



# UEFI & EDK II Training

## EDK II Modules: Libraries, Drivers & Applications

[tianocore.org](http://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

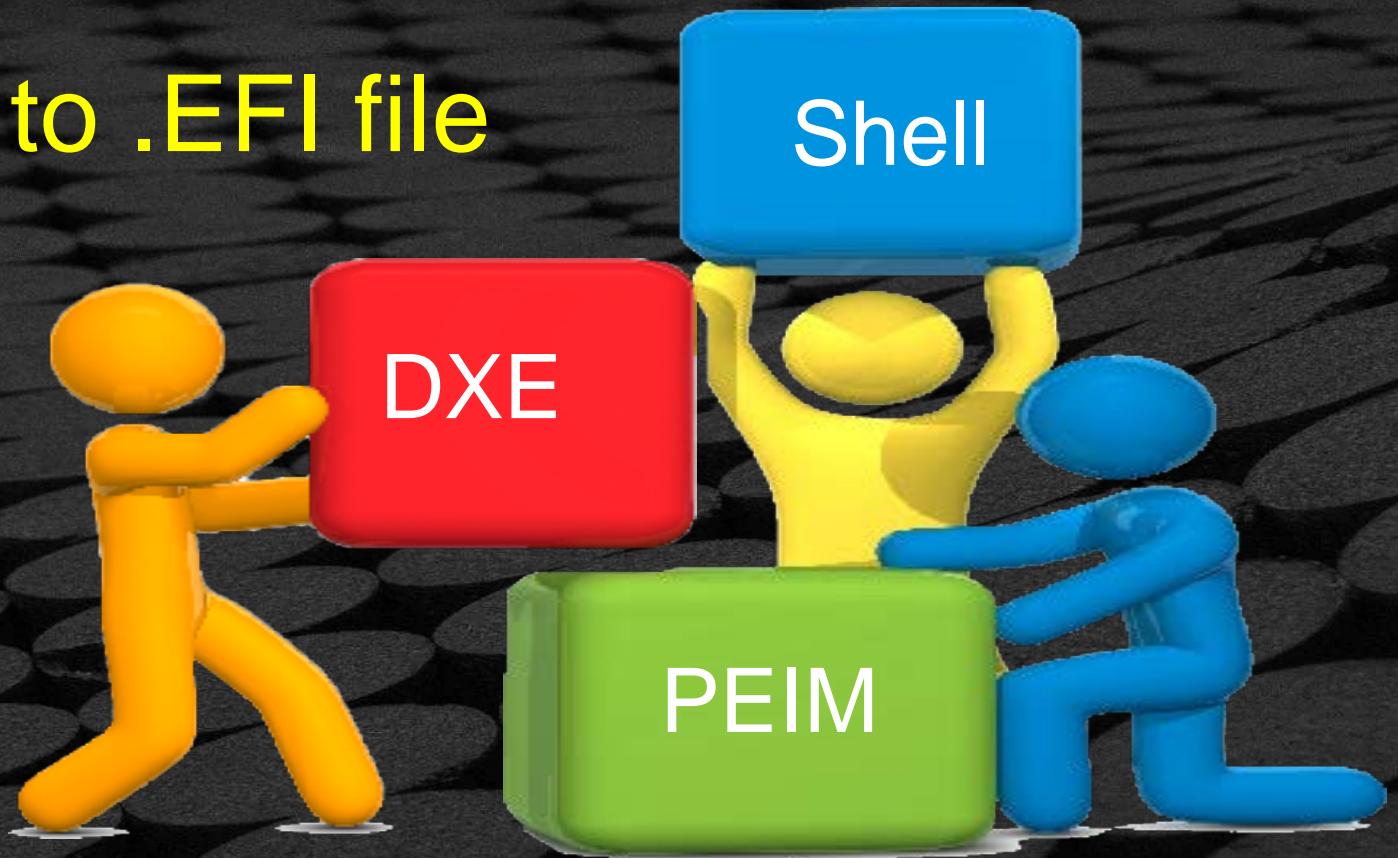
## MODULES



# MODULES

Smallest separate object compiled in EDK II

Compiles to .EFI file



UEFI/DXE Driver  
PEIM  
UEFI App.  
Library

Modules: Building blocks of EDK II

# MODULE TYPES

## Most Used Module Types

PEI\_CORE

BASE

PEIM

DXE\_RUNTIME\_DRIVER

UEFI\_DRIVER

DXE\_CORE

DXE\_DRIVER

UEFI\_APPLICATION

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

# Module Source Contents - minimum file

## MODULE\_TYPE

UEFI Application

UEFI Driver

DXE Driver

UEFI Driver - *Library*

## Example Source Files

# Module Source Contents - minimum file

## MODULE\_TYPE

## Example Source Files

UEFI Application	– Foo.c, Foo.inf
UEFI Driver	– FooDriver.c, FooDriver.h, FooDriver.vfr, FooDriver.uni, FooDriver.inf
DXE Driver	– FooDxe.c, FooDxe.h, FooDxe.inf
UEFI Driver - <i>Library</i>	– FooLib.c., FooLib.h, FooLib.inf w/ LIBRARY_CLASS= FooLib   ...

# Module Source Contents - minimum file

## MODULE\_TYPE

## Example Source Files

UEFI Application	– Foo.c, Foo.inf
UEFI Driver	– FooDriver.c, FooDriver.h, FooDriver.vfr, FooDriver.uni, FooDriver.inf
DXE Driver	– FooDxe.c, FooDxe.h, FooDxe.inf
UEFI Driver - <i>Library</i>	– FooLib.c., FooLib.h, FooLib.inf w/ LIBRARY_CLASS= FooLib   ...

**COMPLEXITY - Number of source files dependent**

# Module Source Contents - minimum file

## MODULE\_TYPE      Example Source Files

UEFI Application	- Foo.c, Foo.inf
UEFI Driver	- FooDriver.c, FooDriver.h, FooDriver.vfr, FooDriver.uni, FooDriver.inf
DXE Driver	- FooDxe.c, FooDxe.h, FooDxe.inf
UEFI Driver - <i>Library</i>	- FooLib.c., FooLib.h, FooLib.inf w/ LIBRARY_CLASS= FooLib   ...

**COMPLEXITY** - Number of source files dependent

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

# Module Source Contents - minimum file

## MODULE\_TYPE      Example Source Files

UEFI Application	- Foo.c, Foo.inf
UEFI Driver	- FooDriver.c, FooDriver.h, FooDriver.vfr, FooDriver.uni, FooDriver.inf
DXE Driver	- FooDxe.c, FooDxe.h, FooDxe.inf
UEFI Driver - <i>Library</i>	- FooLib.c., FooLib.h, FooLib.inf w/ LIBRARY_CLASS= FooLib   ...

**COMPLEXITY** - Number of source files dependent

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

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

# EDK II LIBRARY MODULES

# Library Class

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

# Library Class

[LibraryClasses.common]

<LibraryClassName> | <LibraryInstancePathToInf/Name.inf>

DebugLib | MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf

Name

Implementation (1)

# Library Class

[LibraryClasses.common]

<LibraryClassName> | <LibraryInstancePathToInf/Name.inf>

DebugLib | MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf

Name

Implementation (2)

# Library Class

[LibraryClasses.common]

<LibraryClassName> | <LibraryInstancePathToInf/Name.inf>

DebugLib | MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf

Name

Implementation (3)

# Library Class

[LibraryClasses.common]

<LibraryClassName> | <LibraryInstancePathToInf/Name.inf>

DebugLib | MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf

Name

Implementation (3)

Consistent set of interfaces

# Library Class

[LibraryClasses.common]

<LibraryClassName> | <LibraryInstancePathToInf/Name.inf>

DebugLib | MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf

Name

Implementation (3)

