

# UEFI & EDK II Training

How to Write a UEFI Driver

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



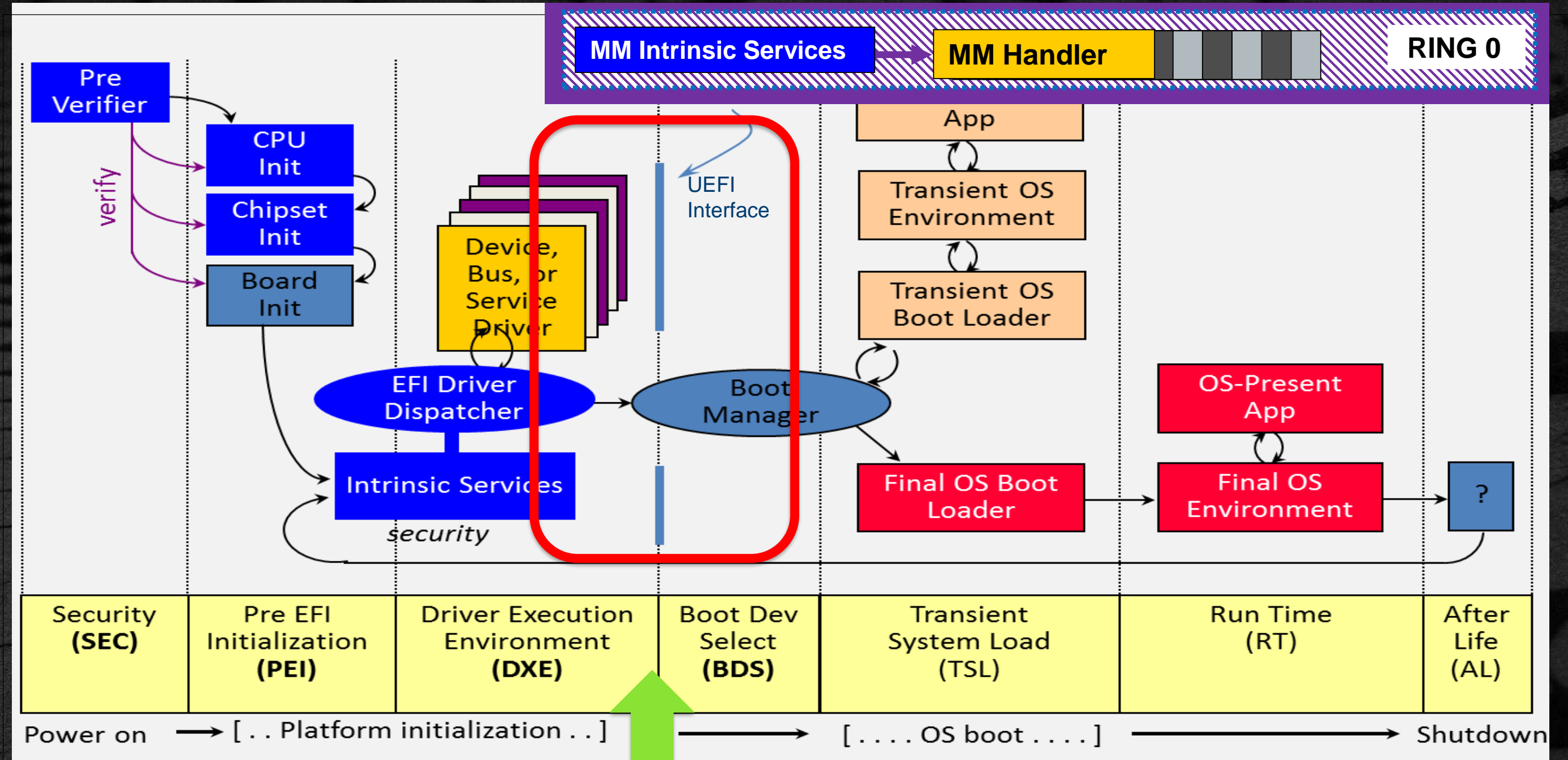
# LESSON OBJECTIVE

- ★ What is the UEFI Driver Model
- ★ Details on Driver Binding Protocol
- ★ Example of UEFI Driver

# UEFI DRIVER MODEL



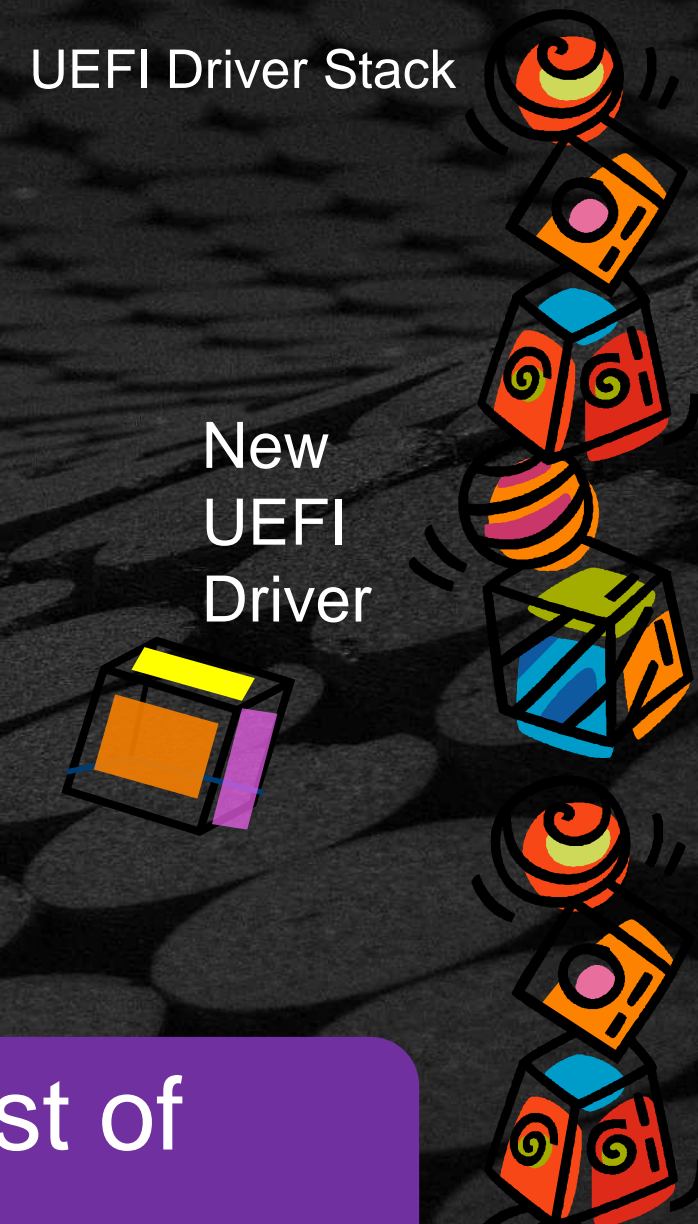
# UEFI Drivers - Location





# What are UEFI Drivers ?

- UEFI Drivers extend firmware
- Portable across platforms
- Enables rapid development
- Produce Protocols



UEFI driver is chained into a link list of  
**Drivers Managing Devices**



# Defining a UEFI Driver

UEFI Loadable Image

May produce/consume protocols

Supports complex bus hierarchies

Driver Binding Protocol matches drivers to devices,  
adds version management

Supports specific hardware, can be unloaded or  
override an existing driver

# What is a UEFI protocol?

## Protocols

- Interfaces consisting of functions and data structures named by a GUID and stored in the Handle Database

## Handle Database

- Everything in the platform system gets a handle, drivers, devices, Images, etc.

