

UEFI & EDK II TRAINING

UEFI SHELL APPLICATION

tianocore.org



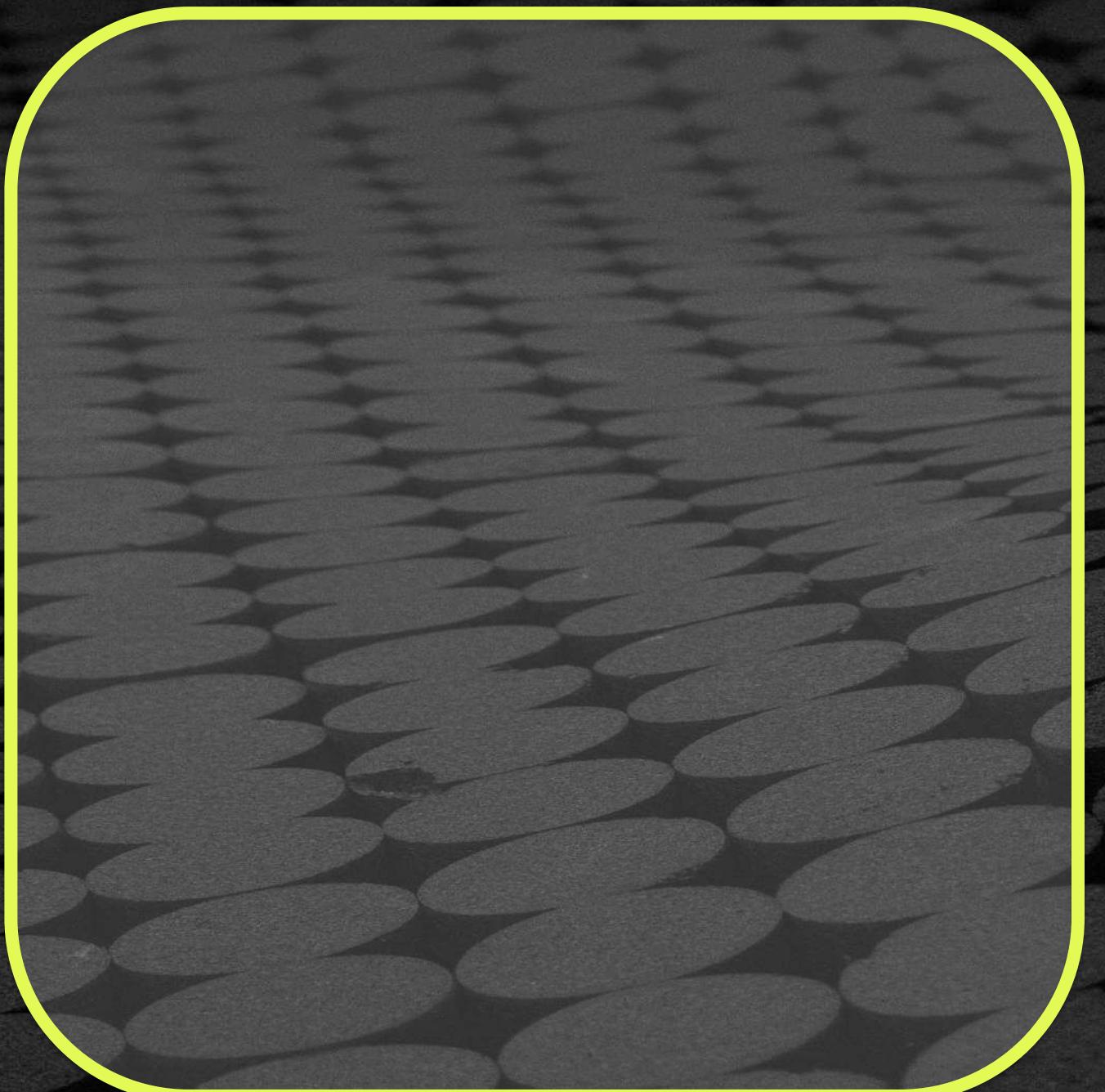
LESSON OBJECTIVE

- * Explain UEFI, the shell, and how they work together
- * Define the shell components
- * Use the shell API in a UEFI application
- * UEFI Shell command Library
- * UEFI Shell scripts