Consistent set of interfaces

Does not describe implementation of the interfaces

# “NULL” Library Class

## Syntax

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

## Shell application example:

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

# “NULL” Library Class

## Constructors

### Syntax

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

### Shell application example:

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

# “NULL” Library Class

## Constructors

## Special Cases

### Syntax

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

### Shell application example:

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

# “NULL” Library Class

## Constructors

## Special Cases

### Syntax

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

NOT “... LibNull” instance

### Shell application example:

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

# “NULL” Library Class

## Constructors

## Special Cases

### Syntax

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

NOT “... LibNull” instance

### Shell application example:

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

### Open Source Example

DxeCrc32GuidedSectionExtractLib  
ShellPkg as used with Profiles

# Locating Library Classes

Library based upon

1. Industry specs (UEFI, etc.)

`MdePkg/MdeModulePkg`

2. Intel's framework<sup>1</sup> specs

`IntelFrameworkPkg/IntelFrameworkModulePkg`

# Locating Library Classes

Library based upon

1. Industry specs (UEFI, etc.)

`MdePkg/MdeModulePkg`

2. Intel's framework<sup>1</sup> specs

`IntelFrameworkPkg/IntelFrameworkModulePkg`

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

*Example: MdePkg.chm*

# Locating Library Classes

Library based upon

1. Industry specs (UEFI, etc.)

`MdePkg/MdeModulePkg`

2. Intel's framework<sup>1</sup> specs

`IntelFrameworkPkg/IntelFrameworkModulePkg`

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

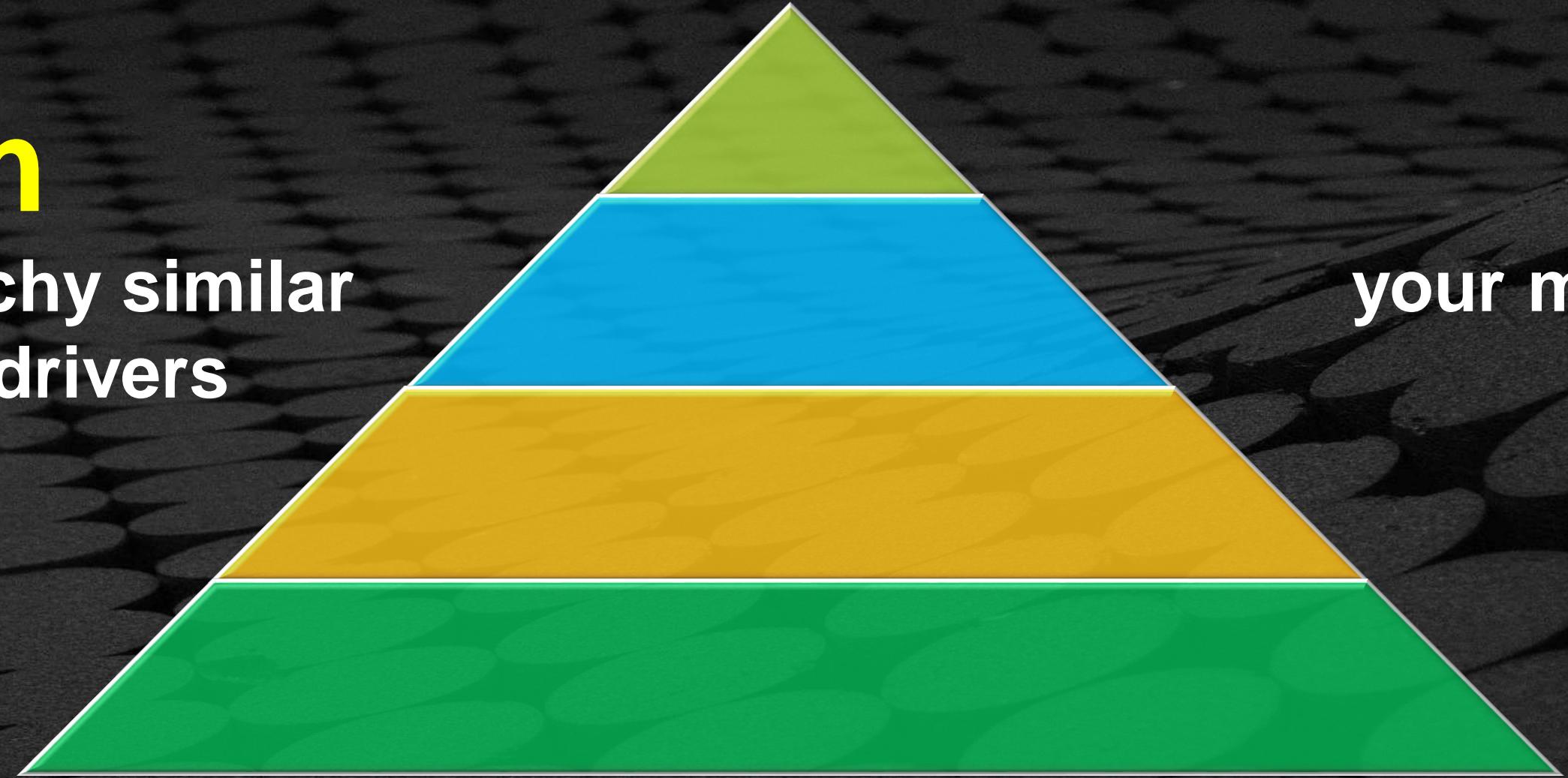
*Example: MdePkg.chm*

Search WorkSpace (.INF) "LIBRARY\_CLASS"