## GUIDs

- The UEFI Platform only knows items in the Handle Database by its GUID



# UEFI Drivers Vs. Applications

A blue rectangular box with a yellow border, containing the text "UEFI Loader".

UEFI Loader



# UEFI Drivers Vs. Applications





# UEFI Drivers Vs. Applications





# UEFI Drivers Vs. Applications



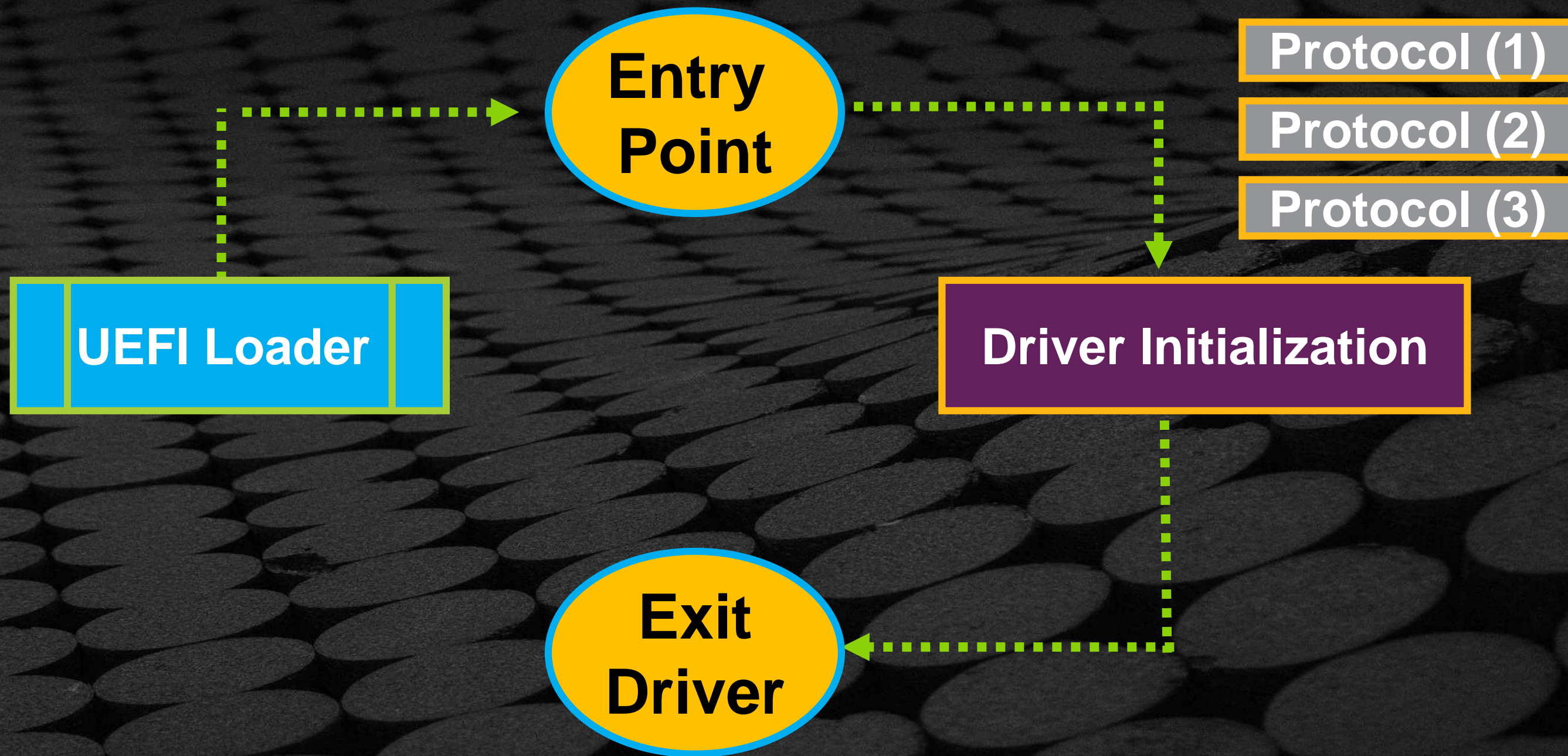


# UEFI Drivers Vs. Applications





# UEFI Drivers Vs. Applications





# UEFI Drivers Vs. Applications

Protocol (1)

Protocol (2)

Protocol (3)

UEFI Loader

Driver Initialization

Exit  
Driver



# UEFI Drivers Vs. Applications

Protocol (1)

Protocol (2)

Protocol (3)