UEFI SHELL OVERVIEW

Components of the UEFI Shell

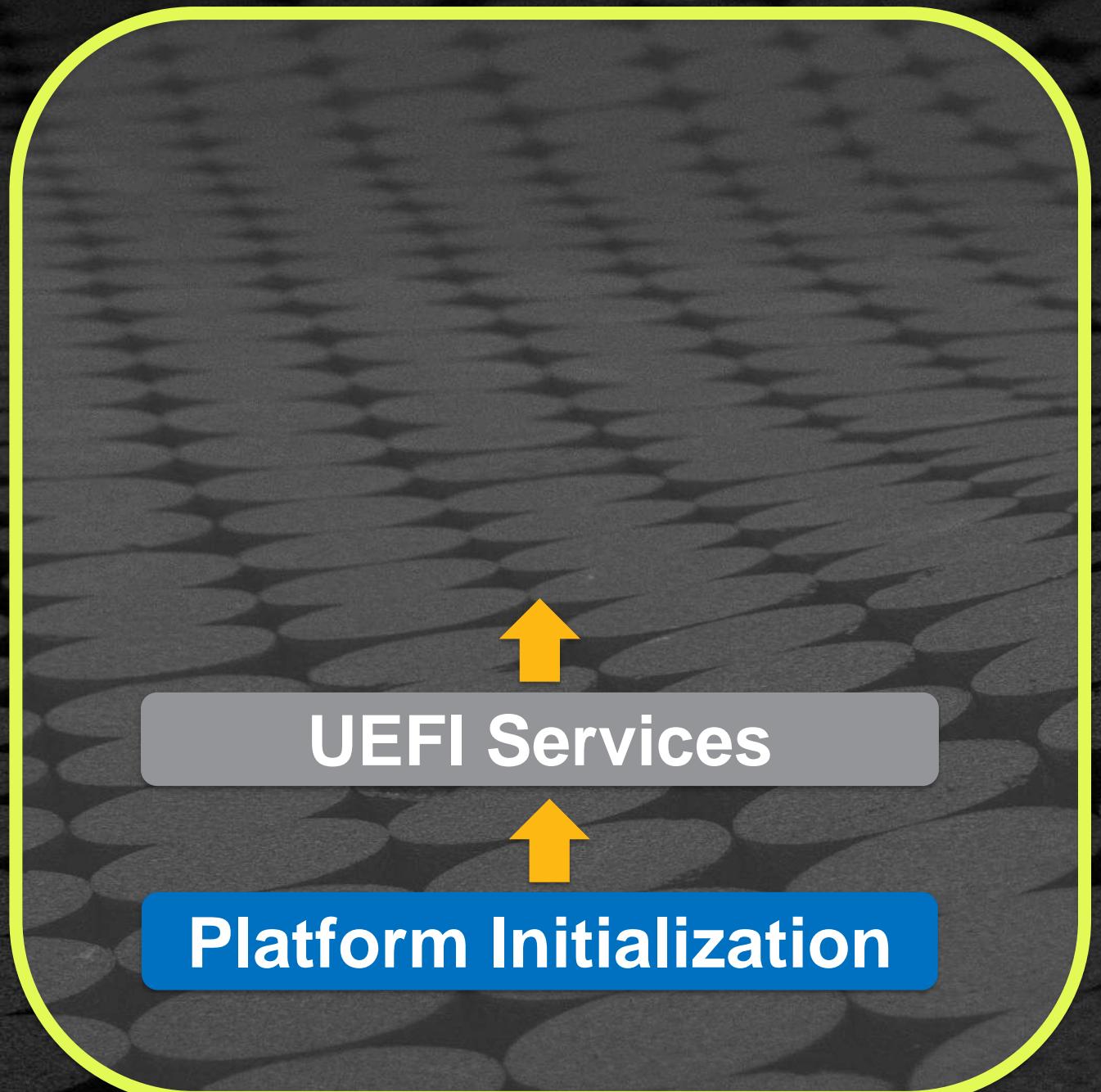
What is a UEFI Shell?



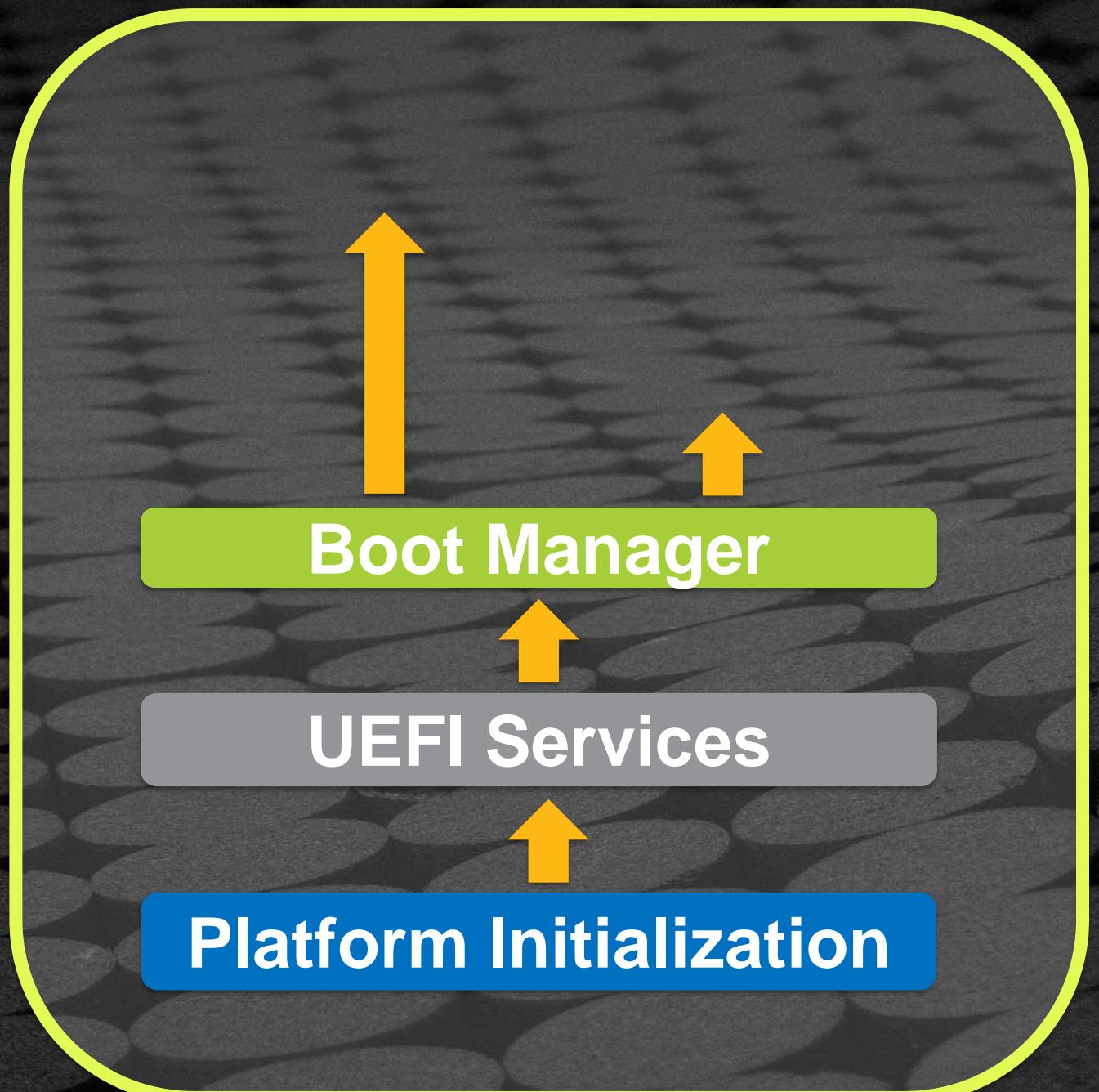
What is a UEFI Shell?