# Library Instance Hierarchy

**Form**  
a hierarchy similar  
to UEFI drivers

**Link**  
your module to  
another



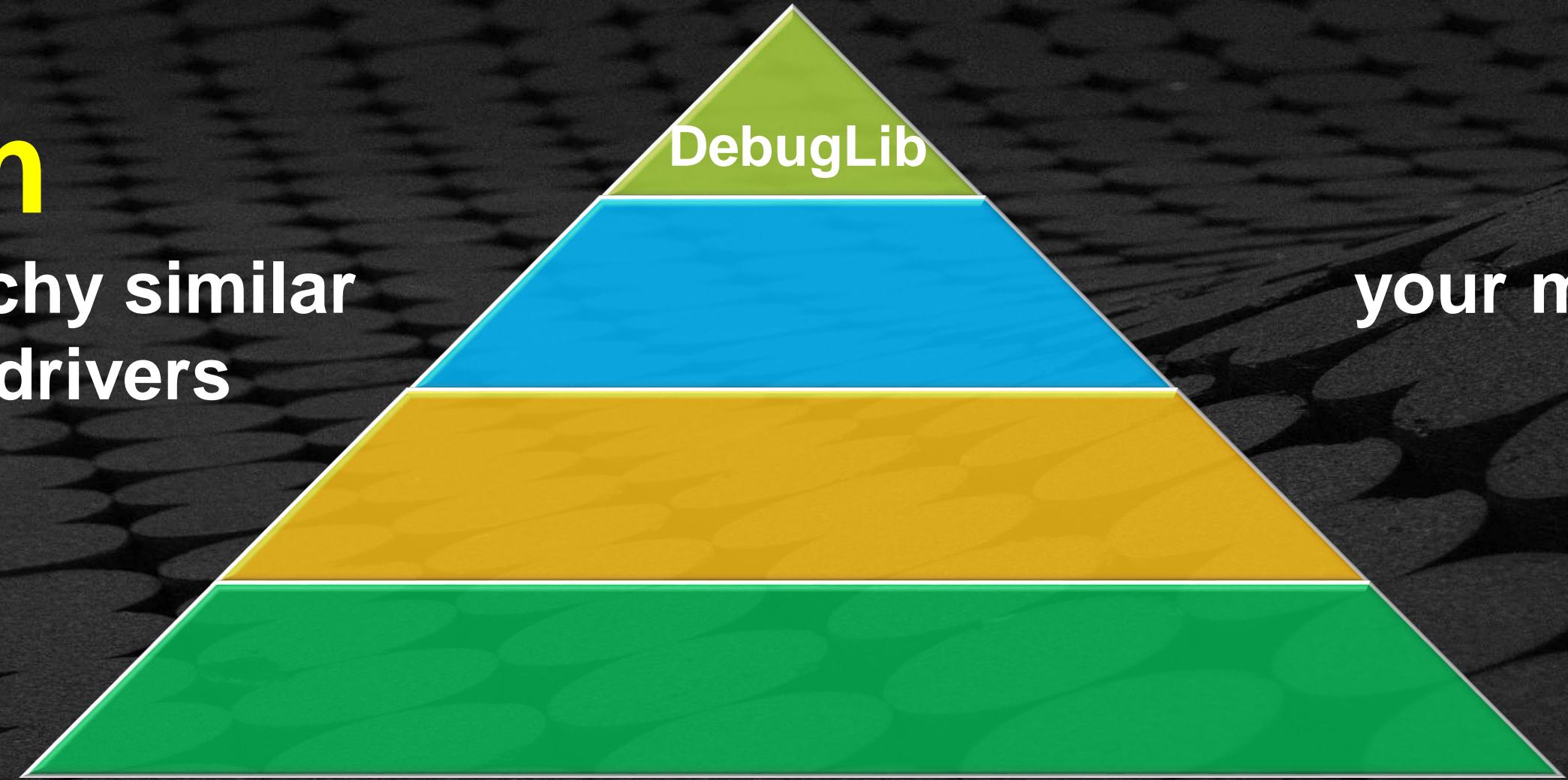
# Library Instance Hierarchy

## Form

a hierarchy similar  
to UEFI drivers

## Link

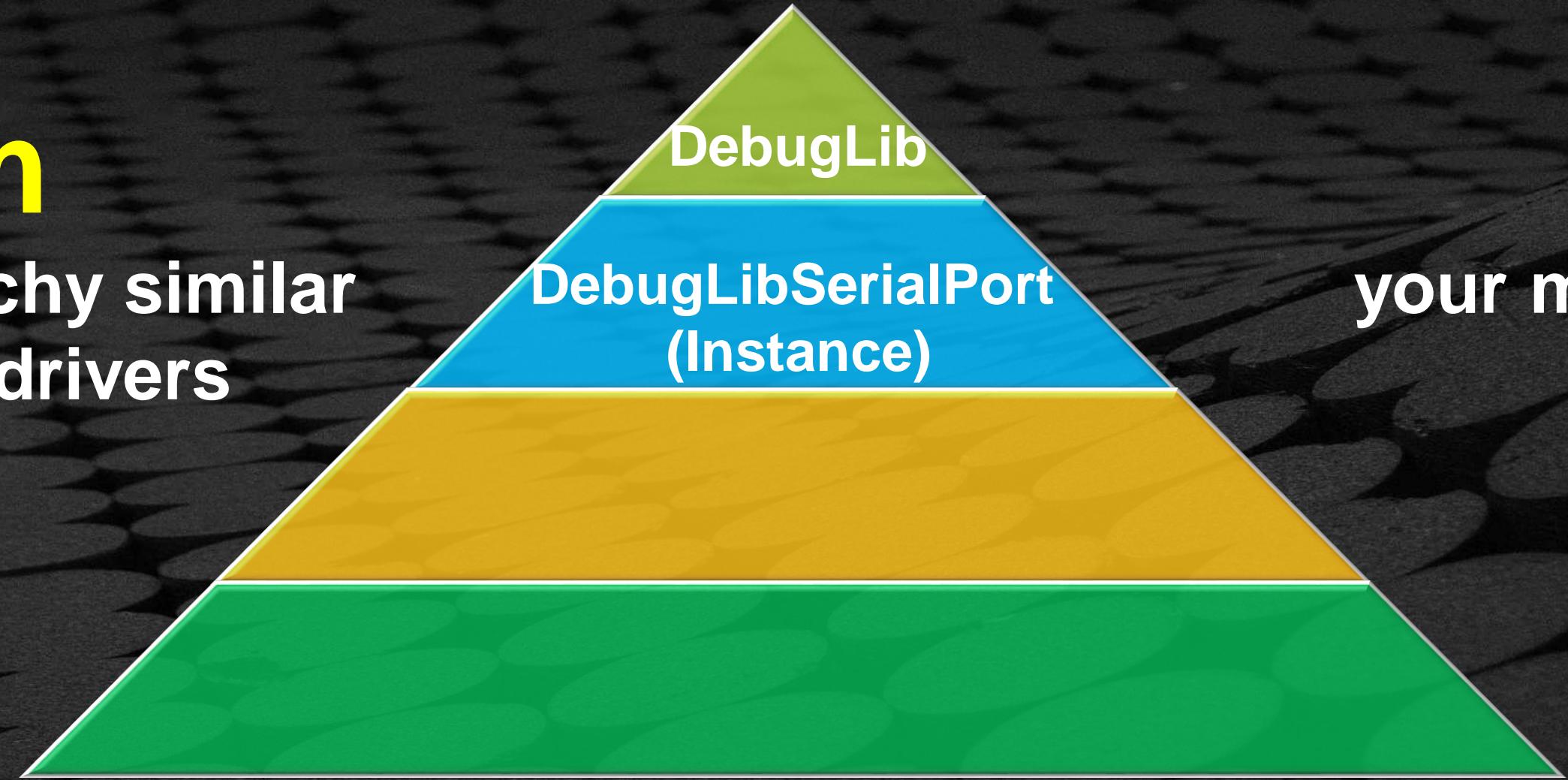
your module to  
another



# Library Instance Hierarchy

**Form**  
a hierarchy similar  
to UEFI drivers

**Link**  
your module to  
another



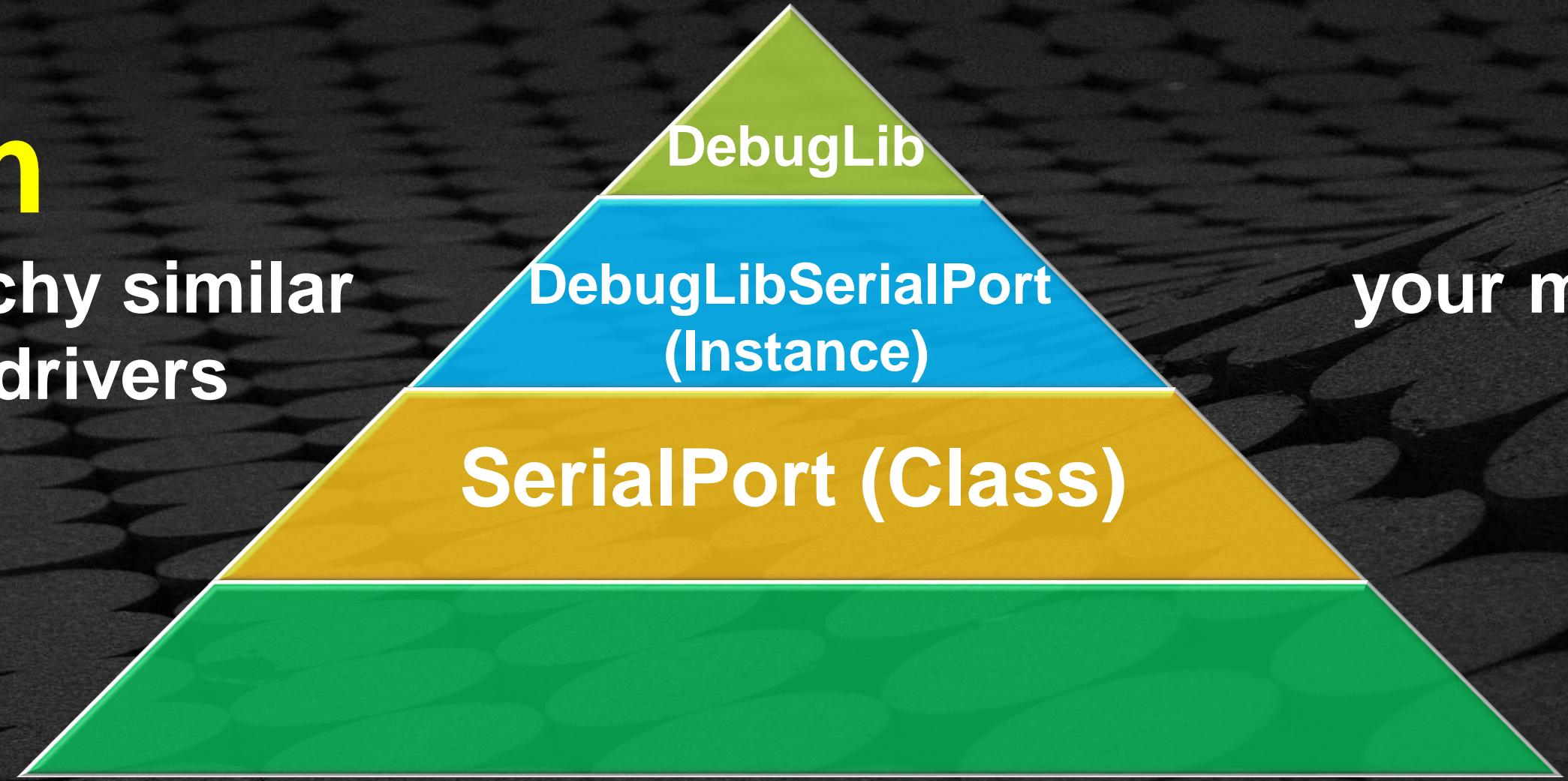
# Library Instance Hierarchy

## Form

a hierarchy similar  
to UEFI drivers

## Link

your module to  
another



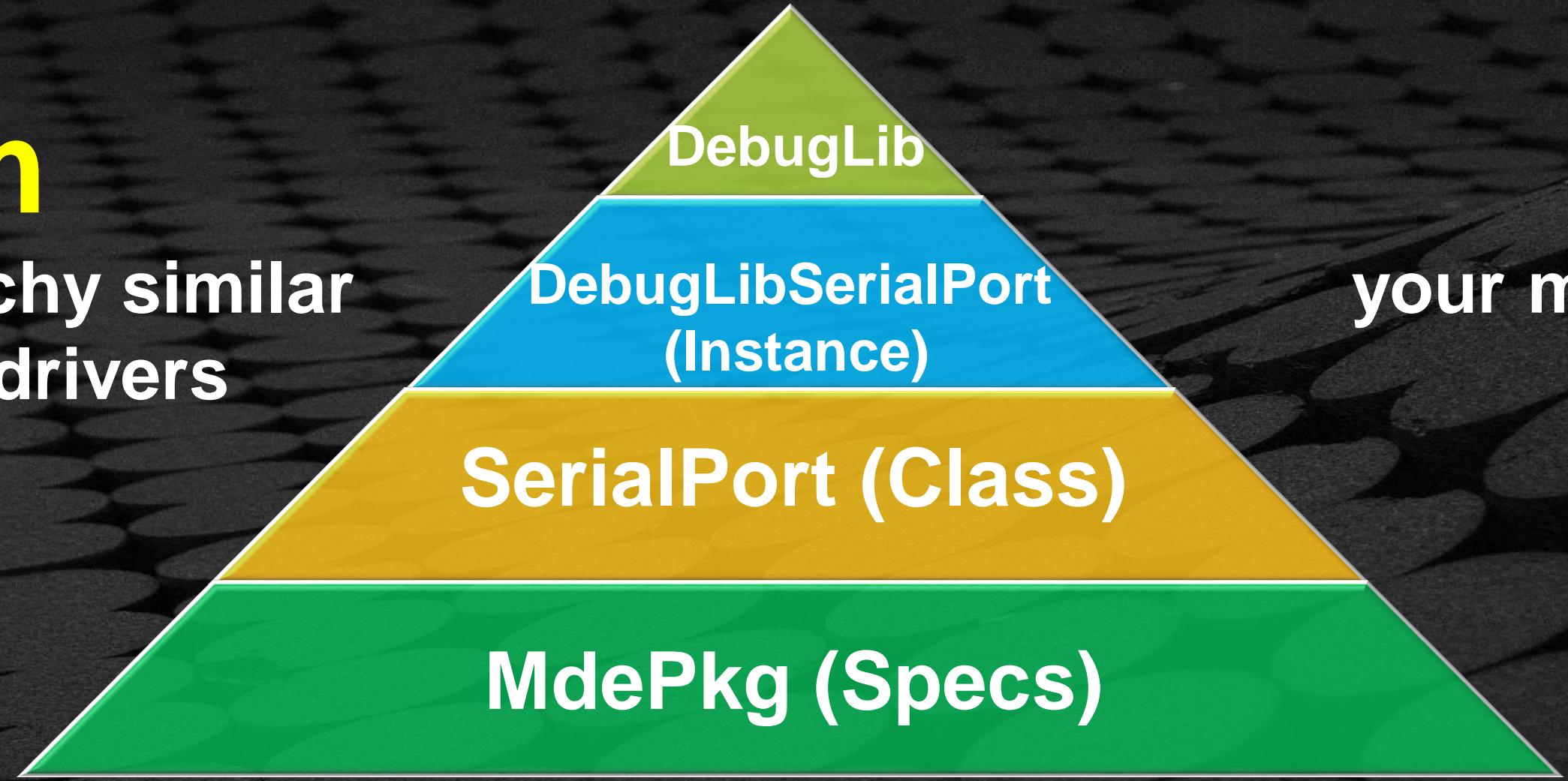
# Library Instance Hierarchy

## Form

a hierarchy similar  
to UEFI drivers

## Link

your module to  
another



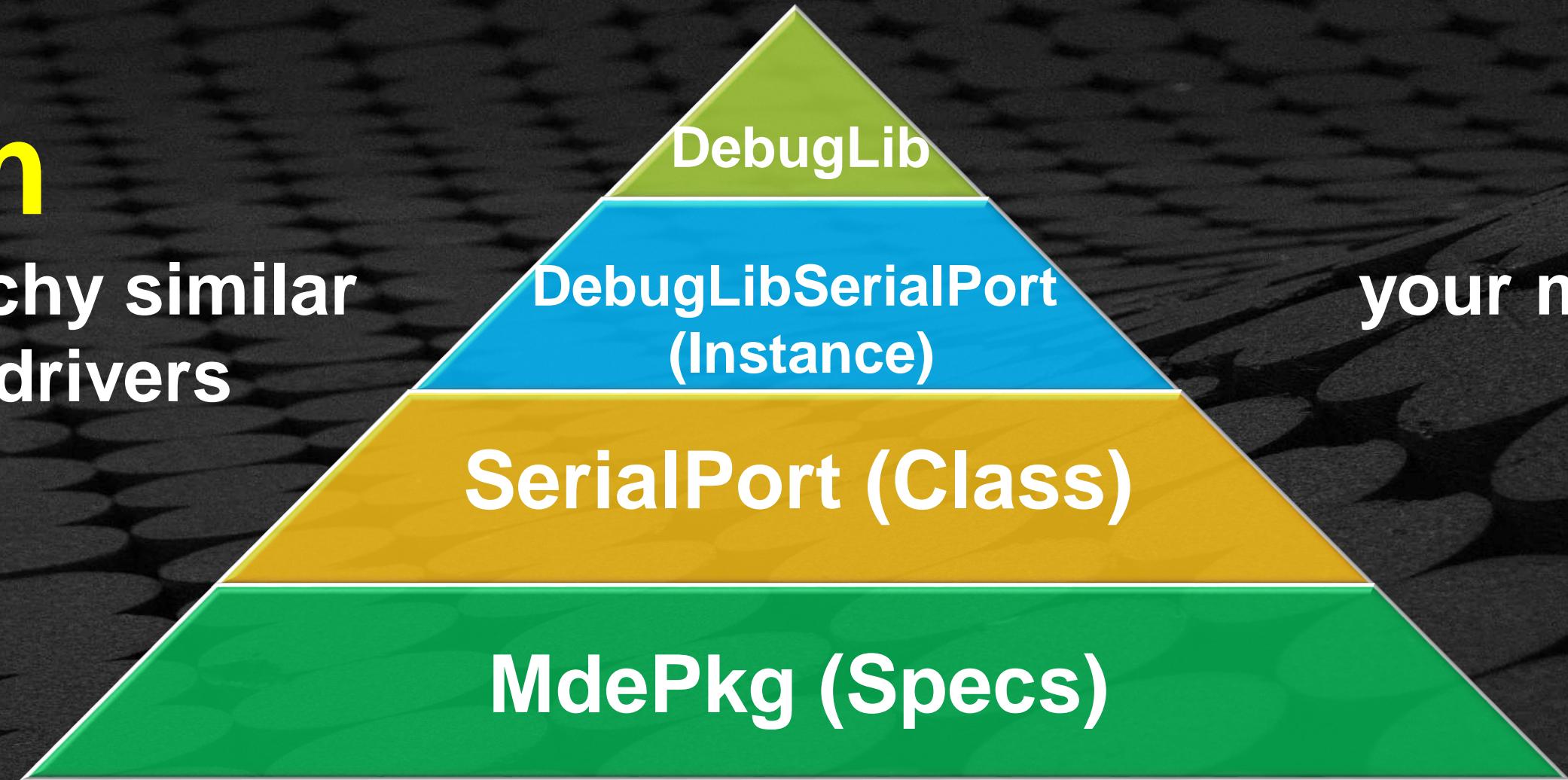
# Library Instance Hierarchy

## 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

# Commonly Used Base Library Classes

BaseLib      DebugLib  
UefiLib  
UefiApplicationEntryPoint

# Commonly Used Base Library Classes

BaseLib      DebugLib

UefiLib

BaseMemoryLib

UefiApplicationEntryPoint

PeiCoreEntryPoint

DxeCoreEntryPoint

PrintLib

UefiBootServicesTableLib

UefiScsiLib

DevicePathLib

PeimEntryPoint

# Commonly Used Base Library Classes

BaseLib      DebugLib

SalLib      CpuLib

UefiLib

BaseMemoryLib

UefiApplicationEntryPoint

UefiUsbLib

UefiDriverEntryPoint

MemoryAllocationLib

PeiCoreEntryPoint

DxeCoreEntryPoint

PrintLib

IoLib

UefiRuntimeLib

UefiBootServicesTableLib

UefiScsiLib

PciLib

DevicePathLib

PeimEntryPoint

# Commonly Used Base Library Classes

BaseLib

DebugLib

SalLib

CpuLib

UefiLib

BaseMemoryLib

UefiApplicationEntryPoint

UefiUsbLib

