tianocore

UEFI & EDK II Training

UEFI AND PLATFORM INITIALIZATION (PI) BOOT FLOW & OVERVIEW

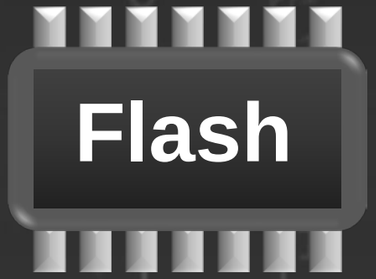
tianocore.org

LESSON OBJECTIVE

- Where is the System Firmware
- Review UEFI Platform Initialization Boot Flow Process
- What about Management Mode (Formerly Known as SMM)
- What is Intel® Firmware Support Package (Intel® FSP)
- The UEFI.org Forum & Tianocore.org

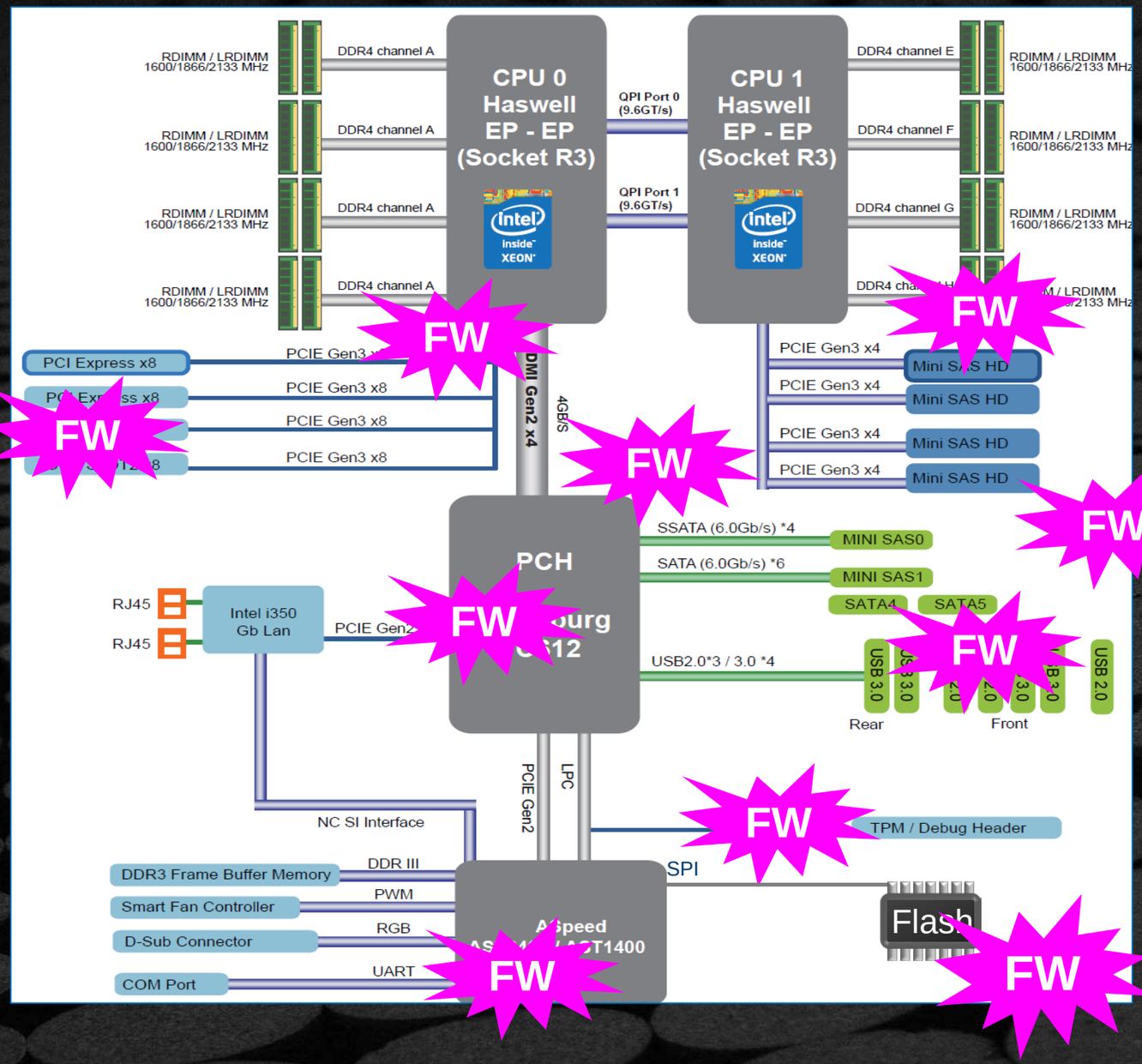
WHERE IS THE FIRMWARE

Where is the UEFI Firmware on a platform



tianocore

Firmware is Everywhere