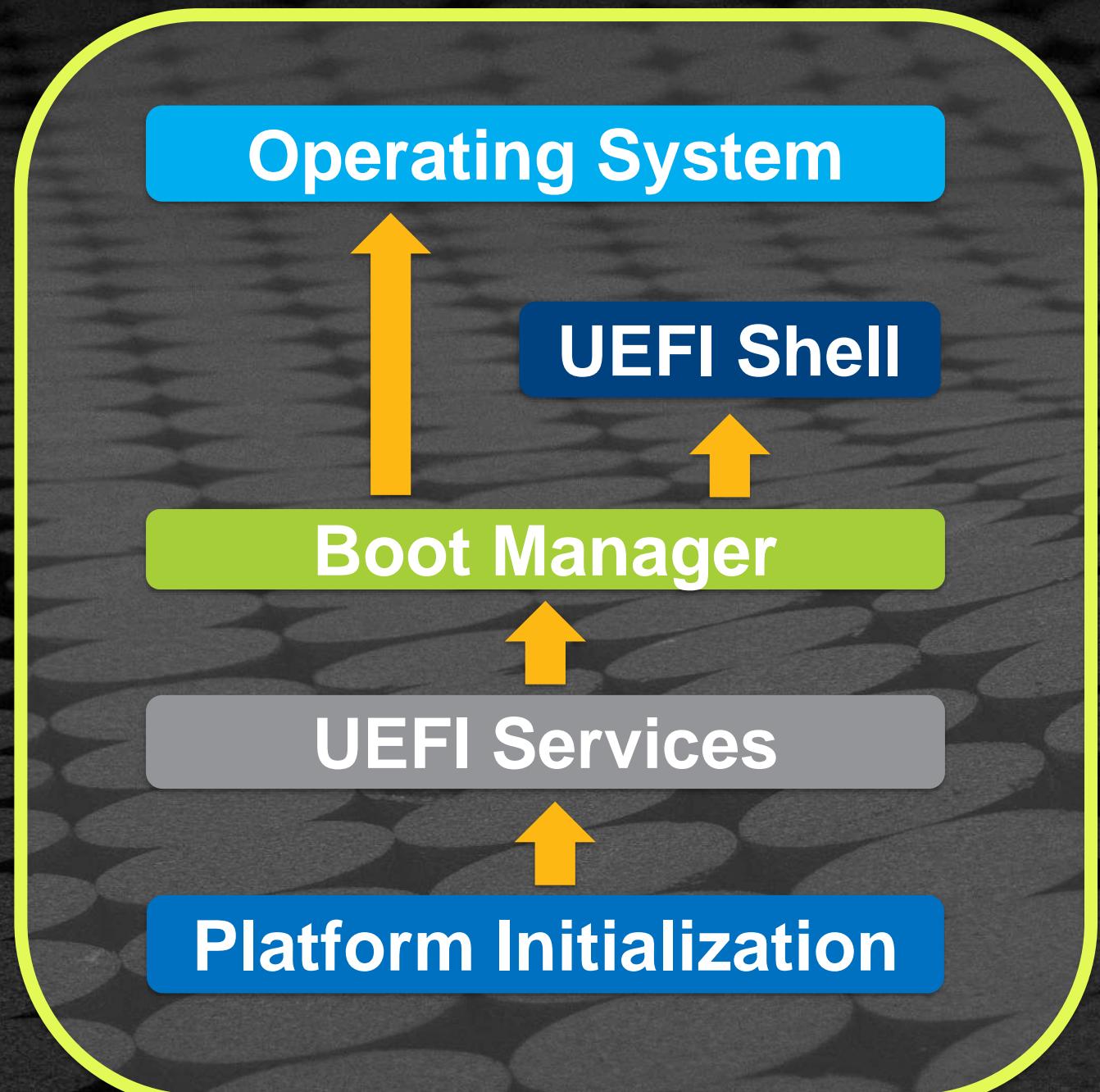
What is a UEFI Shell?



What is a UEFI Shell?



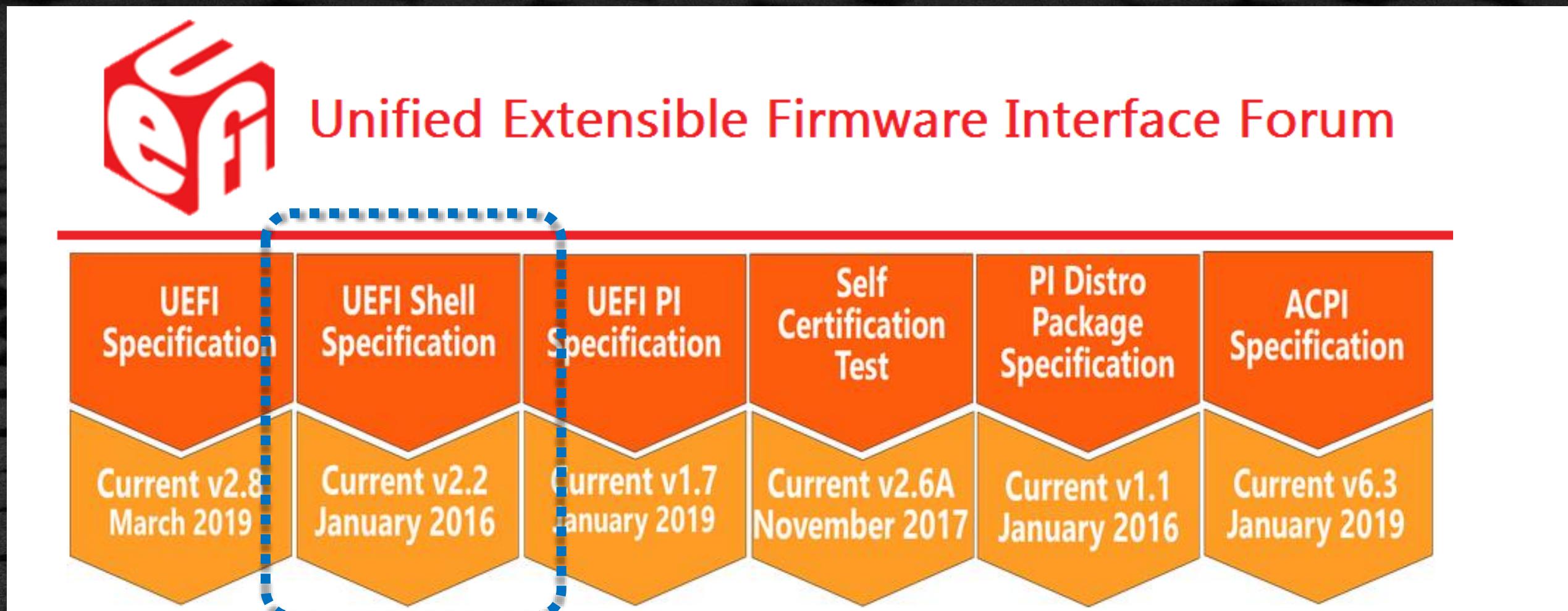
What is a UEFI Shell?