UEFI Loader

Driver Initialization



# UEFI Drivers Vs. Applications

**UEFI Loader**

**Protocol (1)**

**Protocol (2)**

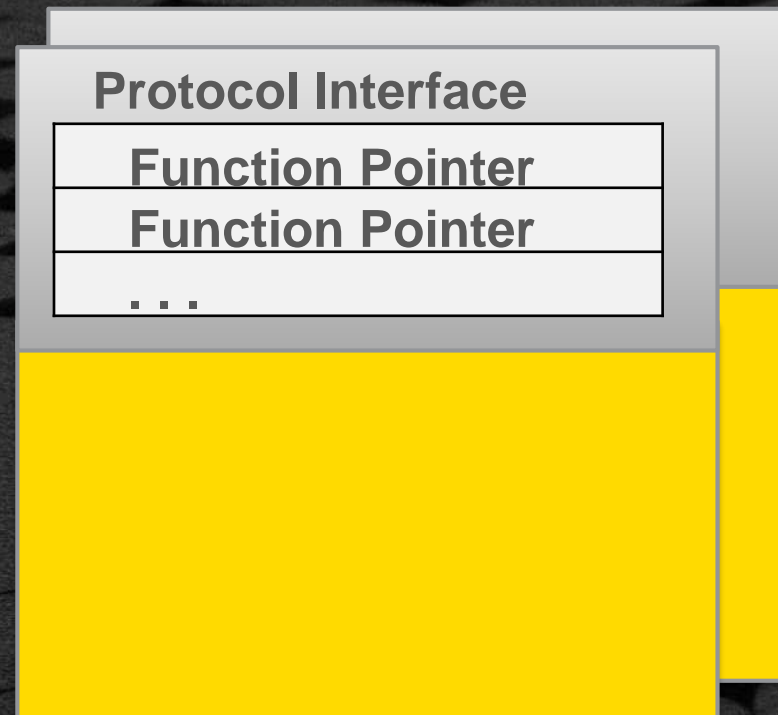
**Protocol (3)**

**Driver Initialization**



# Drivers Produce Protocols

**Handle  
Database**

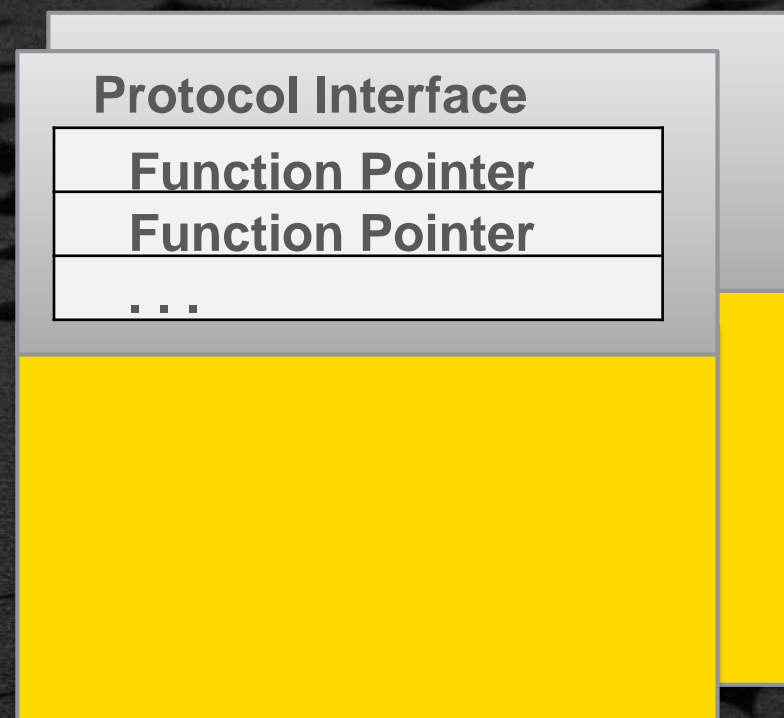






## Construction of a protocol

**Handle  
Database**



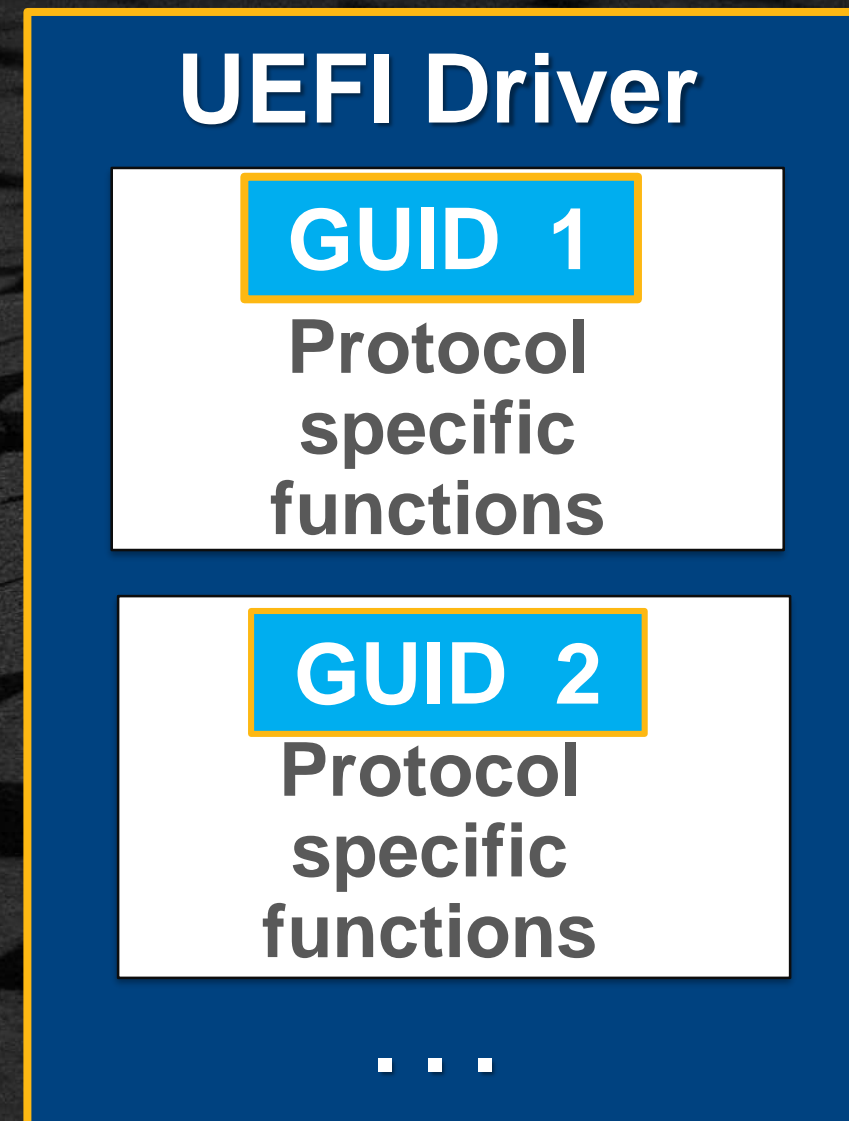
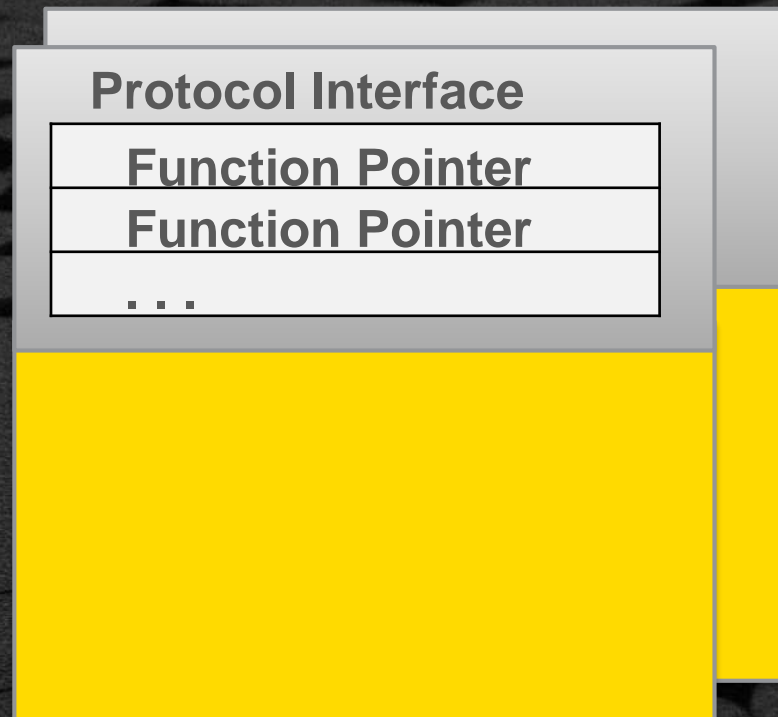