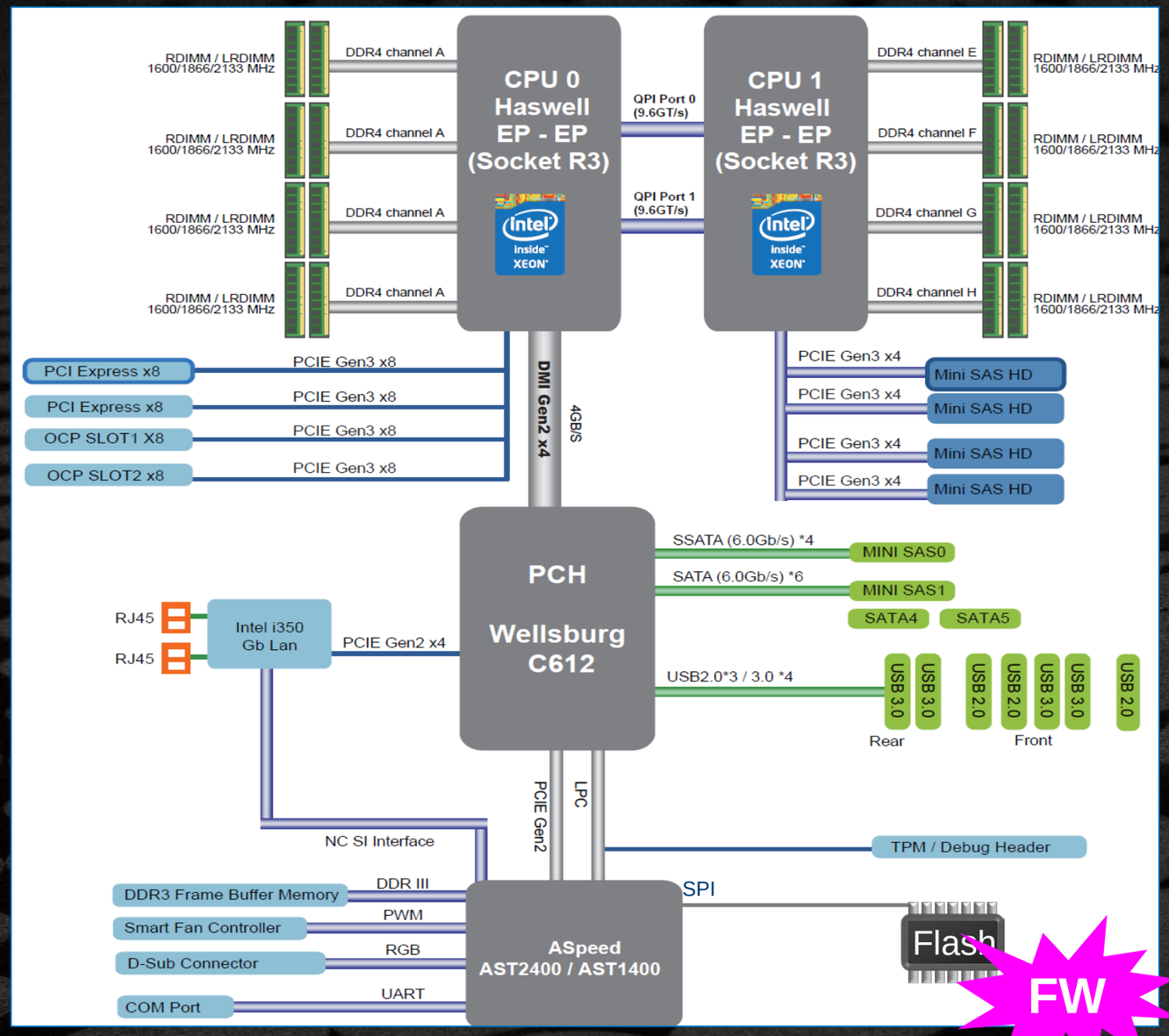


- GBe NIC, WiFi, Bluetooth, WiGig
- Baseband (3G, LTE) Modems
- Sensor Hubs
- NFC, GPS Controllers
- HDD/SSD
- Keyboard and Embedded Controllers
- Battery Gauge
- Baseboard Management Controllers (BMC)
- Graphics/Video
- USB Thumb Drives, keyboards/mice
- Chargers, adapters
- TPM, security coprocessors
- Routers, network appliances

Main system firmware (BIOS, UEFI firmware, coreboot)

Image source

Firmware is Everywhere



Main system firmware (BIOS, UEFI firmware, coreboot)