It's an
Extensive &
Standardized
Pre-OS UEFI
Application

UEFI SHELL SPECIFICATION V. 2.2

<http://www.uefi.org/specsandtesttools>



UEFI Shell v2.0 specification first released 2008 – Latest V2.2 Jan 2016

UEFI SHELL ELEMENTS

Small Size
Profiles

Shell
Commands

New Shell API

Enhanced
Scripting

Small Size Profiles

SMALL SIZE PROFILES

Level / Profile	Commands
Level 0	Shell API Only
Level 1	Basic scripting support
Level 2	File Support, cmd(cd, cp, mv)
Level 3	Adds interactive CLI + Profiles <code>bcfg, comp, dblk, dmem, dmpstore,</code> <code>echo, edit,</code> <code>ipconfig, ping</code>
UEFI Debug Profile	
UEFI Network Profile	
UEFI Driver Profile	<code>drvdiag, openinfo, reconnect, load,</code> <code>unload</code>

Choose the shell that best matches your product needs

Shell Commands

help -b

attrib	-Displays or changes the attributes of files or directories.
cd	-Displays or changes the current directory.
cp	-Copies one or more source files or directories to a destination.
load	-Loads a UEFI driver into memory.
map	-Defines a mapping between a user-defined name and a device handle.
mkdir	-Creates one or more new directories.
mv	-Moves one or more files to a destination within a file system.
parse	-Command used to retrieve a value from a particular record which was output in a standard formatted output.
reset	-Resets the system.
set	-Displays, changes or deletes a UEFI Shell environment variables.
ls	-Lists a directory's contents or file information.
rm	-Deletes one or more files or directories.
vol	-Displays the volume information for the file system that is specified by fs.
date	-Displays and sets the current date for the system.
time	-Displays or sets the current time for the system.
timezone	-Displays or sets time zone information.
stall	-Stalls the operation for a specified number of microseconds.
for	-Starts a loop based on for syntax.
goto	-moves around the point of execution in a script.
if	-Controls which script commands will be executed based on provided conditional expressions.
shift	-moves all in-script parameters down 1 number (allows access over 10).

Press ENTER to continue or 'Q' break:

New Shell API

EFI_SHELL_PROTOCOL

Group	Functions
File Manipulation	OpenFileByName(), WriteFile(), etc. . .
Mapping, Alias & Environmental Variables	GetMapFromDevicePath(), GetFilePathFromDevicePath(), etc . . .
Launch Application or Script	Execute(), BatchIsActive(), IsRootShell(),etc
Miscellaneous	GetPageBreak(), EnablePageBreak() ,etc . .

EFI_SHELL_PROTOCOL is installed on each application image handle

ShellPkg Main Libraries

ShellLib

ShellPkg Main Libraries

ShellLib

HandleParsingLib

ShellPkg Main Libraries

ShellLib

HandleParsingLib

ShellCommandLib

Supports binary portability

Shell protocols

Shell parameters

```
#Include <Library/ShellLib.h>
gEfiShellParametersProtocol
gEfiShellProtocol
```