# Drivers Produce Protocols



## Construction of a protocol

**Handle  
Database**





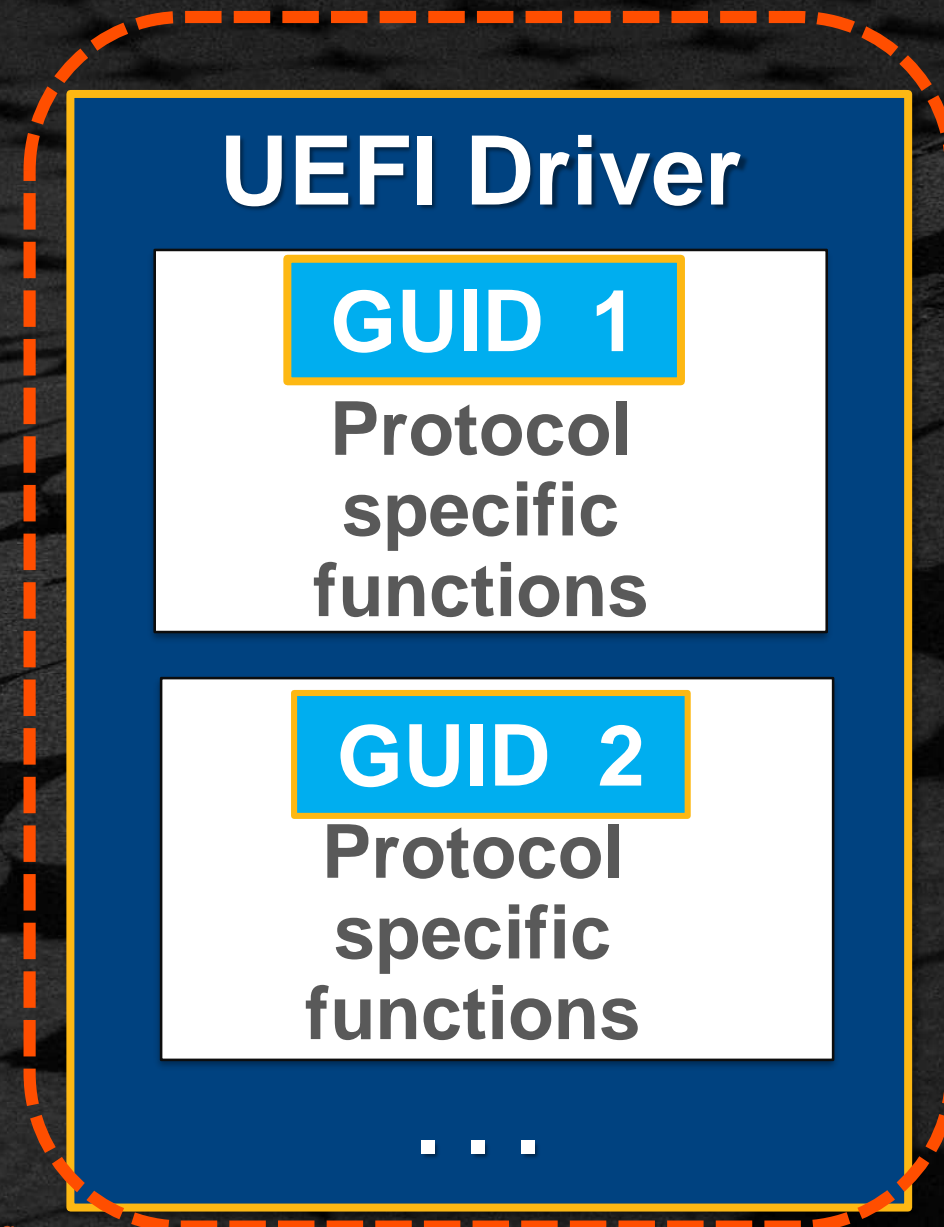
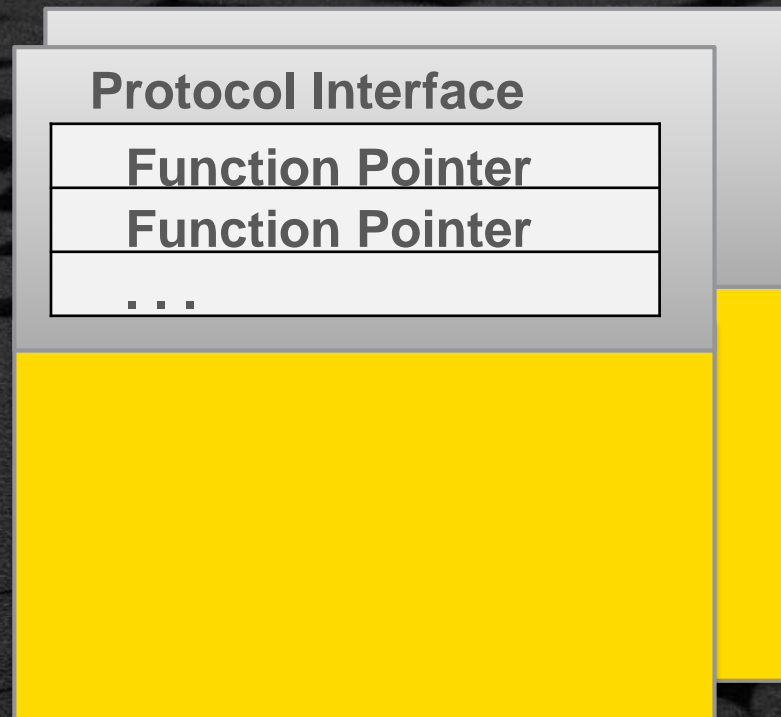
# Drivers Produce Protocols



## Construction of a protocol

InstallProtocolInterface

Handle  
Database





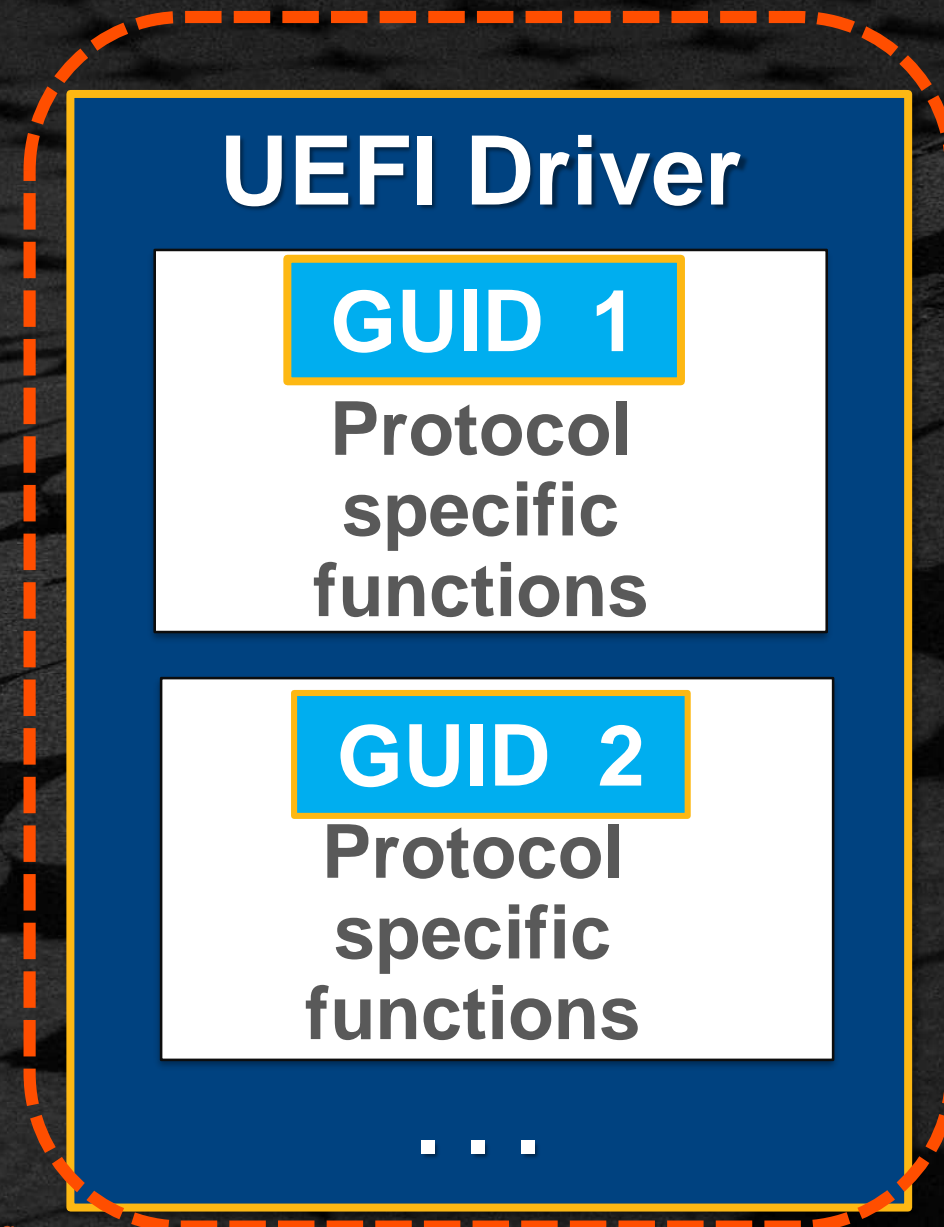
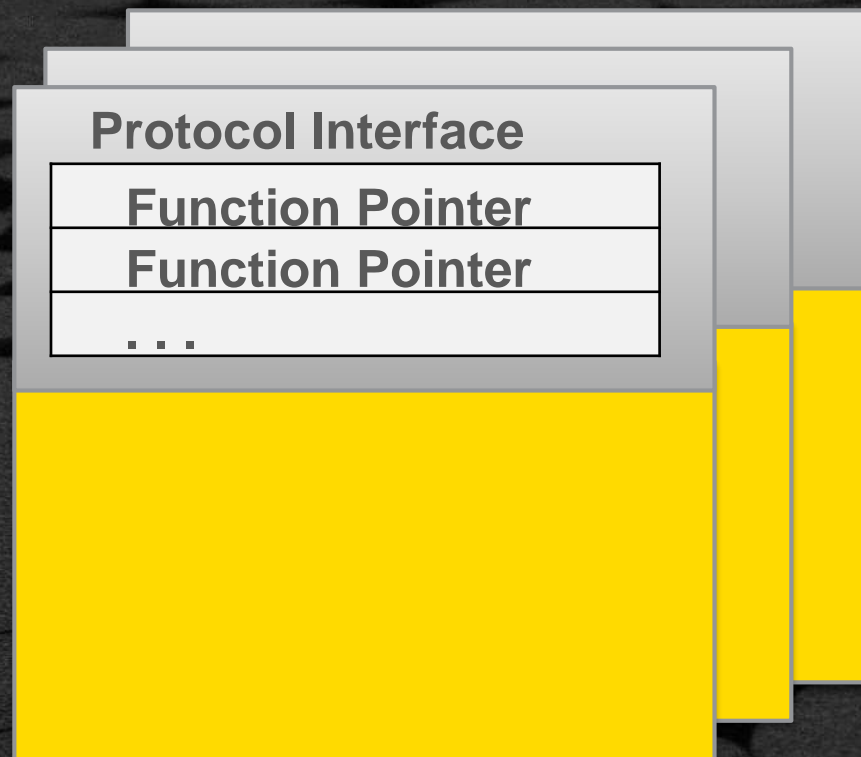
# Drivers Produce Protocols