DxeServicesLib

SynchronizationLib

UefiDriverEntryPoint

MemoryAllocationLib

PeiCoreEntryPoint

DxeCoreEntryPoint

UefiRuntimeServicesTableLib

IoLib

PrintLib

UefiBootServicesTableLib

UefiRuntimeLib

PciSegmentLib

PciLib

UefiScsiLib

DevicePathLib

PciExpressLib

PeiServicesLib

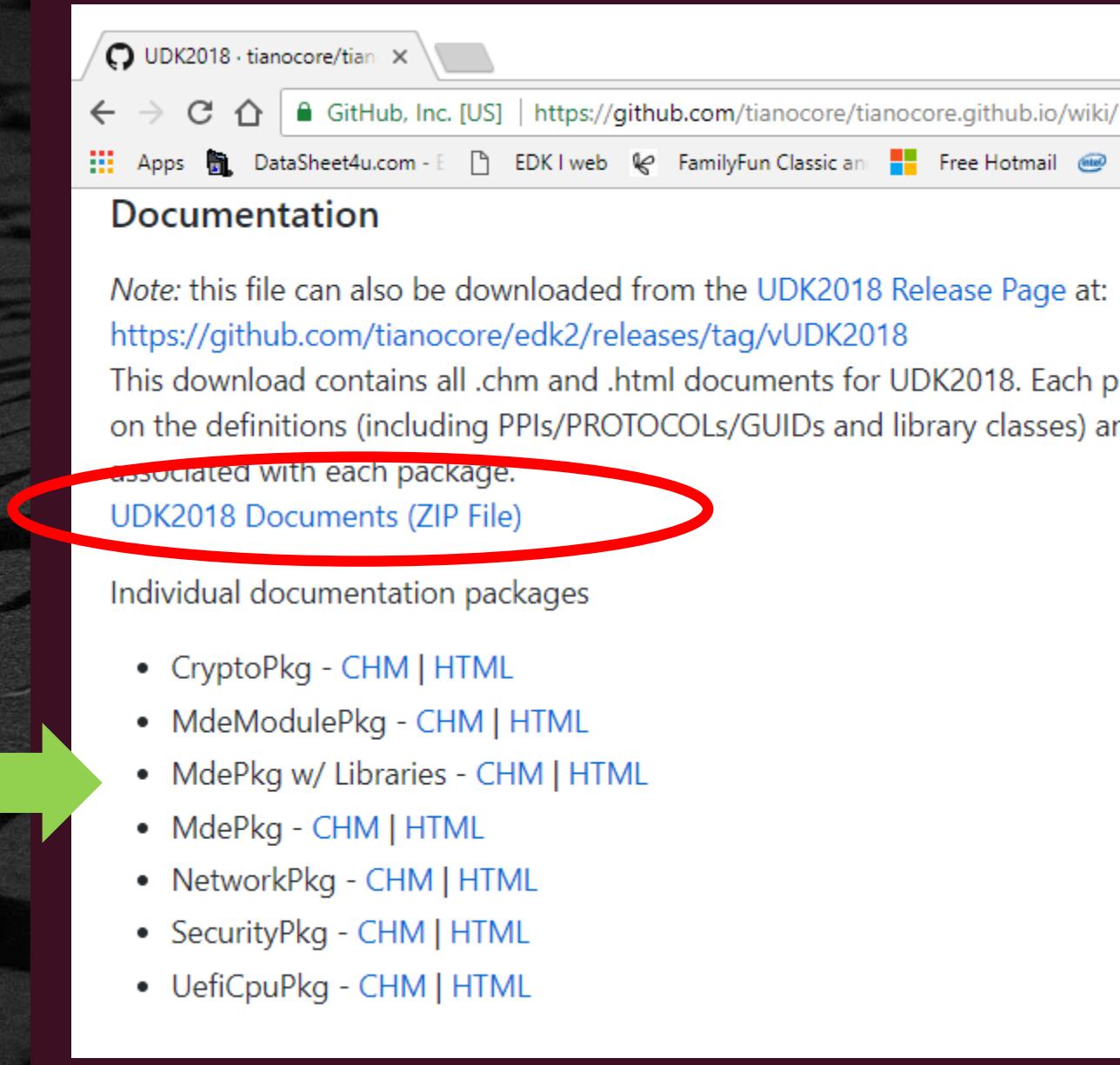
PeimEntryPoint

# MdePkg Library .CHM file Location

tianocore.org UDK2018 documentation on

 Latest UDK Release

 UDK2018



UDK2018 · tianocore/tianocore · GitHub

Documentation

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

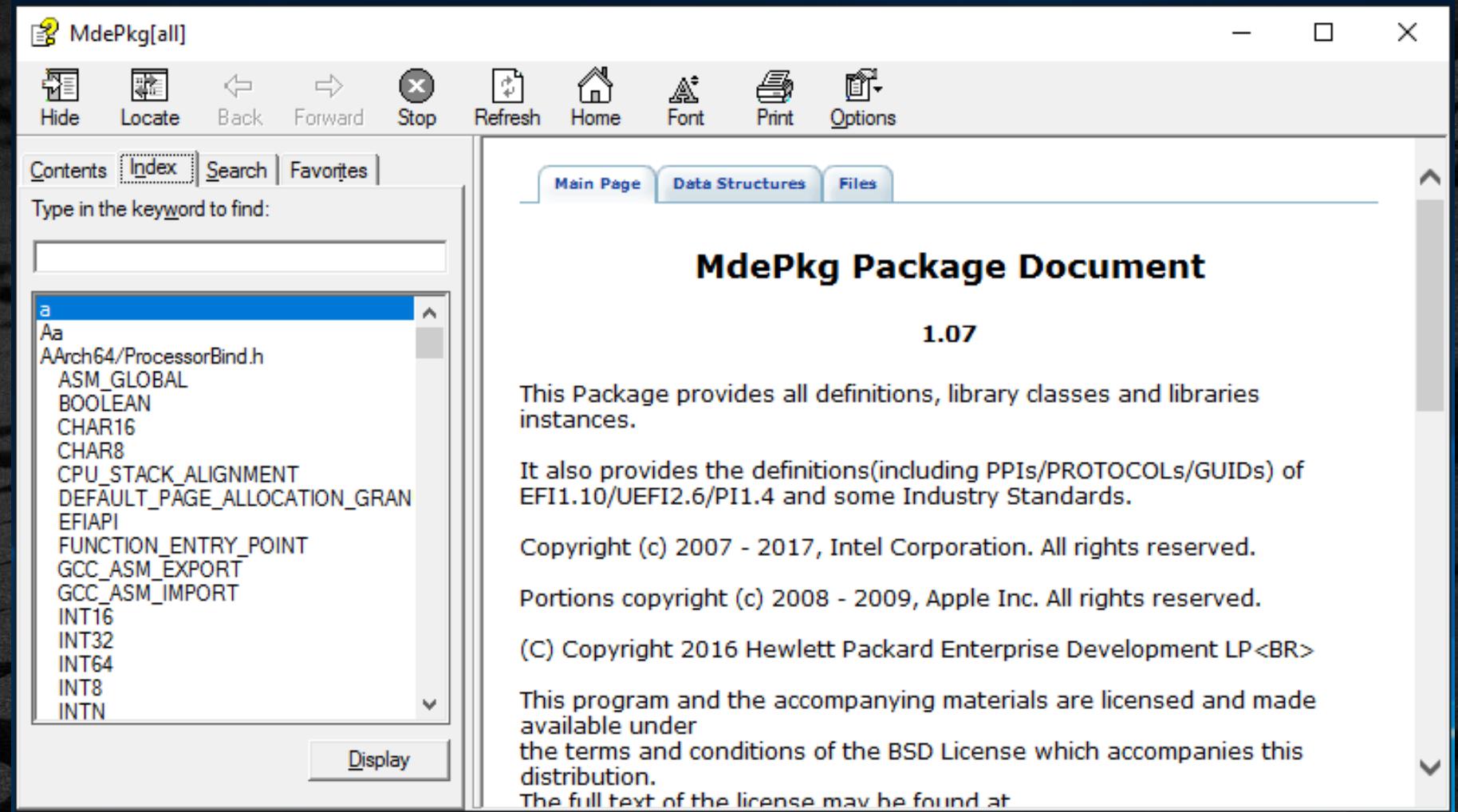
This download contains all .chm and .html documents for UDK2018. Each package contains the definitions (including PPIs/PROTOCOLs/GUIDs and library classes) associated with each package.

[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



The screenshot shows the Microsoft Help Viewer application window titled "MdePkg(all)". The main content area displays the "MdePkg Package Document" page, which includes the version "1.07", a brief description of the package's purpose, copyright information for Intel and Apple, and a note about the BSD license. On the left, there is a search bar and a list of indexed terms starting with 'a'. A scroll bar is visible on the right side of the window.

MdePkg Package Document

1.07

This Package provides all definitions, library classes and libraries instances.

It also provides the definitions(including PPIs/PROTOCOLs/GUIDs) of EFI1.10/UEFI2.6/PI1.4 and some Industry Standards.

Copyright (c) 2007 - 2017, Intel Corporation. All rights reserved.

Portions copyright (c) 2008 - 2009, Apple Inc. All rights reserved.