Shell Call Example

```
// use UEFI shell 2.x interface
//
if (gEfiShellParametersProtocol != NULL) {
    Argc = gEfiShellParametersProtocol->Argc;
    Argv = gEfiShellParametersProtocol->Argv;
//Create the file with Argv[1] with
//          read/write/create
    Status = gEfiShellProtocol->OpenFileByName
        (Argv[1], &Handle,
         EFI_FILE_MODE_READ |
         EFI_FILE_MODE_WRITE |
         EFI_FILE_MODE_CREATE);

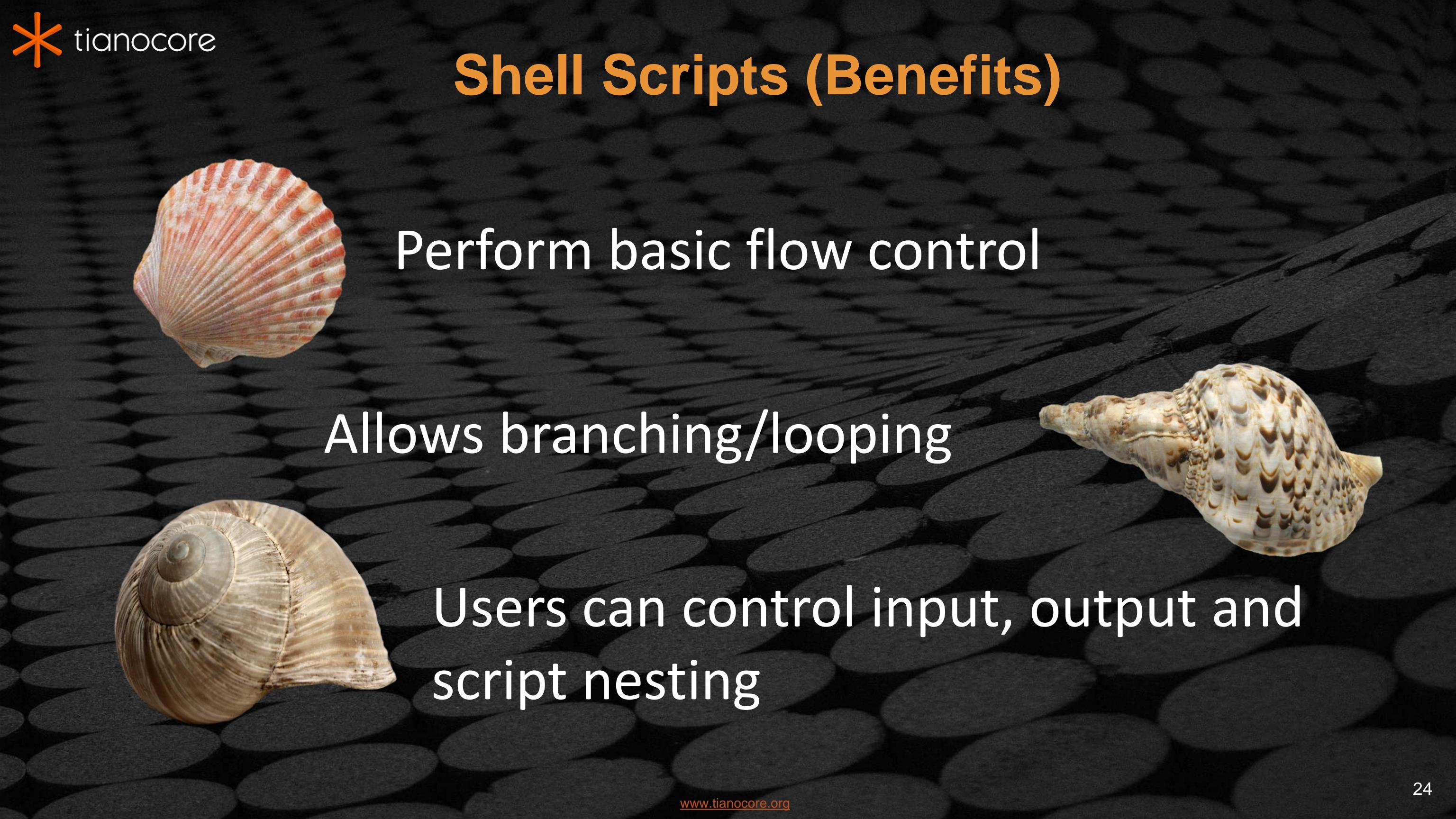
//
// Write the buffer data to the file
    Status = gEfiShellProtocol->WriteFile( Handle,
        (UINTN *)&BufferSize, (void *)Buffer);
```

Enhanced Scripting

Enhanced Scripting

- Contains .nsh extension
- “Startup.nsh” Runs first
- Supports:
 - ✓ Command-line arguments
 - ✓ Standard script commands
 - ✓ Input & output redirection & pipes

Shell Scripts (Benefits)



Perform basic flow control

Allows branching/looping

Users can control input, output and
script nesting

Script that Detects Shell Capabilities

```
# check if Shell supports level 3 commands
# Exit on error
if %uefishellsupport% ult 3 then
    echo Must support UEFI Shell, Level 3
    exit /b 2
endif
# check that Shell supports Debug1 profile.
if profiles(Debug1)then
    echo UEFI Shell supports Debug1 profile
endif
```

UEFI Shell Script Example

Script1.nsh

```
# Simple UEFI Shell script file
echo -off
script2.nsh
if exist %cwd%Mytime.log then
    type Mytime.log
endif
echo "%HThank you." "%VByeBye:) %N"
```

Script2.nsh

```
# Show nested scripts
time > Mytime.log
for %a run (3 1 -1)
    echo %a counting down
endfor
```

Documentation for EDK II ShellPkg



Documentation Link:
[wiki Shell Package](#)

Getting the Shell 2.0

This provides a shell application, a set of NULL-named libraries that provide configurable command sets, and libraries for creating more Shell applications and shell commands. See the [ReadMe](#) for more info.

Source Repository

ShellPkg

This provides source code for the shell applications.

Binary Repository

ShellBinPkg

This provides the binary shell applications. There are a few versions for different usage models. See the [ReadMe](#) for more info.

Shell 2.0 Engineering Resources

- [Shell Execution Requirements](#)
- [Shell Library Primer](#)
- [Creating a Shell Application](#)
- [Porting an EDK Shell Extension](#)
- [Move a Shell Application to internal command](#)
- [Shell FAQ](#)

UEFI Shell 2.2 Vs. EFI Shell 1.0

- UEFI Shell 2.x
 - EFI_SHELL_PARAMETERS_PROTOCOL
- EFI Shell 1.0
 - EFI_SHELL_INTERFACE

```
//  
#include <Protocol/EfiShellInterface.h> //GUID protocol for EFI Shell 1.0  
#include <Protocol/ShellParameters.h>    //GUID protocol for UEFI Shell 2.x  
  
// . . .  
  
EFI_SHELL_PARAMETERS_PROTOCOL      *mEfiShellParametersProtocol;  
EFI_SHELL_INTERFACE                *mEfiShellInterface;  
//
```

See example C file: [MyShellApp.c](#)