## Construction of a protocol

InstallProtocolInterface

Handle  
Database



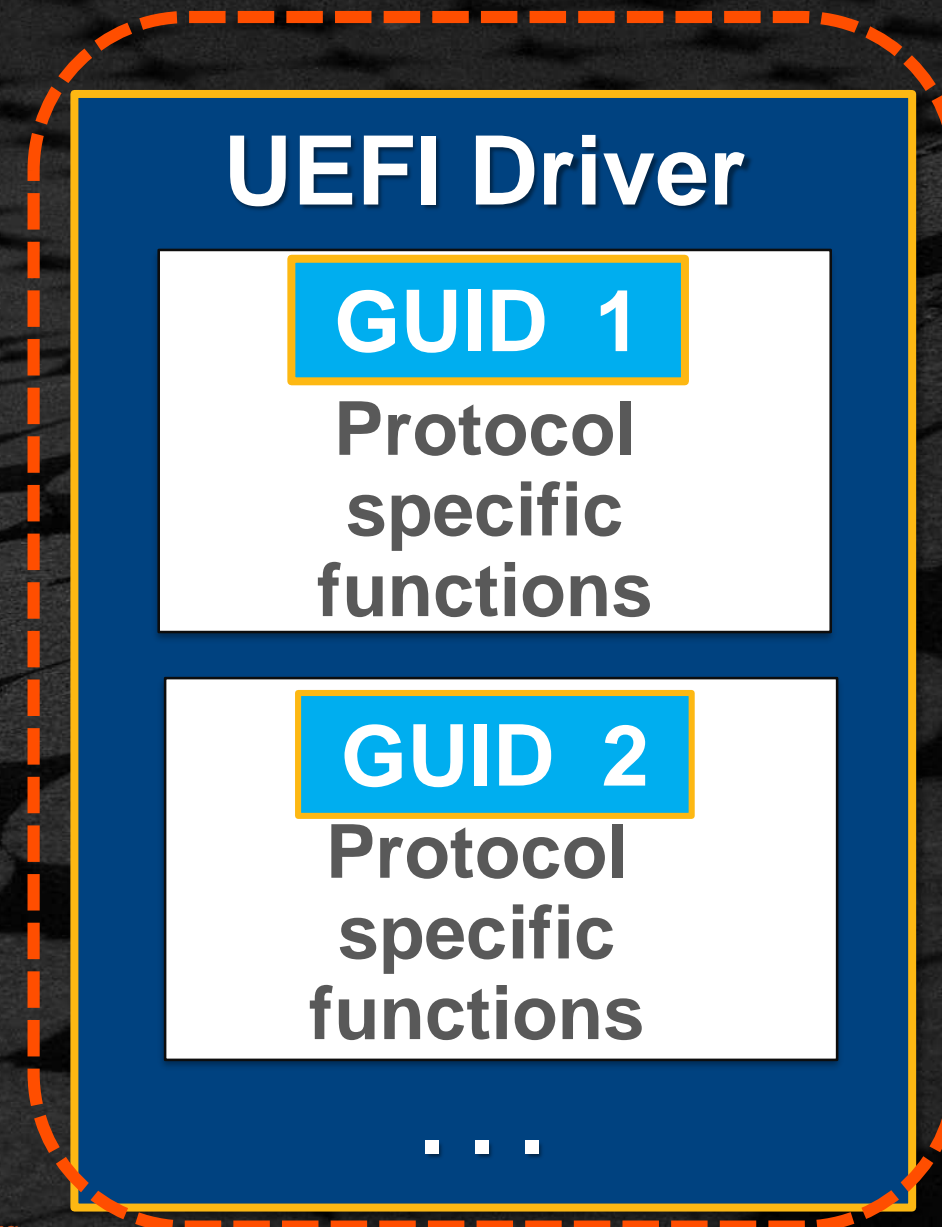
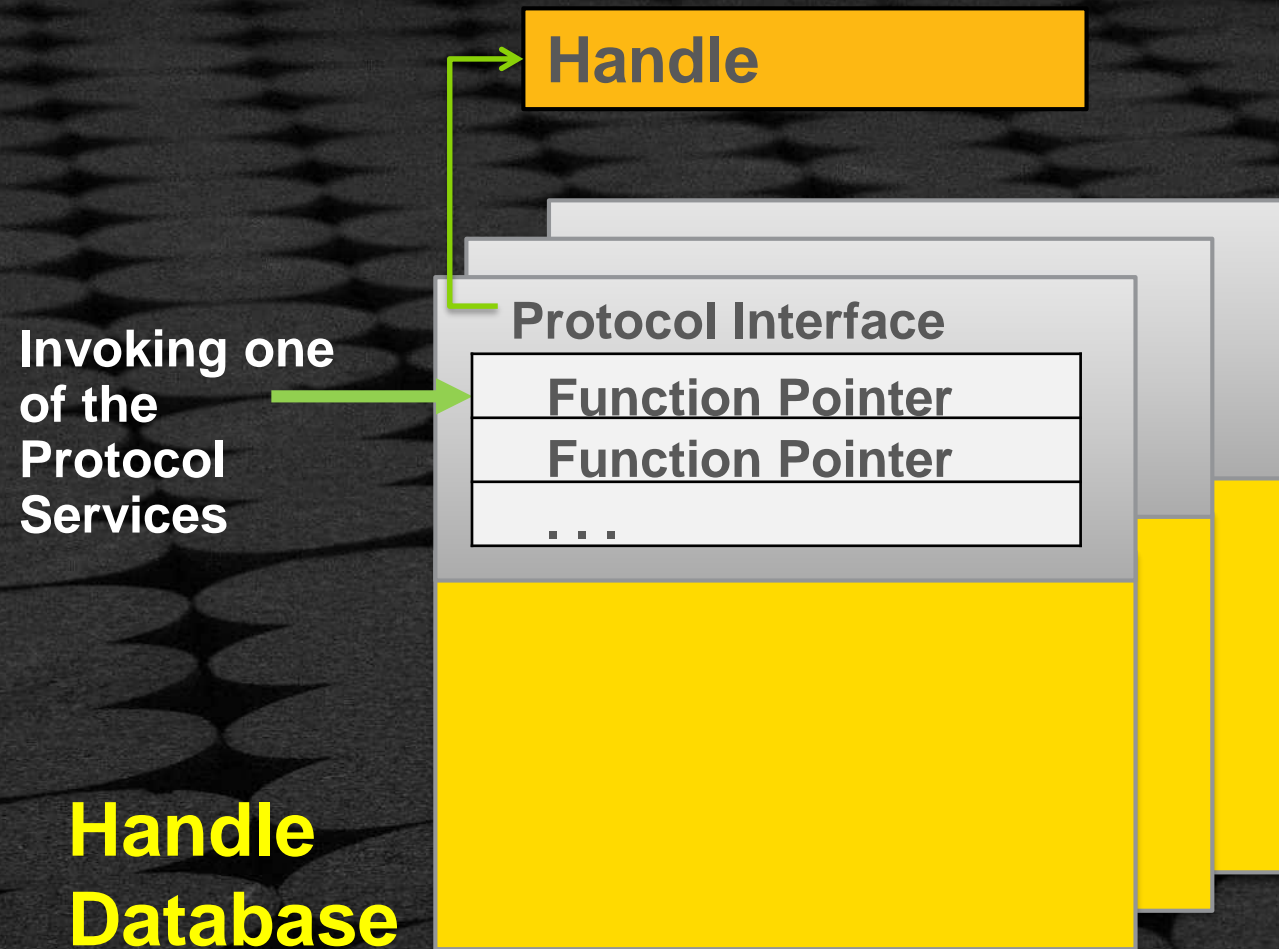