(C) Copyright 2016 Hewlett Packard Enterprise Development LP<BR>

This program and the accompanying materials are licensed and made available under  
the terms and conditions of the BSD License which accompanies this distribution.  
The full text of the license may be found at

a

AArch64/ProcessorBind.h  
ASM\_GLOBAL  
BOOLEAN  
CHAR16  
CHAR8  
CPU\_STACK\_ALIGNMENT  
DEFAULT\_PAGE\_ALLOCATION\_GRAN  
EFIAPI  
FUNCTION\_ENTRY\_POINT  
GCC\_ASM\_EXPORT  
GCC\_ASM\_IMPORT  
INT16  
INT32  
INT64  
INT8  
INTN

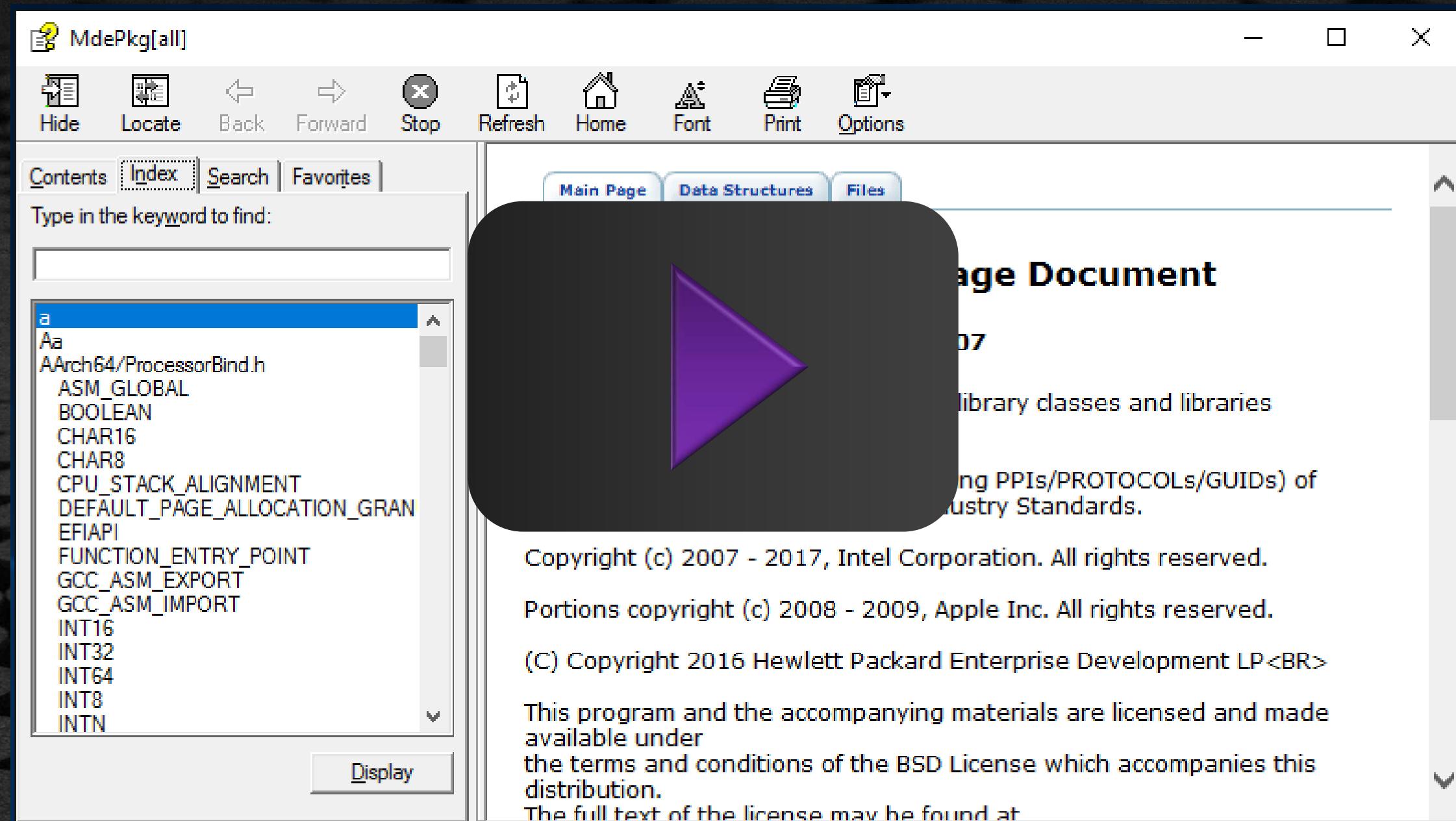
Display

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 Loadable Image

- ★ 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 Loadable Image Usages

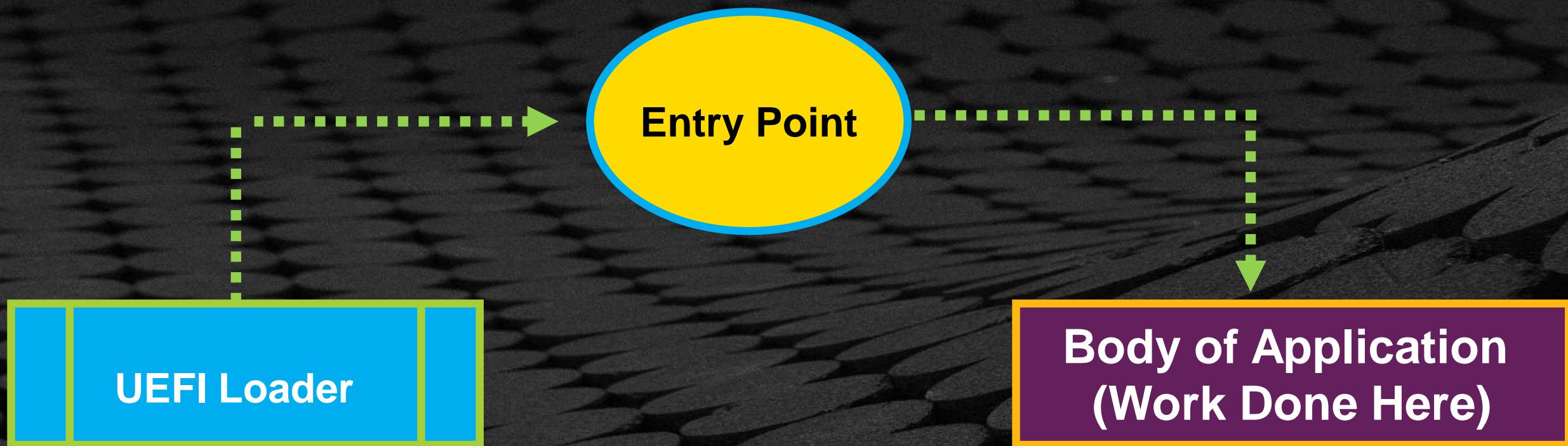
- ★ Platform Diagnostics
- ★ Factory Diagnostics
- ★ Utilities
- ★ Driver Prototyping
- ★ “Platform” Applications
- ★ Portable Across Platforms (IA32, X64, ARM, Itanium, etc.)