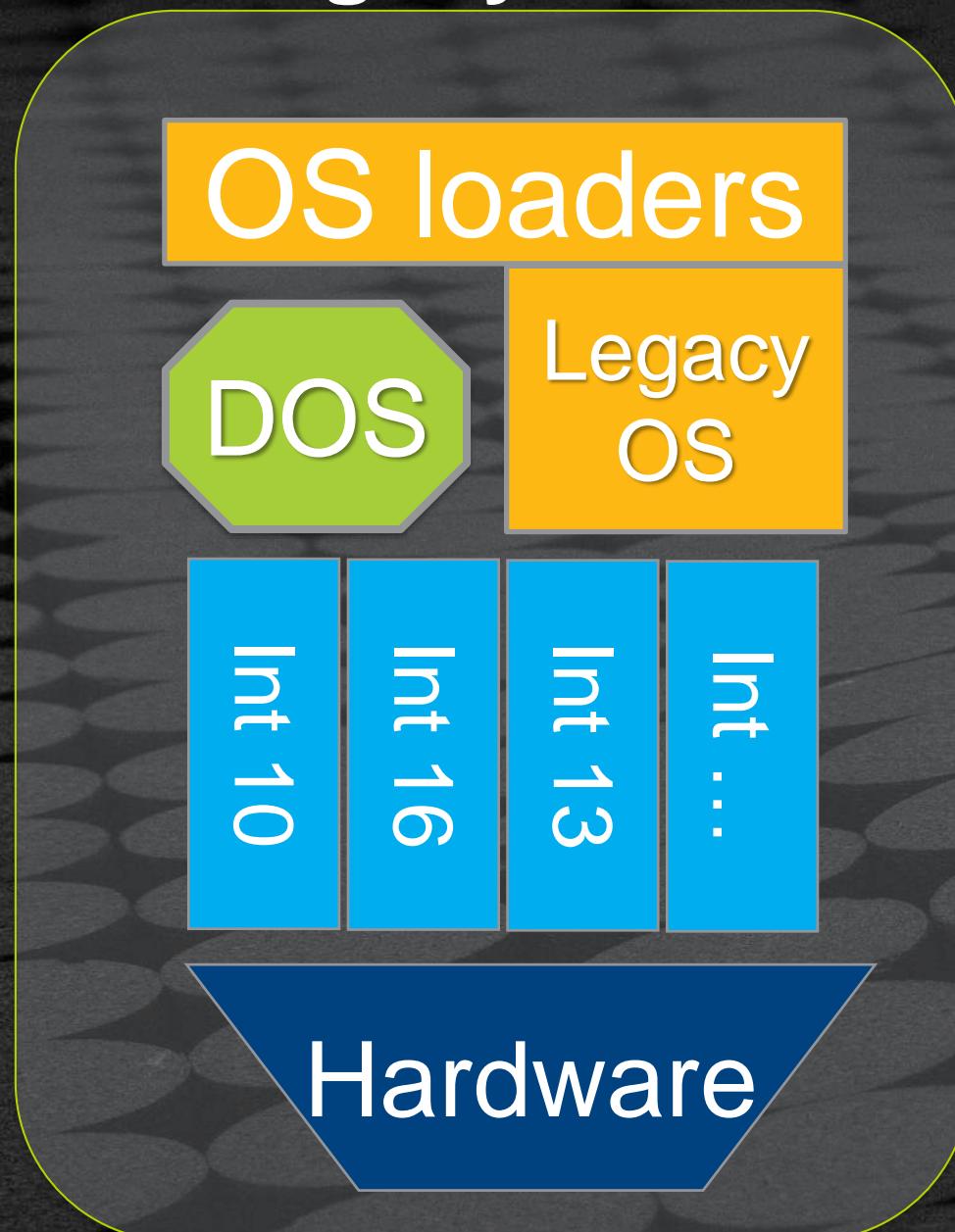
UEFI Shell 2.x Vs. EFI Shell 1.0

```
• • •  
//Check for UEFI Shell 2.x  
Status = gBS->OpenProtocol(ImageHandle,  
                           gEfiShellParametersProtocolGuid,  
                           VOID **)&mEfiShellParametersProtocol,  
                           ImageHandle,  
                           NULL  
                           EFI_OPEN_PROTOCOL_GET_PROTOCOL  
                           );  
if (!EFI_ERROR(Status)) {  
//  
// use UEFI Shell 2.x Parameter Protocol  
//  
| Argc = mEfiShellParametersProtocol->Argc;  
| Argv = mEfiShellParametersProtocol->Argv;  
• • •  
{// Check if EFI shell 1.0 interface
```