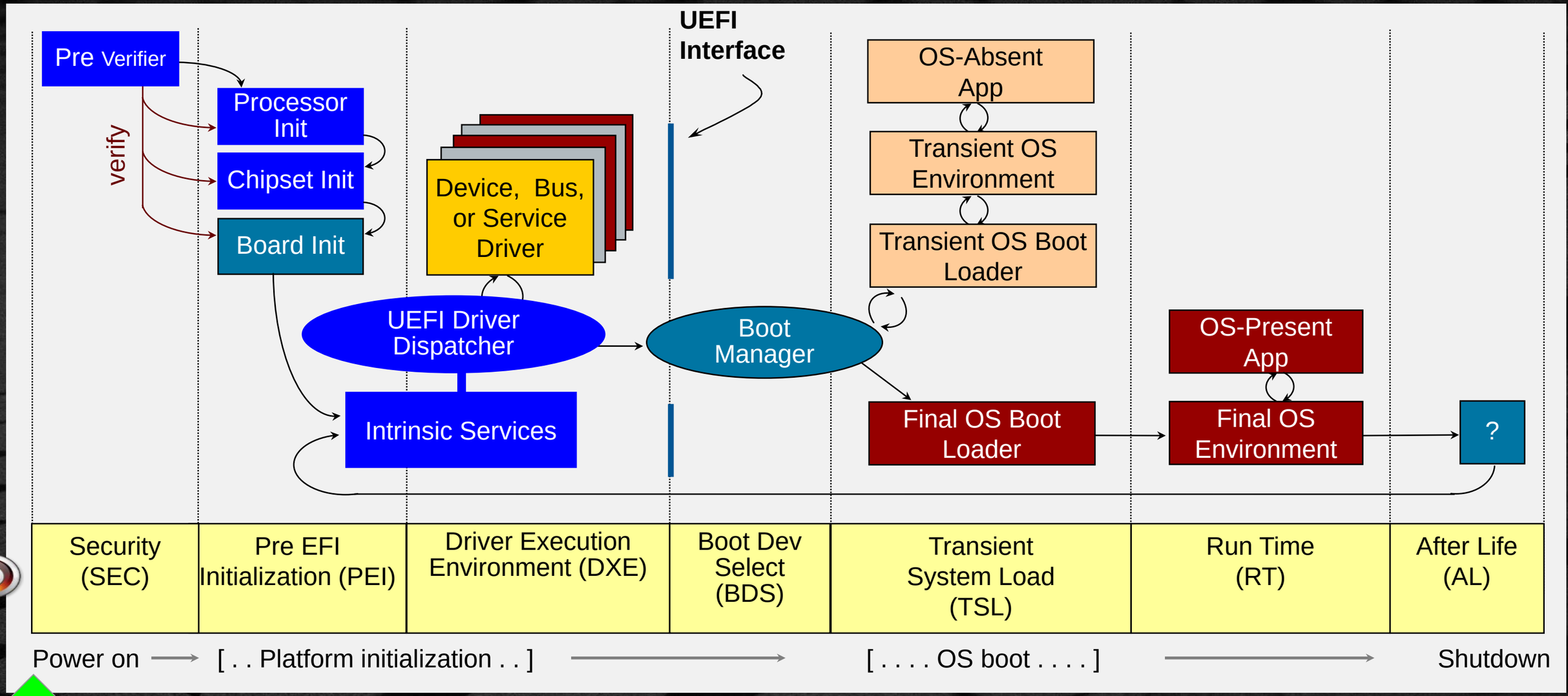
Image source

tianocore

UEFI BOOT EXECUTION FLOW

Starting at the processor reset vector

UEFI – PI & EDK II BOOT FLOW - SEC

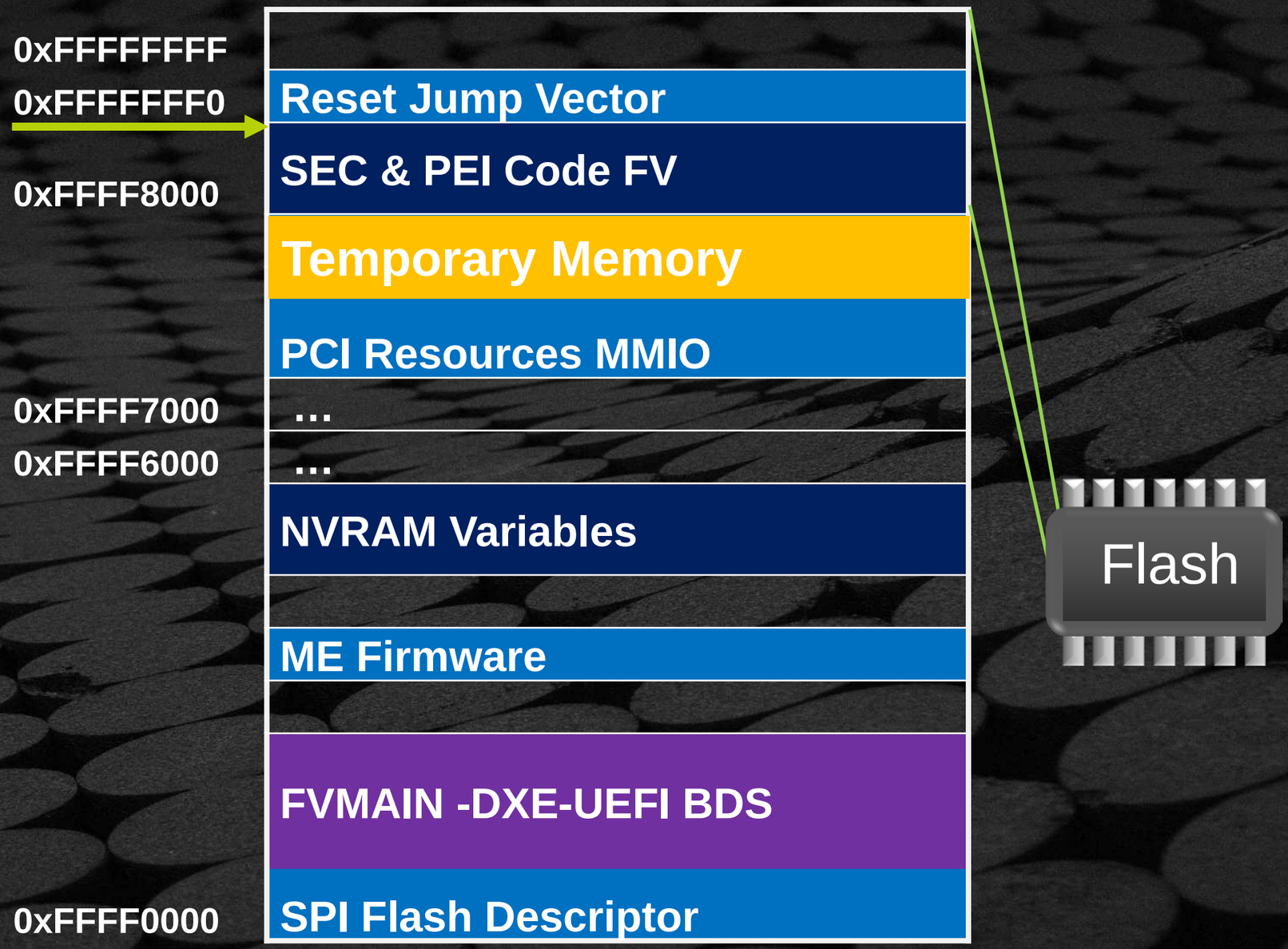


System Reset Vector
Stage 7 on IA

PRE-MEMORY INIT

Address space

Reset 16 bit address
`0xFFFF:0000`



STARTING AT THE RESET VECTOR- SEC

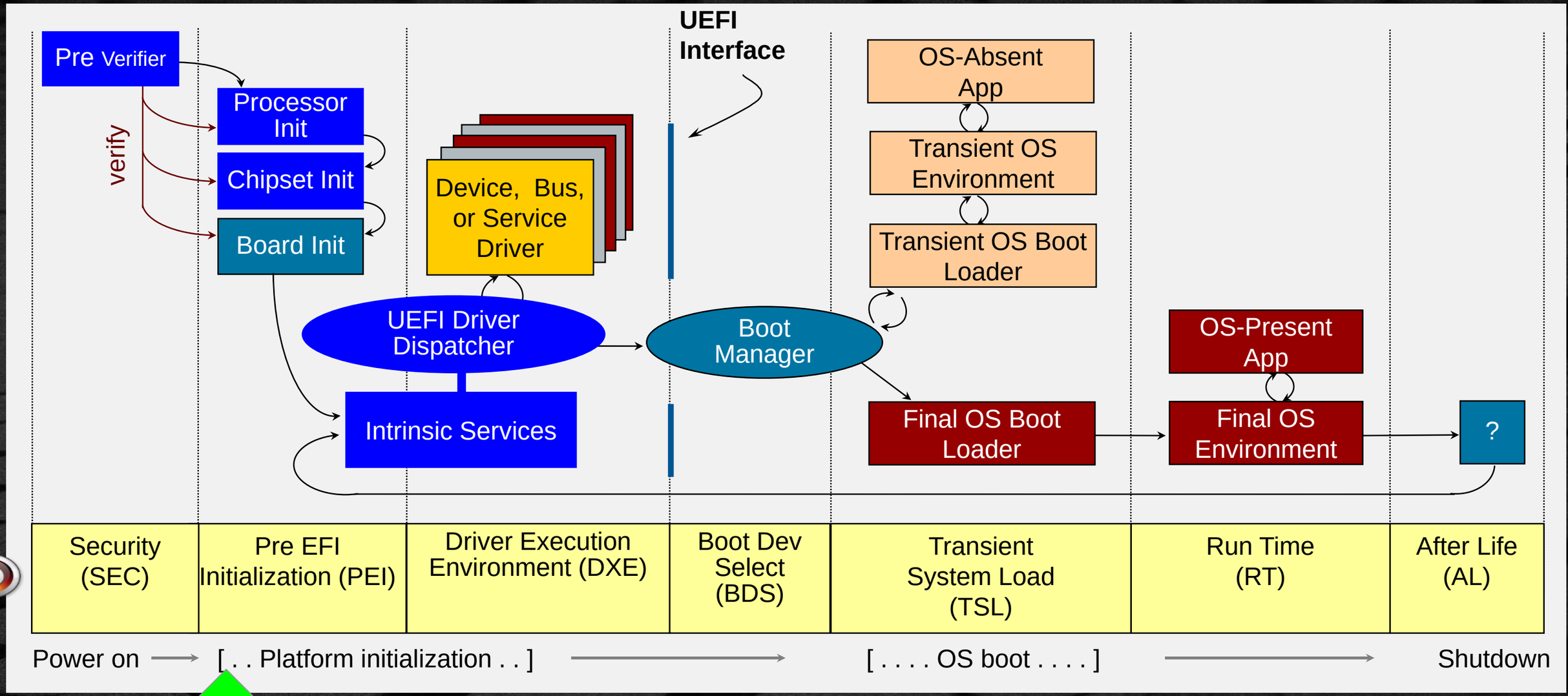
The Processor Executes SEC starting at the first fetch from the reset vector

- SEC Consumes the Reset vector at address space 4GB - 0x10
- Serving as the root of trust
- May choose to authenticate the PEI Foundation
- Initialize the Application Processors (AP) waking stub
- Early microcode update
- Collect BIST (Built-in Self Test)
- Set up TEMP Memory (CAR, NEM)
- Switch to Protected Mode (32 bit flat mode)
- Other characteristics of SEC
  - Executed in place from flash
  - Written in assembly (16-bit & 32-bit) on Intel Architecture
  - BSP is only processor executing (single thread)