# Drivers Produce Protocols

## Construction of a protocol

InstallProtocolInterface

HandleProtocol(GUID, ...)

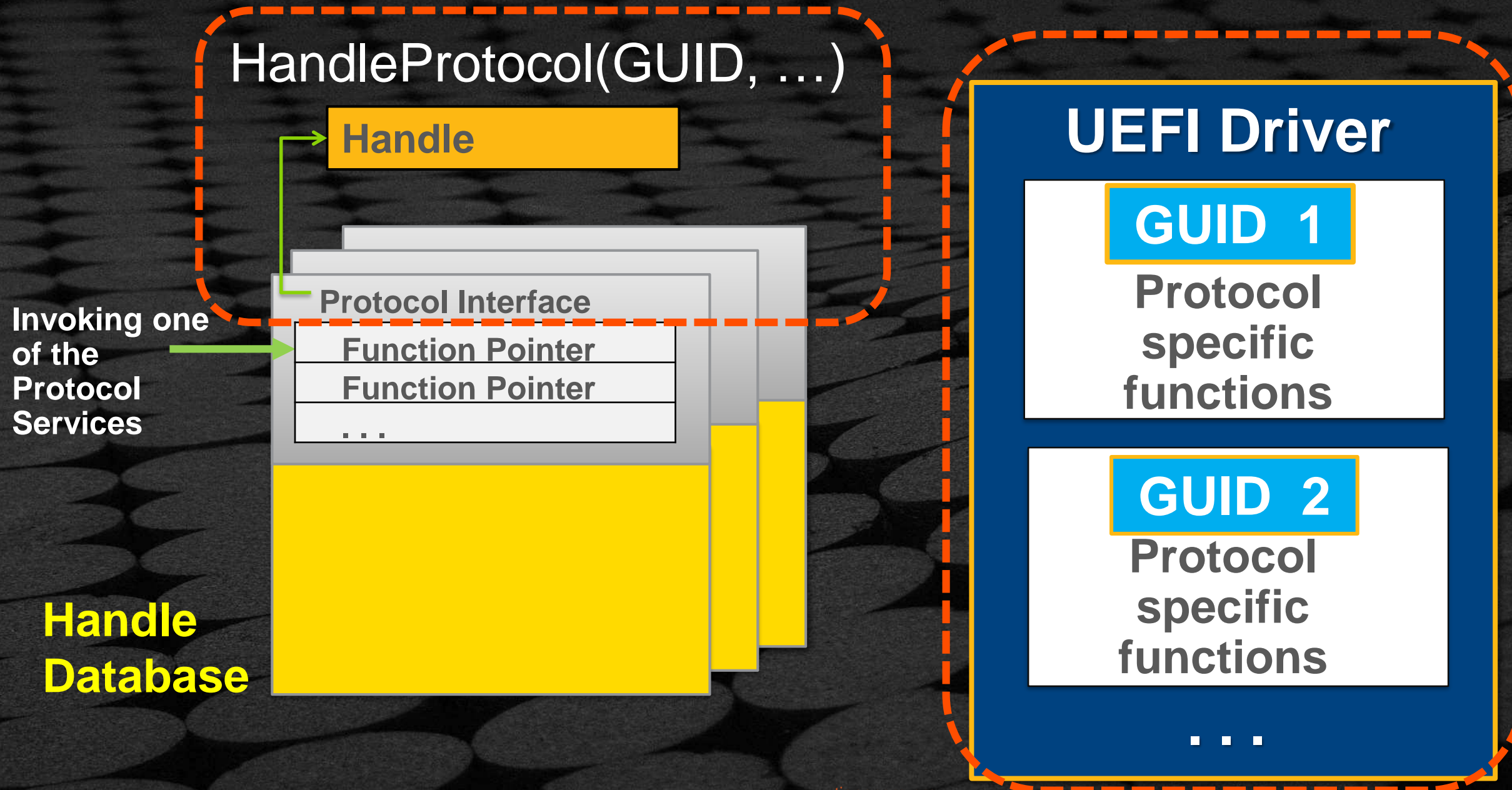




# Drivers Produce Protocols

## Construction of a protocol

InstallProtocolInterface

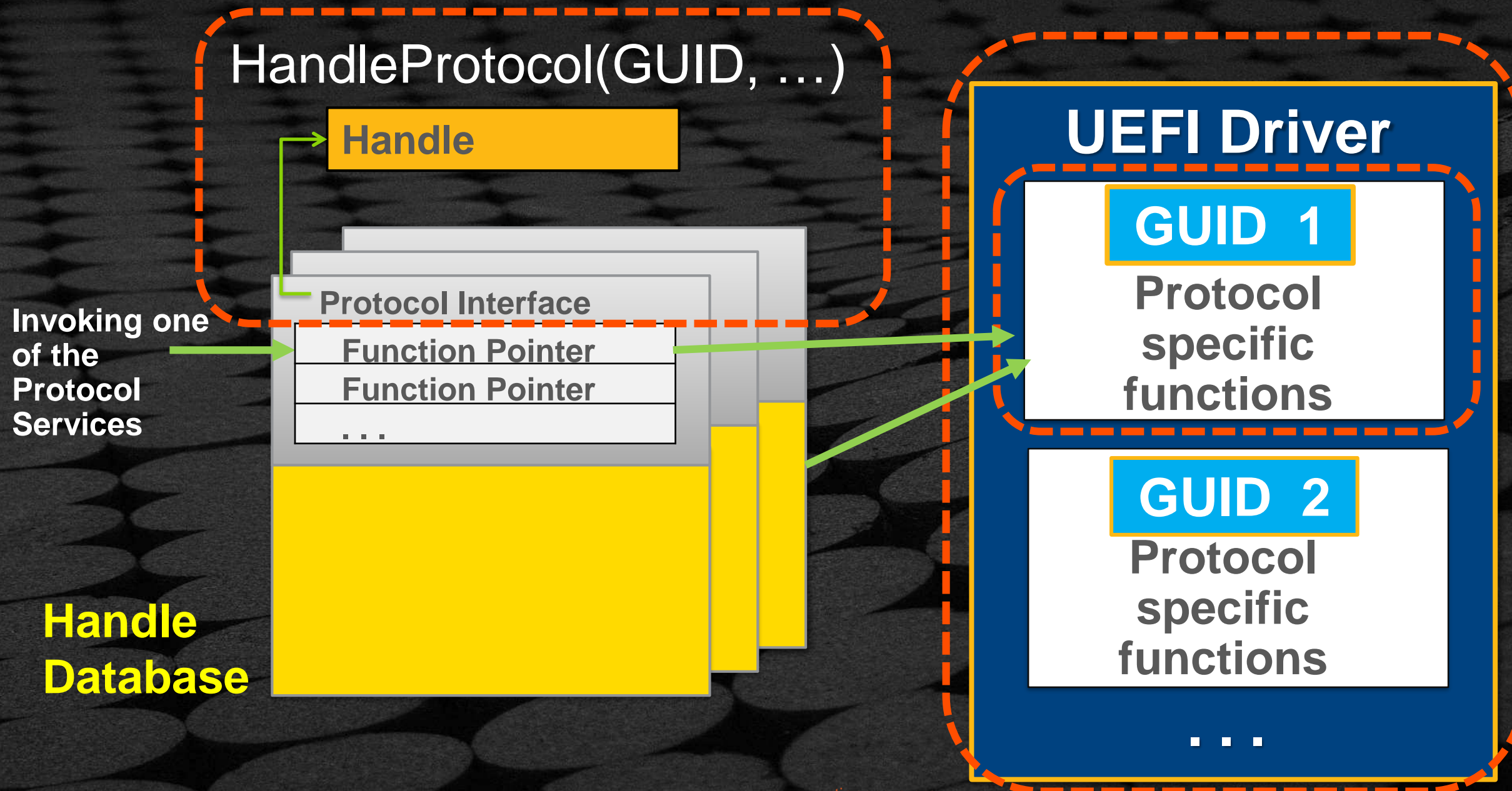




# Drivers Produce Protocols

## Construction of a protocol

InstallProtocolInterface

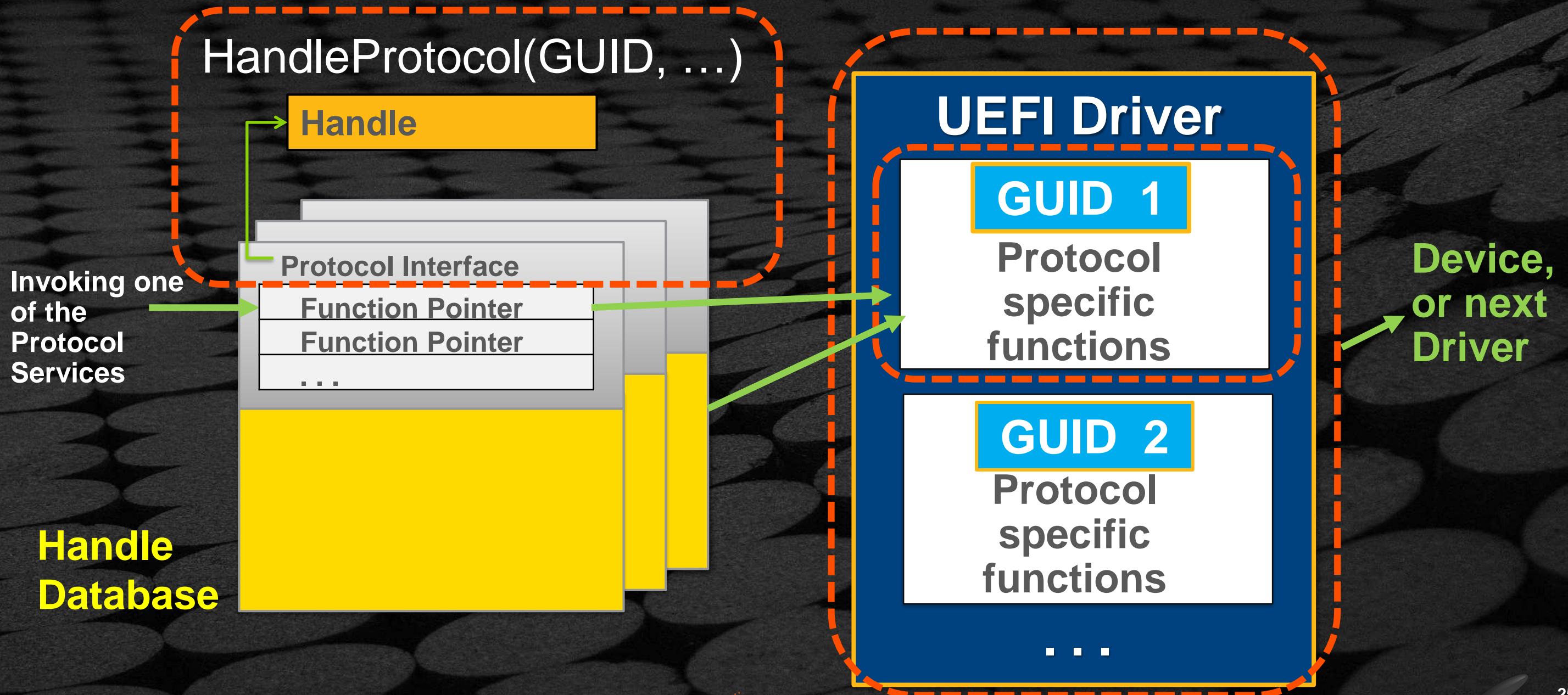




# Drivers Produce Protocols

## Construction of a protocol

InstallProtocolInterface





# UEFI Driver Binding Protocol



## Supported()

Determines if a driver supports a controller



## Start()

Starts a driver on a controller & Installs Protocols



## Stop()

Stops a driver from managing a controller



# Supported - PCI Controller Device Handle

## PCI Controller Device Handle

EFI\_DEVICE\_PATH\_PROTOCOL

EFI\_PCI\_IO\_PROTOCOL

### Tasks

1. **Opens** PCI\_IO Protocol
2. Checks
3. **Closes** PCI\_IO Protocol
4. Returns: *Supported* or *Not Supported*

See § 10.1 UEFI 2.x Spec.