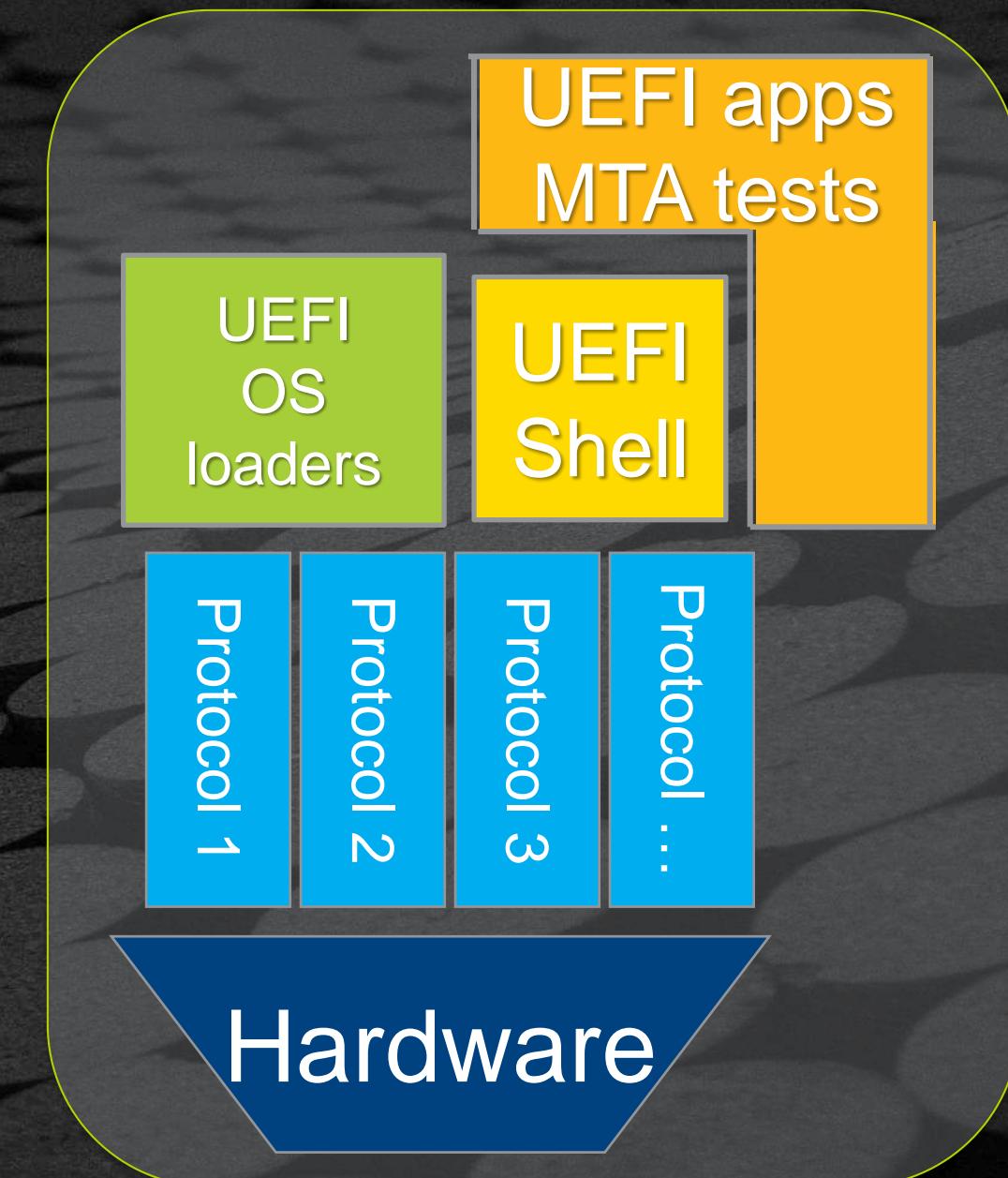
See example C file: [MyShellApp.c](#)

LEGACY VS. UEFI

Legacy BIOS



UEFI



SHELL USAGE



Execute preboot programs

Move files between devices

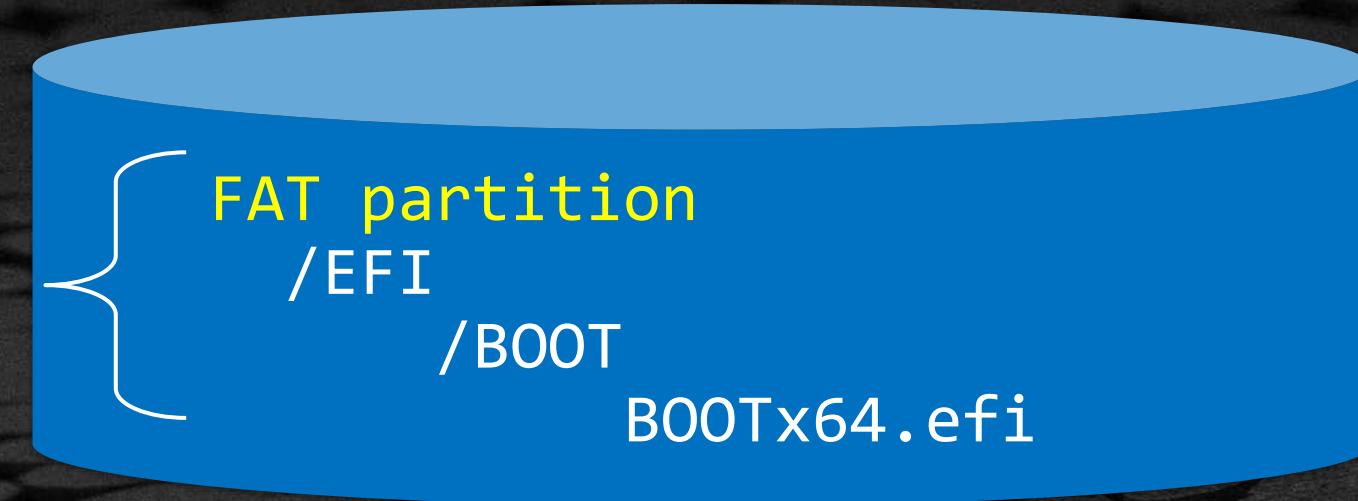


Load a preboot UEFI driver (.efi)