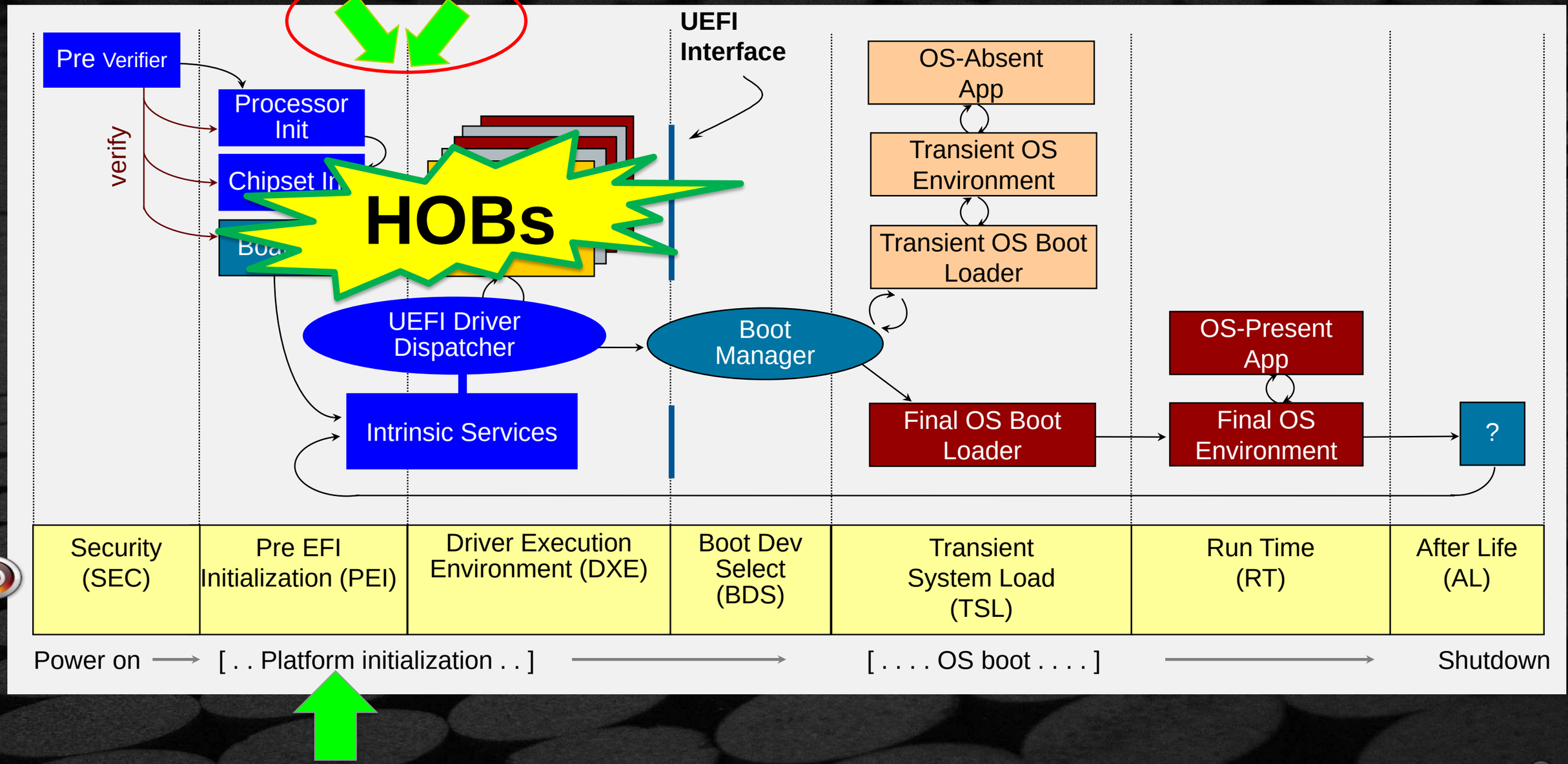
TERMS TO KNOW ABOUT THE FLASH DEVICE

- Firmware Volume (FV)
  - The basic storage with a firmware device
- Firmware File System (FFS)
  - Describes the organization of files within a FV

UEFI – PI & EDK II BOOT FLOW - PEI



UEFI – PI & EDK II BOOT FLOW - DXEIPL

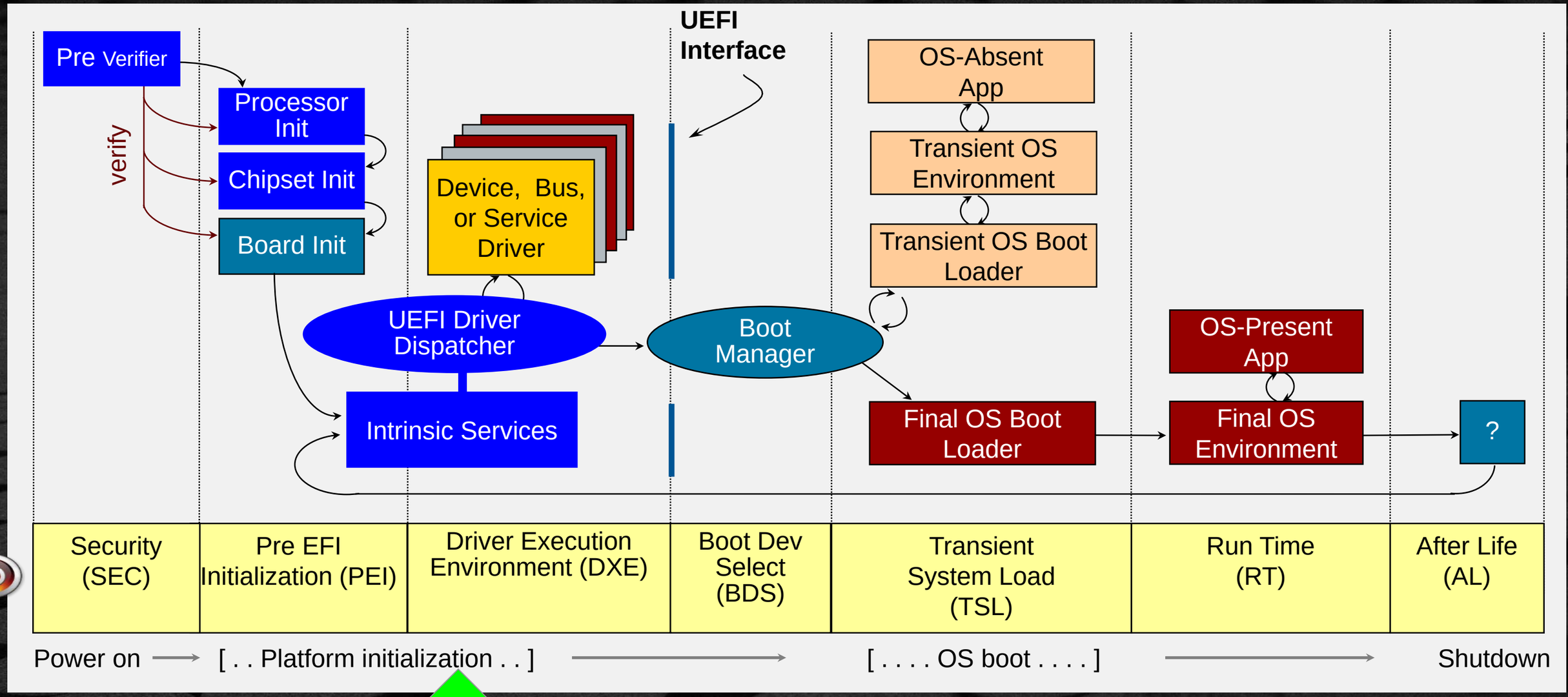


DXE IPL CHARACTERISTICS

DXE IPL

- No hard coded addresses allowed
- Find Largest Physical Memory HOB
  - Ideally this should be near Top Of Memory (TOM)
  - Allocate DXE Stack from Top of Memory
- Build HOB that describes DXE Stack
- Search FVs from HOB List for DXE Core
- Load DXE Core into Memory (PE/COFF)
- Build HOB that describes DXE Core
- Switch Stacks and Handoff to DXE Core