# Executing Applications

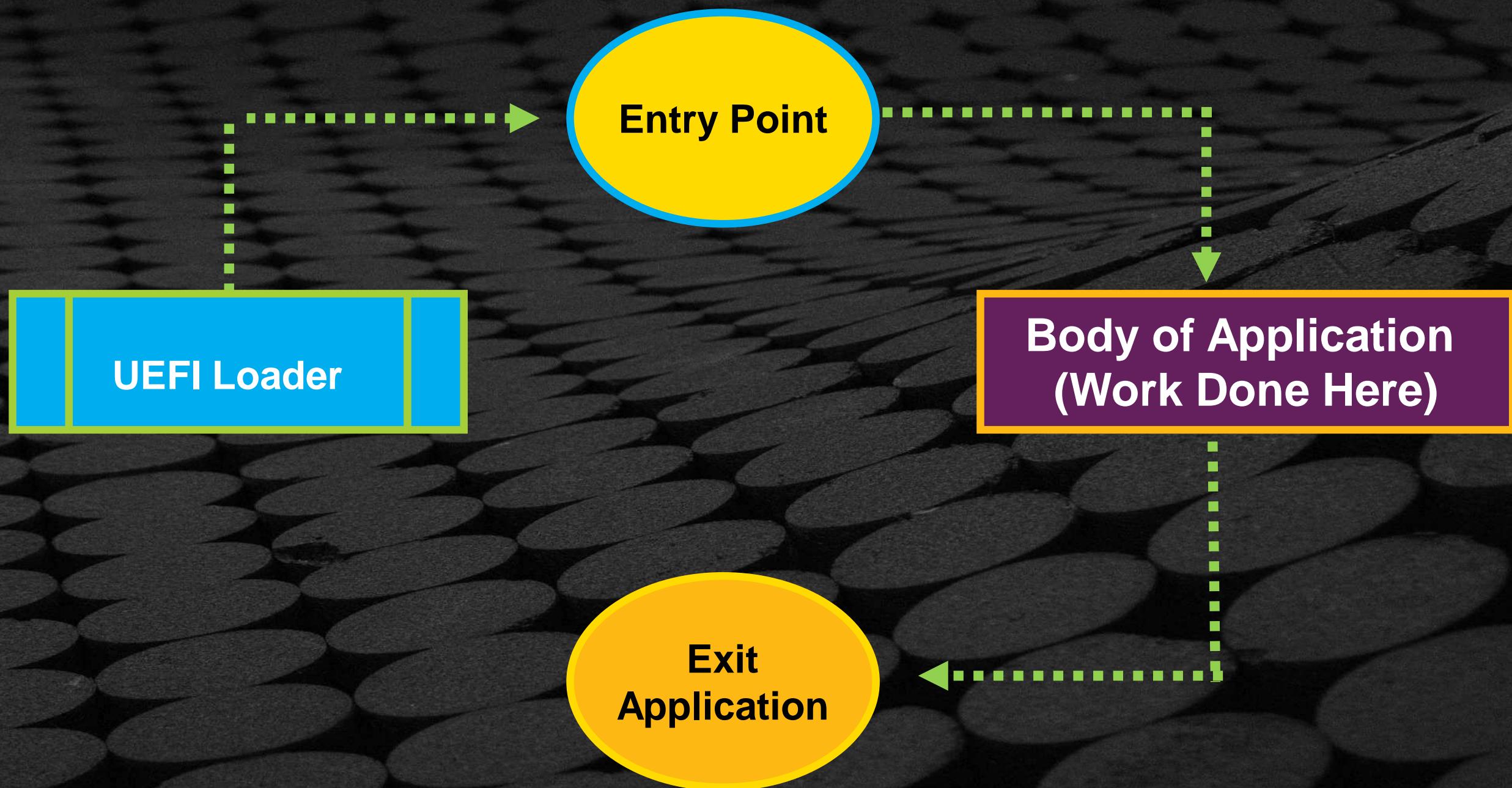


UEFI Loader

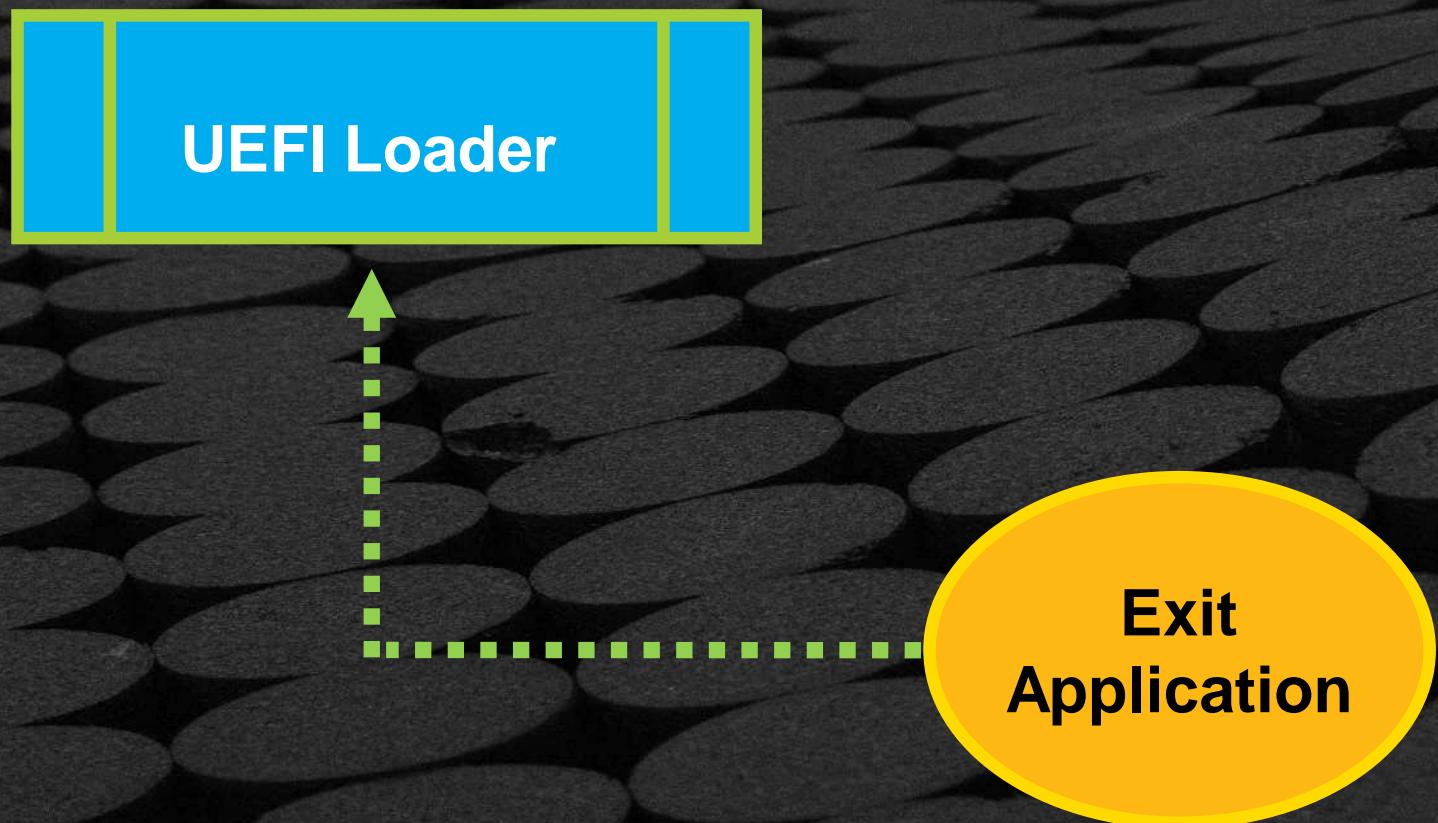
# Executing Applications



# Executing Applications



# Executing Applications



# Executing Applications



UEFI Loader

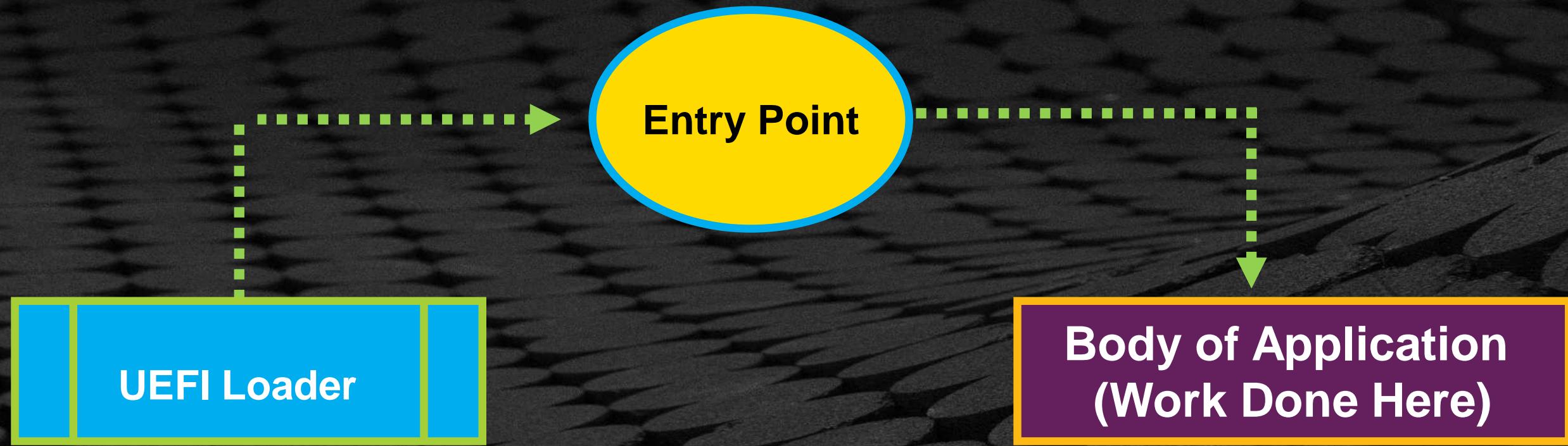
# Executing Applications



UEFI Loader

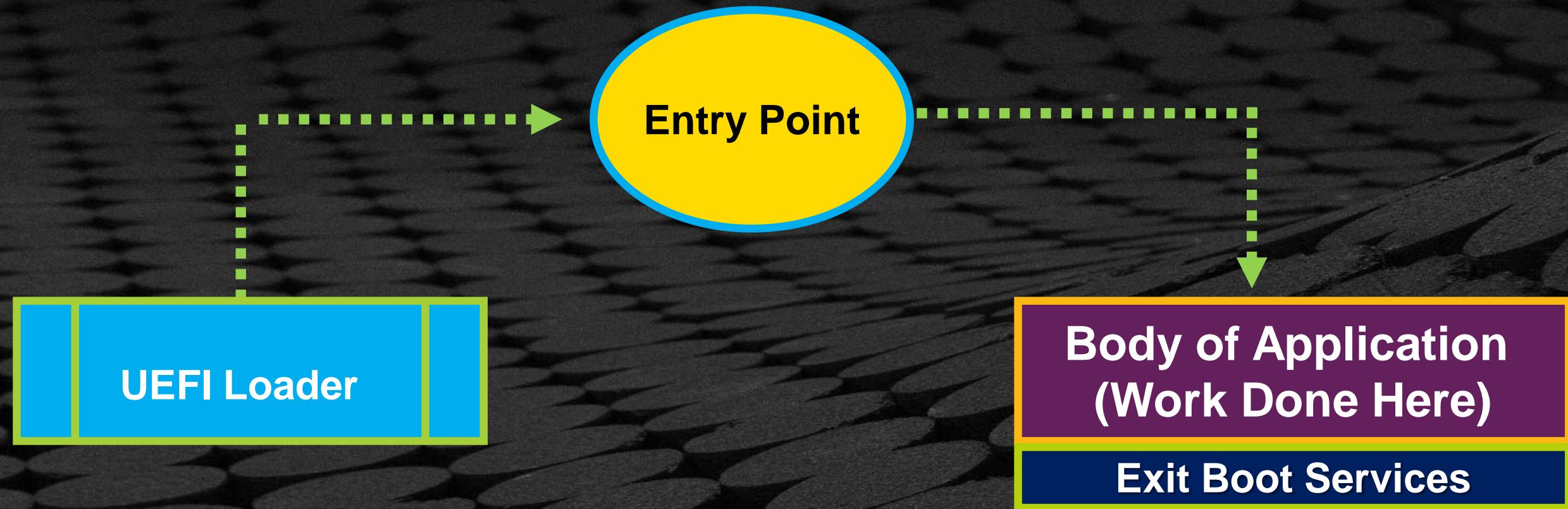
## OS Loader

# Executing Applications



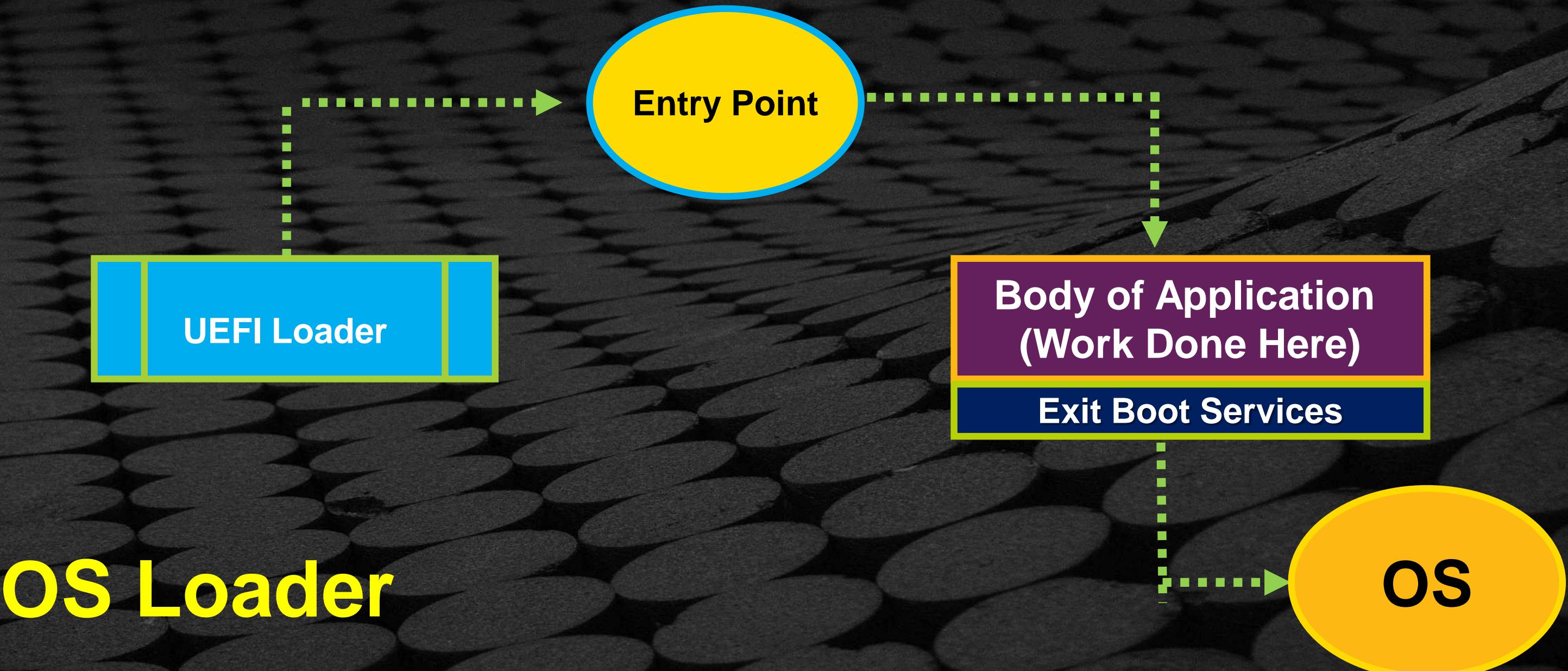
## OS Loader

# Executing Applications

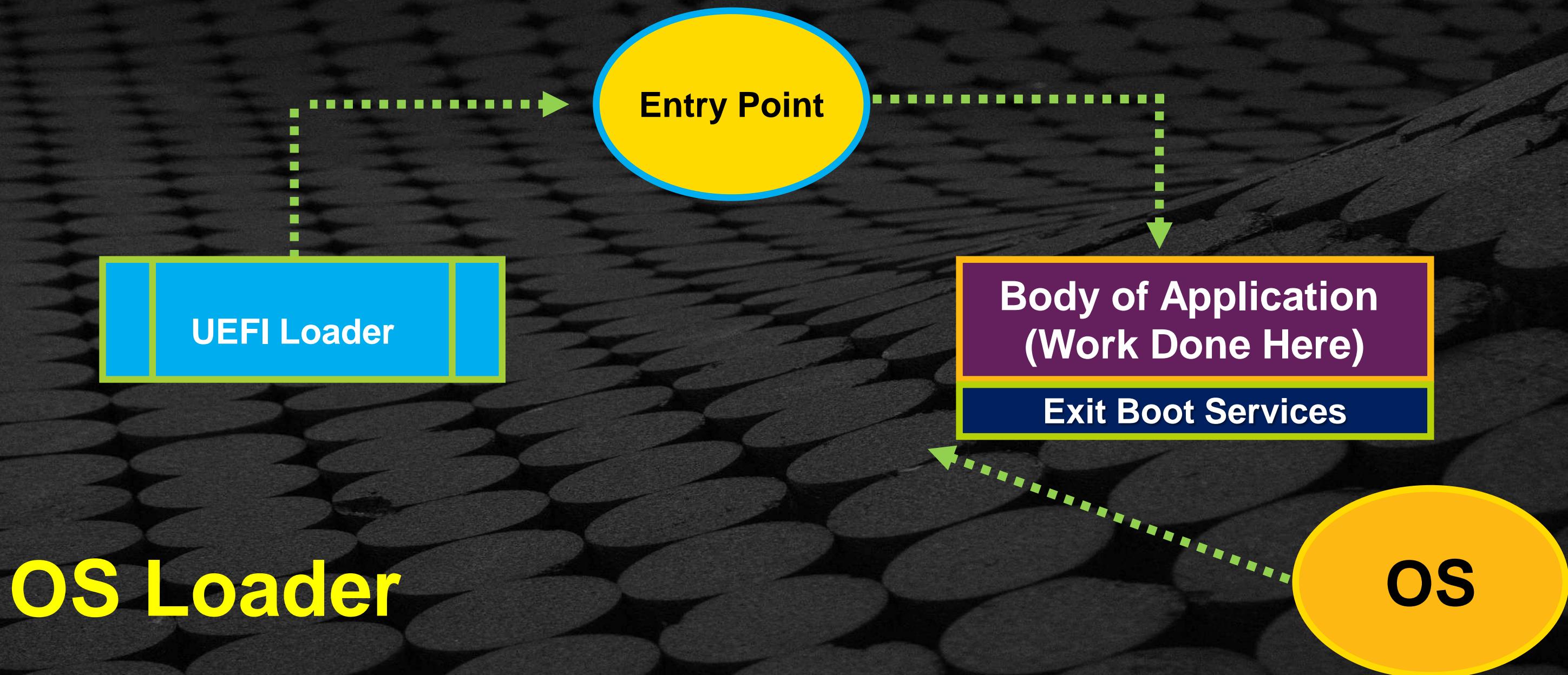


## OS Loader

# Executing Applications



# Executing Applications



# Executing Applications

OS Loader



OS

# Driver Vs. Application

	Driver	Application
<b>Loaded by:</b>	UEFI Loader	UEFI Loader
<b>Interfaces available:</b>	ALL	ALL
<b>Consume protocols?</b>	YES	YES
<b>Produce protocols?</b>	YES	NO
<b>Typically driven by?</b>	System	User
<b>Typical use</b>	Support HW	Any

# Driver Vs. Application

	Driver	Application
<b>Loaded by:</b>	UEFI Loader	UEFI Loader
<b>Interfaces available:</b>	ALL	ALL
<b>Consume protocols?</b>	YES	YES
<b>Produce protocols?</b>	YES	NO
<b>Typically driven by?</b>	System	User
<b>Typical use</b>	Support HW	Any

# Driver Vs. Application

	Driver	Application
<b>Loaded by:</b>	UEFI Loader	UEFI Loader
<b>Interfaces available:</b>	ALL	ALL
<b>Consume protocols?</b>	YES	YES
<b>Produce protocols?</b>	YES	NO
<b>Typically driven by?</b>	System	User
<b>Typical use</b>	Support HW	Any

# EDK II UEFI APPLICATIONS

## How to Write a EDK II UEFI Application

# Application Files Placement

 Application source code can go anywhere in the EDK II workspace

# Application Files Placement

- ★ Application source code can go anywhere in the EDK II workspace
- ★ All code and include files go under a single directory containing an INF

# Application Files Placement

- ★ Application source code can go anywhere in the EDK II workspace
- ★ All code and include files go under a single directory containing an INF
- ★ EDK Sample Applications can be found here:

**/MdeModulePkg/Application**

# Application Files Placement

- Application source code can go anywhere in the EDK II workspace
- All code and include files go under a single directory containing an INF
- EDK Sample Applications can be found here:  
**/MdeModulePkg/Application**
- Typically, modules reside within a package:

```
/MyWorkSpace  
  /MyPkg  
    /Application  
      /MyApp
```



```
MyApp.c  
MyApp.inf
```



# Module .INF File

## Syntax

INF text file example