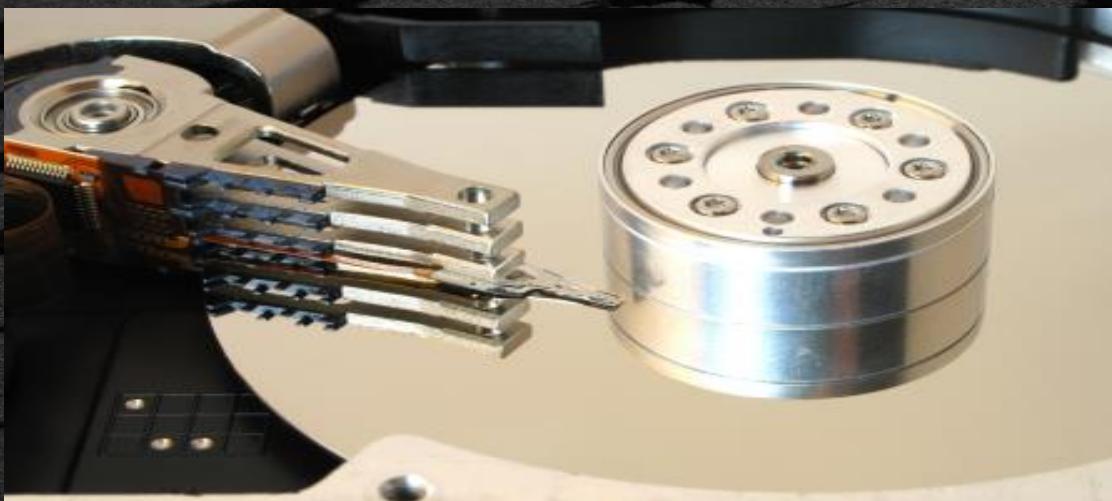


ACCESSING THE SHELL

/EFI/boot/BOOTx64.efi



BOOTx64.efi = OS loader, UEFI application, or UEFI Shell



UEFI File System & Device Path

```
Shell> map
```

```
Device mapping table
```

```
fs0  : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/  
HD(Part1,Sig8983DFE0-F474-01C2-507B-9E5F8078F531)
```

```
blk0 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Slave)
```

```
blk1 : Acpi(PNP0A03,0)/Pci(1F|1)/Ata(Primary,Master)
```

```
blk2 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)
```

```
blk3 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/  
HD(Part1,Sig8983DFE0-F474-01C2-507B-9E5F8078F531)
```

```
blk4 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/  
HD(Part2,Sig898D07A0-F474-01C2-F1B3-12714F758821)
```

```
blk5 : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/Scsi(Pun0,Lun0)/  
HD(Part3,Sig89919B80-F474-01C2-D931-F8428177D974)
```

UEFI File System & Device Path

```
fs0  : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/  
Scsi(Pun0,Lun0)/HD(Part1, Sig8983DFE0-F474  
01C2-507B-9E5F8078F531)
```

UEFI File System & Device Path

```
fs0  : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/  
Scsi(Pun0,Lun0)/HD(Part1, Sig8983DFE0-F474  
01C2-507B-9E5F8078F531)
```

- fs0:
- Acpi(PNP0A03,1)
- Pci(1F|0)/Pci(2|0)
- Scsi(Pun0,Lun0)
- HD(Part1,Sig8983DFE0-F474-01C2-507B-9E5F8078F531)

UEFI File System & Device Path

```
fs0  : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/  
Scsi(Pun0,Lun0)/HD(Part1, Sig8983DFE0-F474  
01C2-507B-9E5F8078F531)
```

- fs0:
 - Acpi(PNP0A03,1)
 - Pci(1F|0)/Pci(2|0)
 - Scsi(Pun0,Lun0)
 - HD(Part1,Sig8983DFE0-F474-01C2-507B-9E5F8078F531)

UEFI File System & Device Path

```
fs0  : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/  
Scsi(Pun0,Lun0)/HD(Part1, Sig8983DFE0-F474  
01C2-507B-9E5F8078F531)
```

- fs0:
- Acpi(PNP0A03,1)
- Pci(1F|0)/Pci(2|0)
- Scsi(Pun0,Lun0)
- HD(Part1,Sig8983DFE0-F474-01C2-507B-9E5F8078F531)

UEFI File System & Device Path

```
fs0  : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/  
Scsi(Pun0,Lun0)/HD(Part1, Sig8983DFE0-F474  
01C2-507B-9E5F8078F531)
```

- fs0:
- Acpi(PNP0A03,1)
- Pci(1F|0)/Pci(2|0)
- Scsi(Pun0,Lun0)
- HD(Part1,Sig8983DFE0-F474-01C2-507B-9E5F8078F531)

UEFI File System & Device Path

```
fs0  : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/  
Scsi(Pun0,Lun0)/HD(Part1, Sig8983DFE0-F474  
01C2-507B-9E5F8078F531)
```

- fs0:
- Acpi(PNP0A03,1)
- Pci(1F|0)/Pci(2|0)
- Scsi(Pun0,Lun0)
- HD(Part1,Sig8983DFE0-F474-01C2-507B-9E5F8078F531)

UEFI File System & Device Path

```
fs0  : Acpi(PNP0A03,1)/Pci(1F|0)/Pci(2|0)/  
Scsi(Pun0,Lun0)/HD(Part1, Sig8983DFE0-F474  
01C2-507B-9E5F8078F531)
```

- fs0:
- Acpi(PNP0A03,1)
- Pci(1F|0)/Pci(2|0)
- Scsi(Pun0,Lun0)
- HD(Part1,Sig8983DFE0-F474-01C2-507B-9E5F8078F531)

EFI Variable `BOOT0000` == *Some Device Path*

SUMMARY

- ★ Explain UEFI, the shell, and how they work together
- ★ Define the shell components
- ★ Use the shell API in a UEFI application
- ★ UEFI Shell command Library
- ★ UEFI Shell scripts

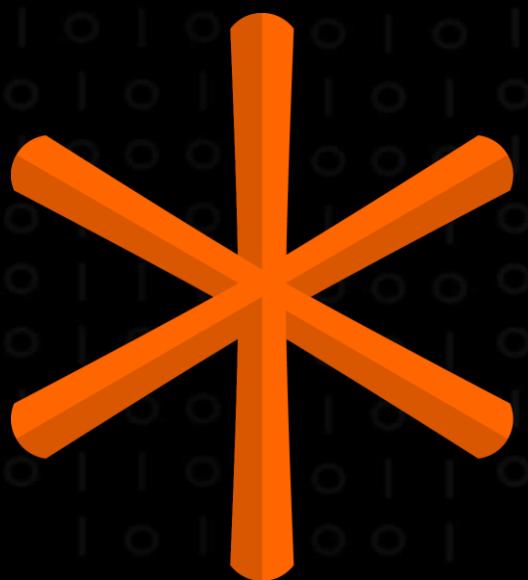
Questions?



RETURN TO MAIN TRAINING PAGE



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



tianocore

