdLib.dec
  ShellPkg/ShellPkg.dec
  MdePkg/MdePkg.dec
```

```
[LibraryClasses]
  LibC
  LibStdio
```

# Sample Application 'C' file Using EDK II EADK

This sample looks a lot like actual “C” source.

```
#include <stdio.h>
```

```
int
```

```
Main
```

```
    int Argc,
```

```
    char **Argv
```

```
    )
```

```
{
```

```
    return 0
```

```
}
```

# More on “NULL” named Library

The “NULL” library classes are conceptually an “anonymous library”. It enables one to statically link code into a module even if the module doesn't directly call functions in that library. All libraries, both regular libraries with a declared LibraryClass as well as these anonymous libraries, can publish both a constructor and a destructor. The EDK II build system will automatically generate a small amount of C code that invokes all library constructors before the entry point for the module is invoked, and all destructors after the entry point returns. This is useful for building statically linked plug-ins.

The favorite example of this in-action is the UEFI Shell, here is what a typical DSC declaration for the UEFI Shell looks like:

```
ShellPkg/Application/Shell/Shell.inf {
  <PcdsFixedAtBuild>
    gEfiShellPkgTokenSpaceGuid.PcdShellLibAutoInitialize|FALSE
  <LibraryClasses>
    NULL|ShellPkg/Library/UefiShellLevel1CommandsLib/UefiShellLevel1CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellLevel2CommandsLib/UefiShellLevel2CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellLevel3CommandsLib/UefiShellLevel3CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellDriver1CommandsLib/UefiShellDriver1CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellInstall1CommandsLib/UefiShellInstall1CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellDebug1CommandsLib/UefiShellDebug1CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellNetwork1CommandsLib/UefiShellNetwork1CommandsLib.inf
    NULL|ShellPkg/Library/UefiShellNetwork2CommandsLib/UefiShellNetwork2CommandsLib.inf
    ShellCommandLib|ShellPkg/Library/UefiShellCommandLib/UefiShellCommandLib.inf
    HandleParsingLib|ShellPkg/Library/UefiHandleParsingLib/UefiHandleParsingLib.inf
    BcfgCommandLib|ShellPkg/Library/UefiShellBcfgCommandLib/UefiShellBcfgCommandLib.inf
    ShellCEntryLib|ShellPkg/Library/UefiShellCEntryLib/UefiShellCEntryLib.inf
    ShellLib|ShellPkg/Library/UefiShellLib/UefiShellLib.inf
}
```

# More on “NULL” named Library Cont.

If you take a look at ShellPkg/Library/UefiShellLevel1CommandsLib/UefiShellLevel1CommandsLib.inf, you will see that the constructor for that anonymous library is named ShellLevel1CommandsLibConstructor(). Now, let's go and look at the definition for ShellLevel1CommandsLibConstructor() in ShellPkg/Library/UefiShellLevel1CommandsLib/UefiShellLevel1CommandsLib.c and note the following code snippet:

```
ShellCommandRegisterCommandName(L"stall", ShellCommandRunStall    ...
ShellCommandRegisterCommandName(L"for",  ShellCommandRunFor      ...
ShellCommandRegisterCommandName(L"goto",  ShellCommandRunGoto     ...
ShellCommandRegisterCommandName(L"if",     ShellCommandRunIf       ...
ShellCommandRegisterCommandName(L"shift",  ShellCommandRunShift    ...
ShellCommandRegisterCommandName(L"exit",   ShellCommandRunExit     ...
ShellCommandRegisterCommandName(L"else",   ShellCommandRunElse     ...
ShellCommandRegisterCommandName(L"endif",  ShellCommandRunEndIf    ...
ShellCommandRegisterCommandName(L"endfor", ShellCommandRunEndFor   ...
```

This library is installing new commands into the UEFI shell during its initialization procedure. This allows one to add custom commands to the shell as statically linked built-ins. A typical use case would be implementing platform specific diagnostic/recovery utilities.