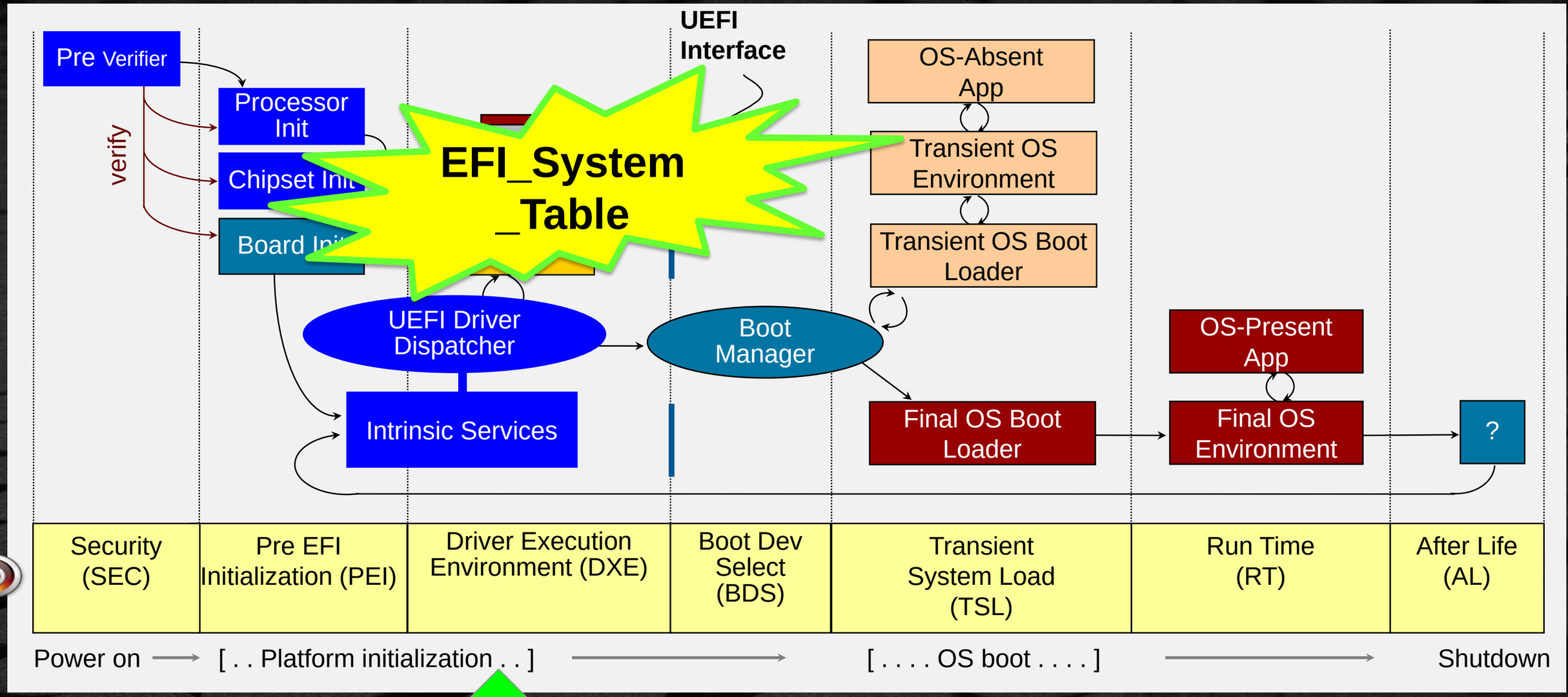
UEFI – PI & EDK II BOOT FLOW – DXE



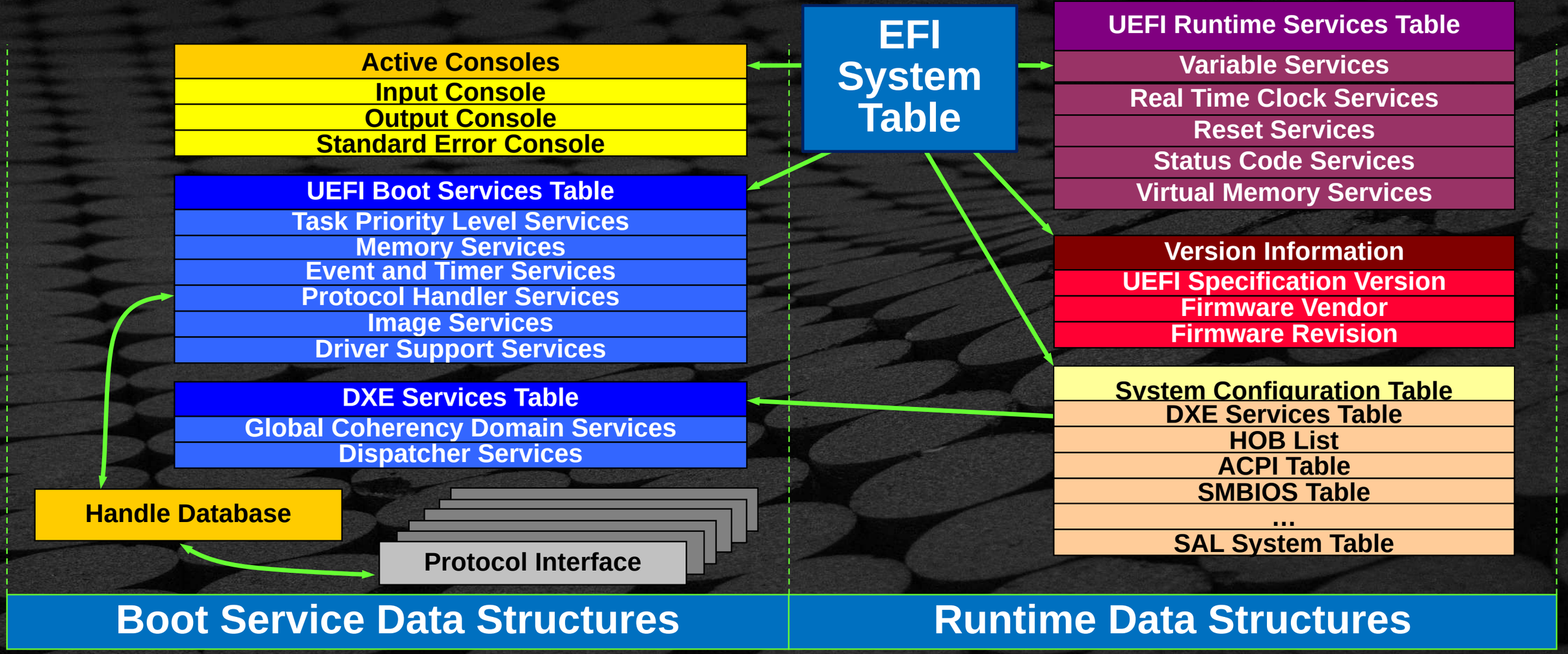
DXE CHARACTERISTICS & RESPONSIBILITIES

- Consumes HOB List from PEI
- Builds UEFI and DXE Service Tables
- EFI System Table
- UEFI Boot Services Table & UEFI Runtime Services Table
- Hands off control to the DXE Dispatcher
- and more . . .