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

# 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
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
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]
```

# BUILDING AN APPLICATION

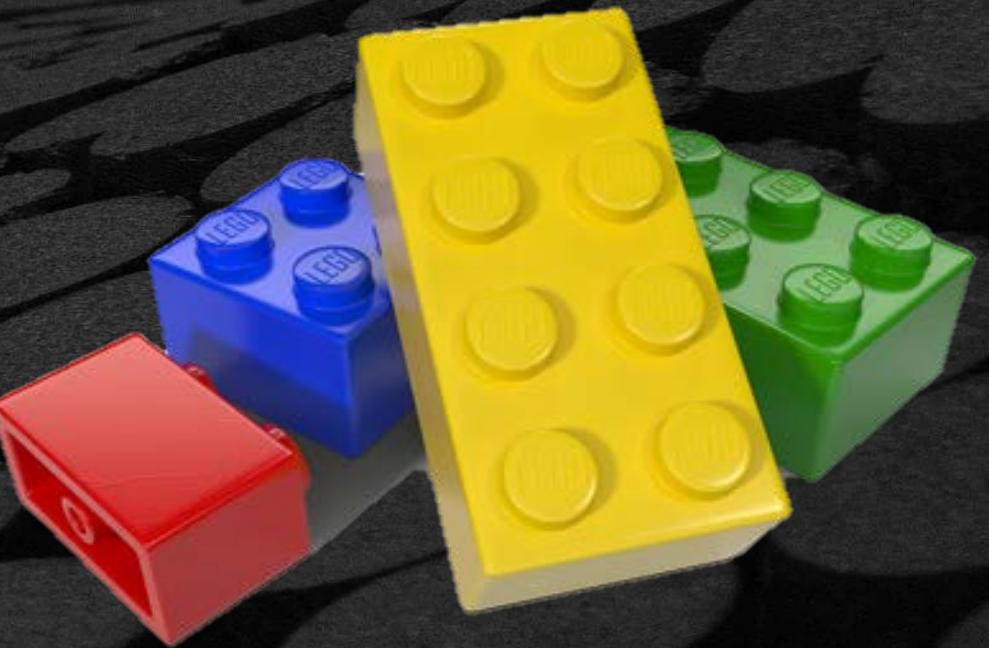
Platform .DSC references .INF

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

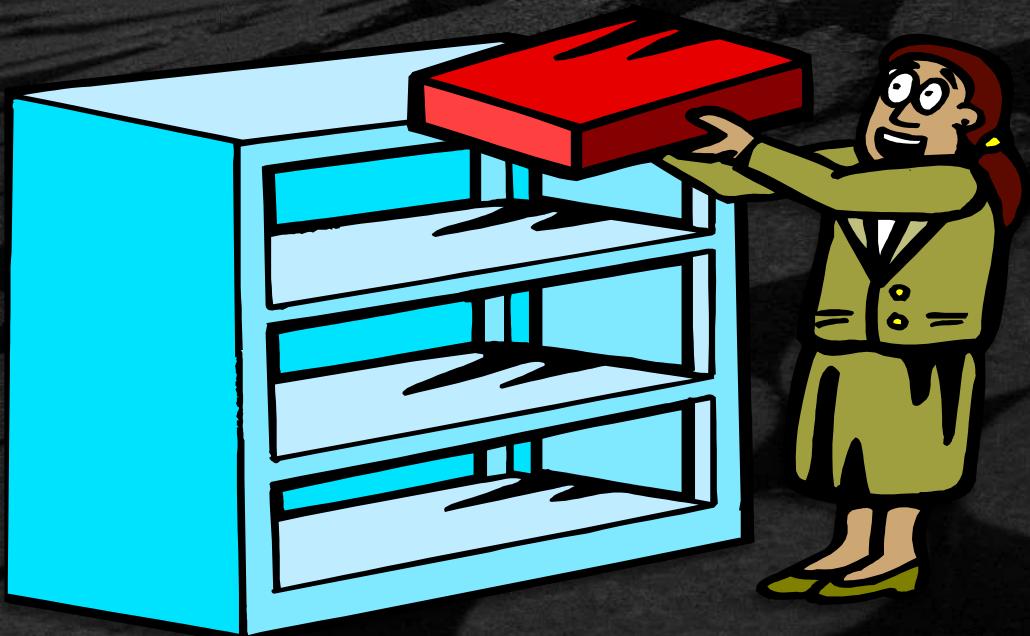
```
#include <Uefi.h>
#include <Library/UefiApplicationEntryPoint.h>

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

# 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 (
    IN int Argc,
    IN char **Argv
)
{
    return 0;
}
```

# Driver File Placement

 Driver source files can be located anywhere in the EDK II workspace

# Driver File Placement

- ★ Driver source files can be located anywhere in the EDK II workspace
- ★ All code and include files go under a single directory containing the driver INF

# Driver File Placement

- ★ Driver source files can be located anywhere in the EDK II workspace
- ★ All code and include files go under a single directory containing the driver INF
- ★ Good example drivers in EDK II:  
**MdeModulePkg/Bus/Scsi/ScsiDiskDxe**  
**IntelFrameworkModulePkg/Universal/BdsDxe**

# Driver File Placement

- ★ Driver source files can be located anywhere in the EDK II workspace
- ★ All code and include files go under a single directory containing the driver INF
- ★ Good example drivers in EDK II:  
**MdeModulePkg/Bus/Scsi/ScsiDiskDxe**  
**IntelFrameworkModulePkg/Universal/BdsDxe**
- ★ Drivers normally reside in a package

# 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



# Changes for a UEFI Driver Module

Applications can be converted to a driver

It remains in memory after it runs...



# Changes for a UEFI Driver Module

Applications can be converted to a driver

It remains in memory after it runs...

UEFI Driver Module requirements:

- Driver Binding Protocol
- Component Name2 Protocol (recommended)



# Changes for a UEFI Driver Module

Applications can be converted to a driver

It remains in memory after it runs...

UEFI Driver Module requirements:

- Driver Binding Protocol
- Component Name2 Protocol (recommended)

DXE/PEIM/other Driver requirements



# Sample INF file

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

[Sources]
MyFile.c

[Packages]
MdePkg/MdePkg.dec

[LibraryClasses]
UefiDriverEntryPoint

[Guids]

[Ppis]

[Protocols]
```