### Inputs:

- “This”
- Controller to manage
- Remaining Device Path

### Supported()

- Checks to see if a driver supports a controller
- Check should not change hardware state of controller
- Minimize execution time, move complex I/O to Start()
- May be called for controller that is already managed
- Child is optionally specified



# Start - PCI Controller Device Handle

## PCI Controller Device Handle

EFI\_DEVICE\_PATH\_PROTOCOL

EFI\_PCI\_IO\_PROTOCOL

EFI\_BLOCK\_IO\_PROTOCOL

### Inputs:

- “This”
- Controller to manage,
- Remaining Device Path

### Start()

- **Opens** PCI I/O
- Starts a driver on a controller
- Can create ALL child handles or ONE child handle



# Stop - PCI Controller Device Handle

## PCI Controller Device Handle

EFI\_DEVICE\_PATH\_PROTOCOL

EFI\_PCI\_IO\_PROTOCOL

EFI\_BLOCK\_IO\_PROTOCOL

### Inputs:

- “This”
- Controller to manage,
- Remaining Device Path

### Stop()

- **Closes** PCI I/O
- Stops a driver from managing a controller
- Destroys all specified child handles
- If no children specified, controller is stopped
- Stopping a bus controller requires 2 calls
  - One call to stop the children. A second call to stop the bus controller itself



# Stop - PCI Controller Device Handle

## PCI Controller Device Handle

EFI\_DEVICE\_PATH\_PROTOCOL

EFI\_PCI\_IO\_PROTOCOL

### Inputs:

- “This”
- Controller to manage,
- Remaining Device Path

### Stop()

- **Closes** PCI I/O
- Stops a driver from managing a controller
- Destroys all specified child handles
- If no children specified, controller is stopped
- Stopping a bus controller requires 2 calls
  - One call to stop the children. A second call to stop the bus controller itself



# UEFI DRIVER EXAMPLE

Examine details of the UEFI Driver - ScsiDiskDxe



# Example: UEFI Driver - ScsiDiskDxe



[edk2/MdeModulePkg/Bus/Scsi/ScsiDiskDxe](#)

- ScsiDiskDxe.inf
- ScsiDisk.c
- ScsiDisk.h



# Example: UEFI Driver - ScsiDiskDxe



[edk2/MdeModulePkg/Bus/Scsi/ScsiDiskDxe](https://github.com/tianocore/edk2/MdeModulePkg/Bus/Scsi/ScsiDiskDxe)

- ScsiDiskDxe.inf
- ScsiDisk.c
- ScsiDisk.h



.inf

 [.inf] Entry, Global Protocols



# Example: UEFI Driver - ScsiDiskDxe

```
[Defines]
  INF_VERSION           = 0x00010005
  BASE_NAME             = ScsiDisk
  MODULE_UNI_FILE       = ScsiDisk.uni
  FILE_GUID             = 0A66E322-3740-4cce-AD62-BD172CECCA35
  MODULE_TYPE           = UEFI_DRIVER
  VERSION_STRING        = 1.0

  ENTRY_POINT           = InitializeScsiDisk

[Sources]
  ComponentName.c
  ScsiDisk.c
  ScsiDisk.h

[Packages]
  MdePkg/MdePkg.dec
```

[Link to .inf](#) - Entry point function InitializeScsiDisk  
Guids and Protocols



# Example: UEFI Driver - ScsiDiskDxe



[edk2/MdeModulePkg/Bus/Scsi/ScsiDiskDxe](https://github.com/tianocore/edk2/MdeModulePkg/Bus/Scsi/ScsiDiskDxe)

- ScsiDiskDxe.inf
- ScsiDisk.c
- ScsiDisk.h

.inf

.h

 [.inf] Entry, Global Protocols

 [.h] Driver's Private Data Structure declaration



# Example: ScsiDisk.h

```
#ifndef _SCSI_DISK_H_  
#define _SCSI_DISK_H_  
  
#include <Uefi.h>  
  
#include <Protocol/ScsiIo.h>  
#include <Protocol/ComponentName.h>  
#include <Protocol/BlockIo.h>  
#include <Protocol/BlockIo2.h>  
#include <Protocol/EraseBlock.h>  
#include <Protocol/DriverBinding.h>  
#include <Protocol/ScsiPassThruExt.h>  
#include <Protocol/ScsiPassThru.h>  
#include <Protocol/DiskInfo.h>
```

[Link to ScsiDisk.h](#) UEFI Driver's Private Data Structure declaration

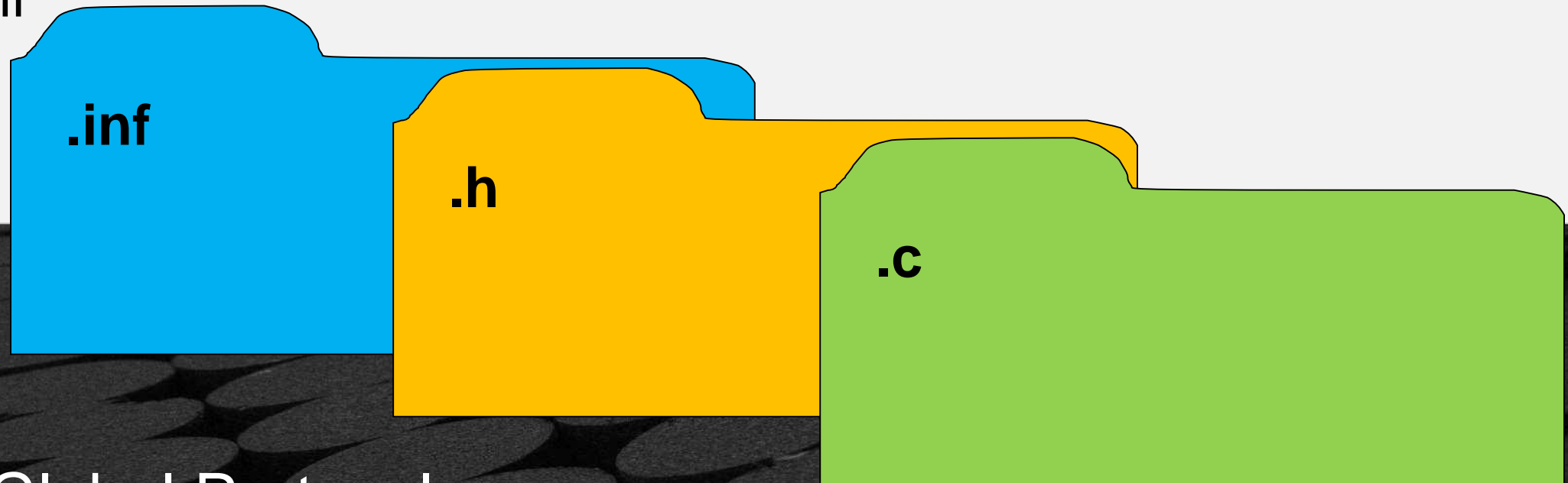


# Example: UEFI Driver - ScsiDiskDxe



[edk2/MdeModulePkg/Bus/Scsi/ScsiDiskDxe](https://github.com/tianocore/edk2/MdeModulePkg/Bus/Scsi/ScsiDiskDxe)

- ScsiDiskDxe.inf
- ScsiDisk.c
- ScsiDisk.h



-  [.inf] Entry, Global Protocols
-  [.h] Driver's Private Data Structure declaration
-  [.c] Review the Supported, Start and Stop functions



# Example: ScsiDisk.c

```
#include "ScsiDisk.h"
```

```
EFI_DRIVER_BINDING_PROTOCOL gScsiDiskDriverBinding = {  
    ScsiDiskDriverBindingSupported,  
    ScsiDiskDriverBindingStart,  
    ScsiDiskDriverBindingStop,  
    0xa,  
    NULL,  
    NULL  
};
```

```
EFI_DISK_INFO_PROTOCOL gScsiDiskInfoProtocolTemplate = {  
    EFI_DISK_INFO_SCSI_INTERFACE_GUID,  
    ScsiDiskInfoInquiry,  
    ScsiDiskInfoIdentify,  
    ScsiDiskInfoSenseData,  
    ScsiDiskInfoWhichIde  
};
```

Review the Initialization Entry point,  
Supported, Start and Stop functions

[Link to ScsiDisk.c](#)



# SUMMARY

- ★ UEFI Drivers manage HW and extend the Firmware
- ★ The UEFI Driver Binding Protocol: Supported, Start and Stop
- ★ Example of UEFI Driver ScsiDisk Driver



# Questions?





# Return to Main Training Page



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