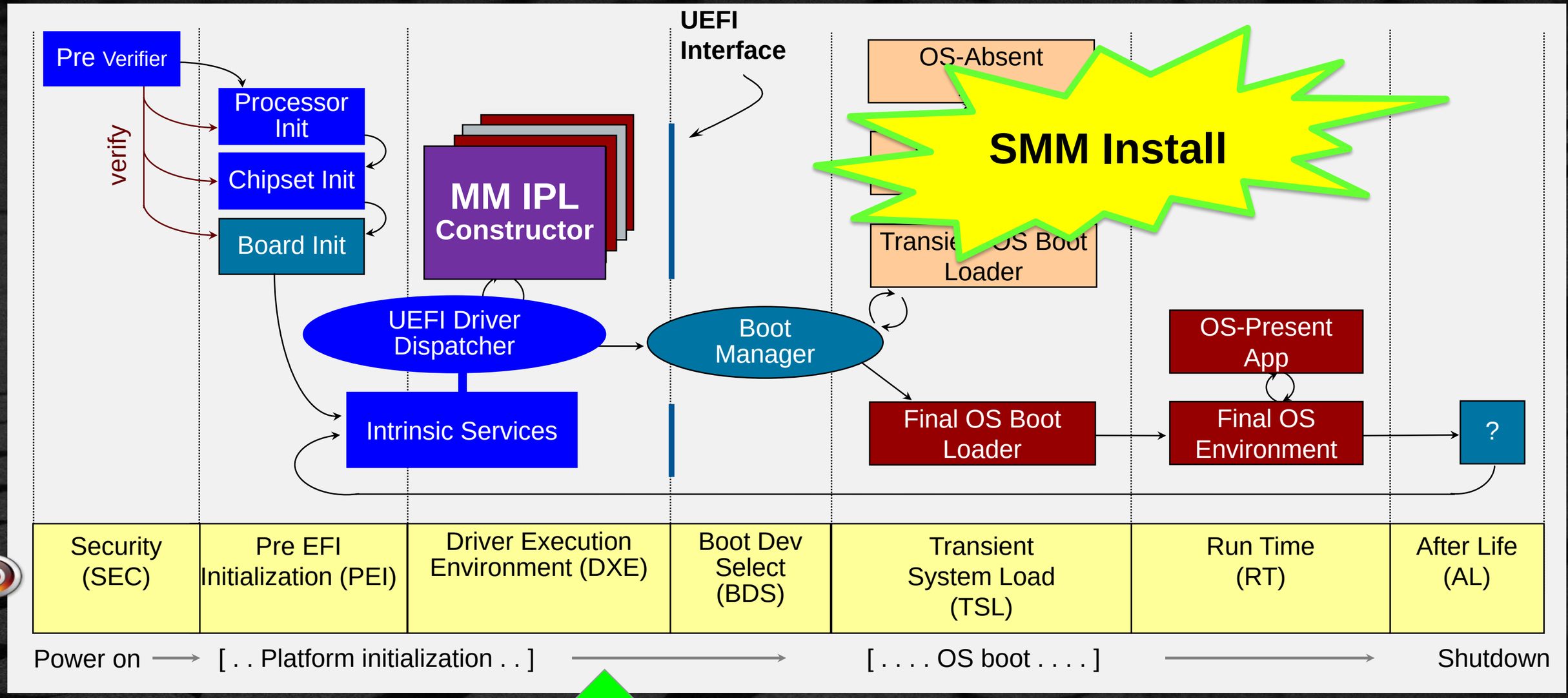
UEFI – PI & EDK II BOOT FLOW – DXE



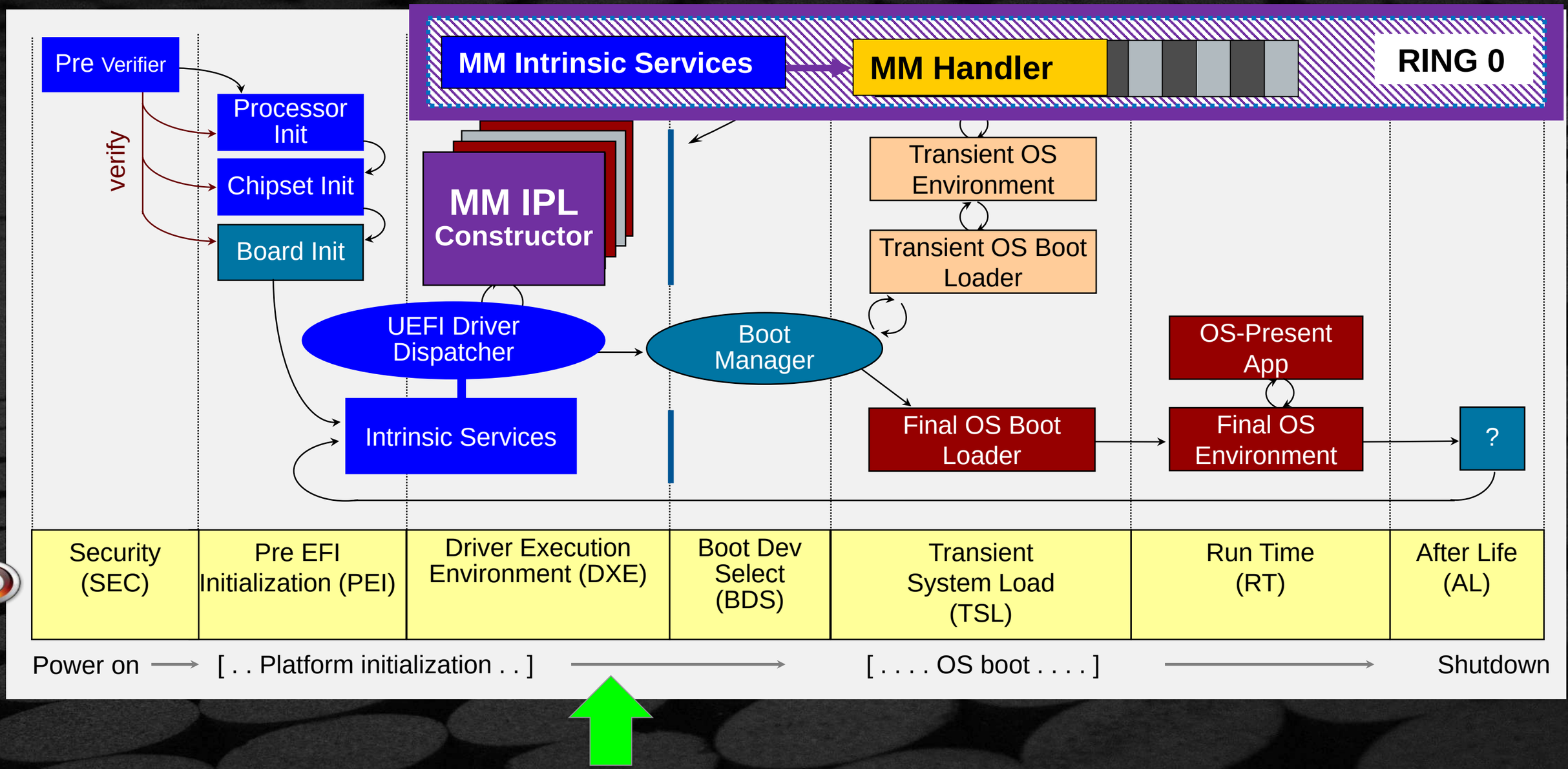
UEFI SYSTEM TABLE