# INF Usage Fields – DIST files

## UEFI Spec – Package Distribution

*Optional*

Usage Fields used by Build tools for creating the  
.Dist files for BIN Modules

- [GUID]
- [PCD]
- [PROTOCOL]
- [PPIS]

1 Usage Block

– “##” After the entry

$n$  Usage Blocks

– “##” Precede the entry

### Usage Key Word

## UNDEFINED  
## CONSUMES  
## SOMETIMES\_CONSUMES  
## PRODUCES  
## SOMETIMES\_PRODUCES  
## TO\_START  
## BY\_START  
## NOTIFY



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

## INF File

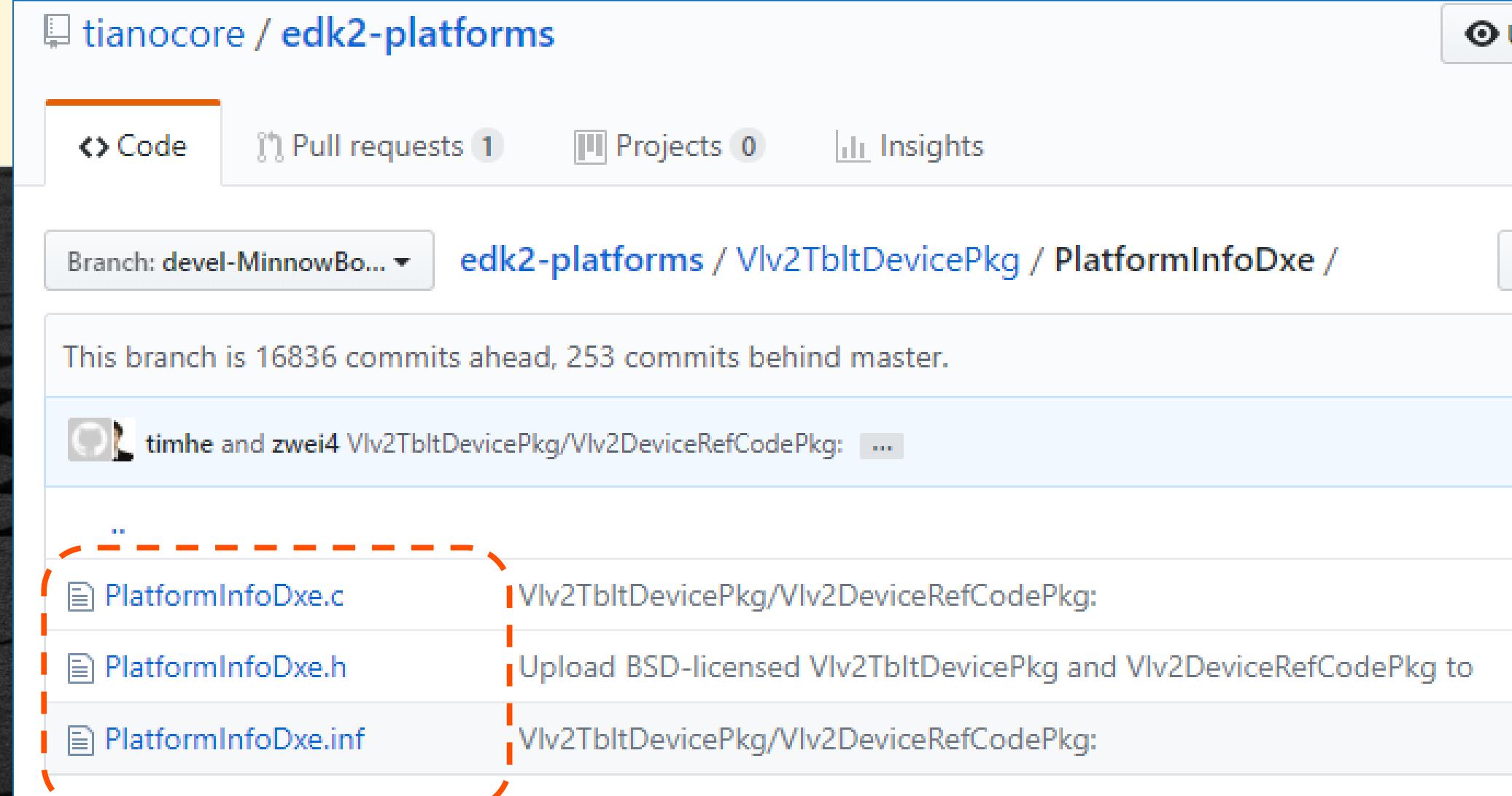
[Protocols]

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

# DXE Driver Example - PlatformInfoDxe



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



tianocore / [edk2-platforms](#)

[Code](#) [Pull requests 1](#) [Projects 0](#) [Insights](#)

Branch: [devel-MinnowBo...](#) [edk2-platforms / Vlv2TbtDevicePkg / PlatformInfoDxe /](#)

This branch is 16836 commits ahead, 253 commits behind master.

 [timhe and zwei4](#) [Vlv2TbtDevicePkg/Vlv2DeviceRefCodePkg:](#) [...](#)

[PlatformInfoDxe.c](#) [Vlv2TbtDevicePkg/Vlv2DeviceRefCodePkg:](#)  
[PlatformInfoDxe.h](#) [Upload BSD-licensed Vlv2TbtDevicePkg and Vlv2DeviceRefCodePkg to](#)  
[PlatformInfoDxe.inf](#) [Vlv2TbtDevicePkg/Vlv2DeviceRefCodePkg:](#)

# 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

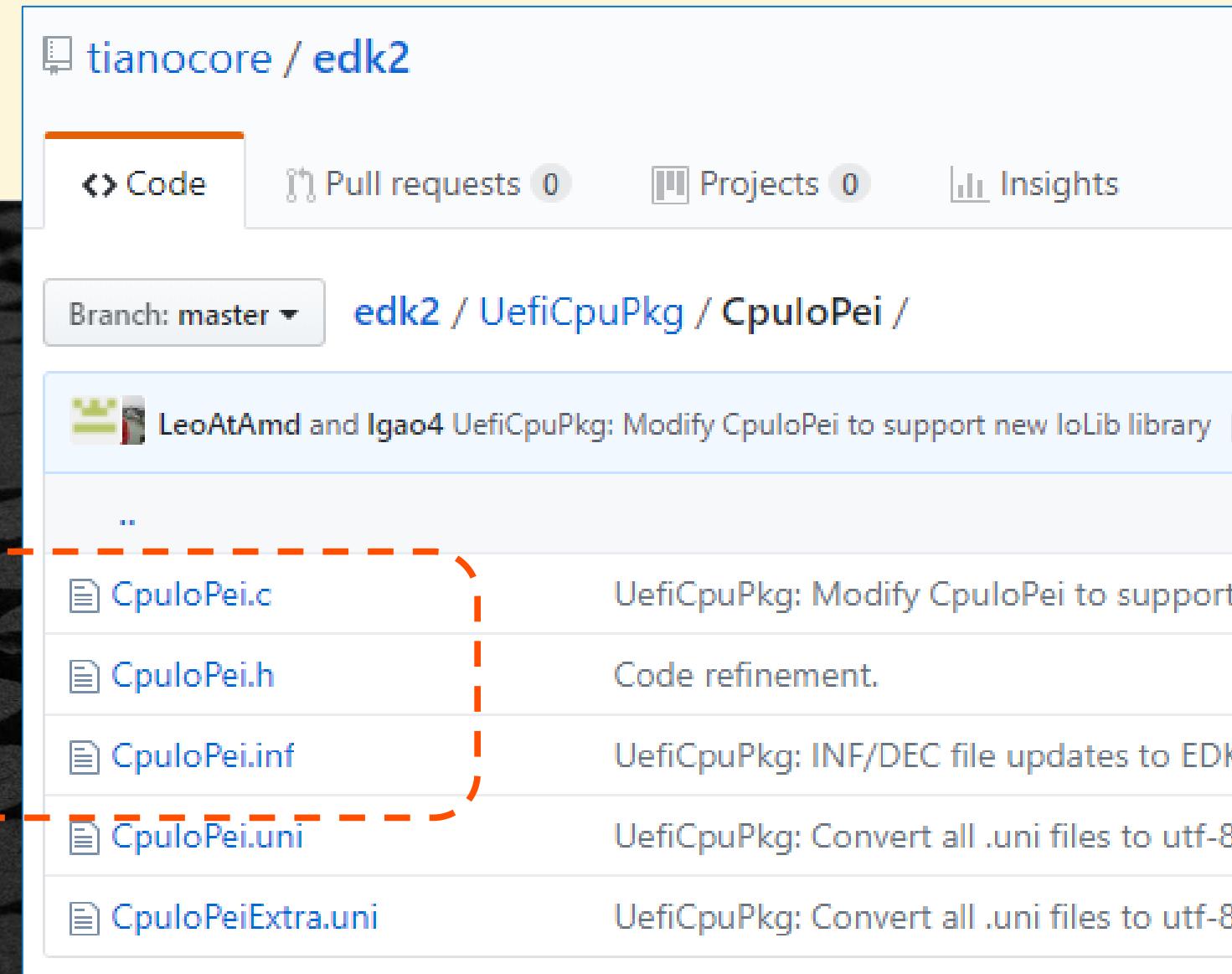
```
[Defines]  
• • •  
MODULE_TYPE      = DXE_DRIVER  
VERSION_STRING   = 1.0  
ENTRY_POINT     = PlatformInfoInit  
• • •  
[Depex]  
gEfiVariableArchProtocolGuid AND  
gEfiVariableWriteArchProtocolGuid
```

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



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

## Entry Point

### “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

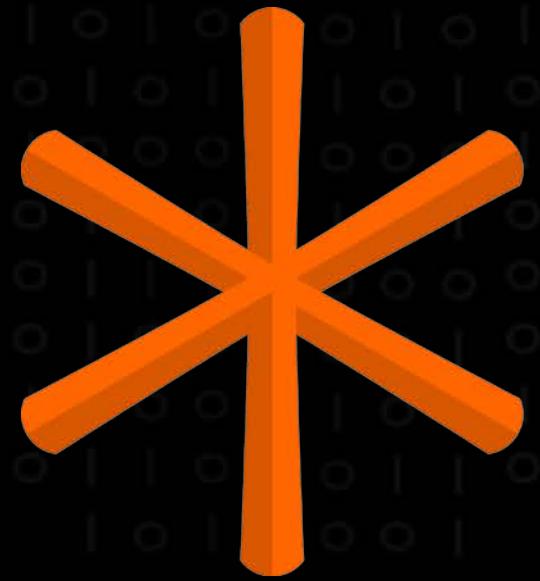
```
[Defines]
. . .
MODULE_TYPE          = PEIM
VERSTON_STRTNG      = 1.0
ENTRY_POINT          = CpuIoInitialize
. . .

[Depex]
TRUE
```

# 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





# tianocore