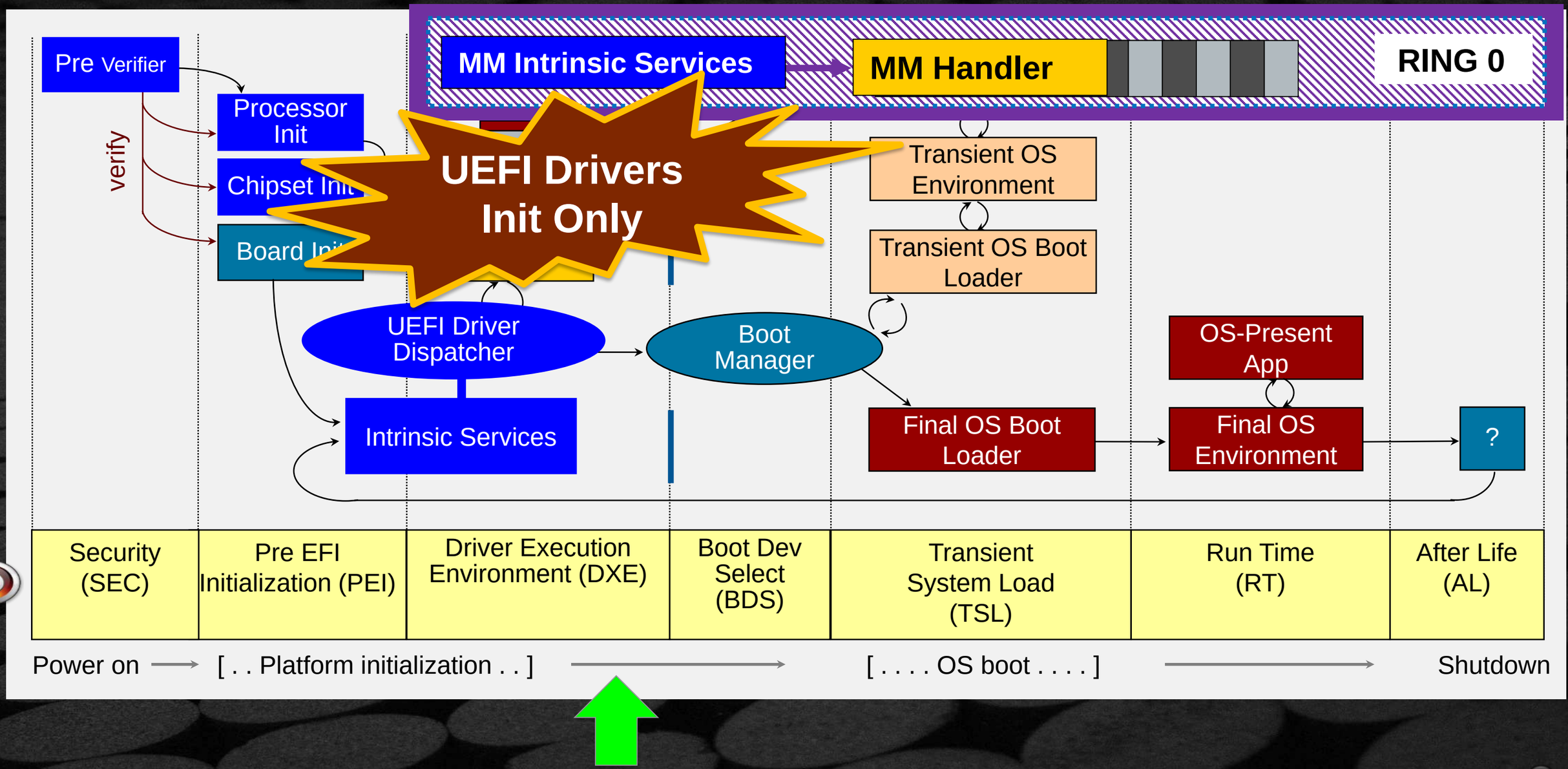
UEFI - PI & EDK II BOOT FLOW - SMM



UEFI - PI & EDK II BOOT FLOW - SMM



UEFI – PI & EDK II BOOT FLOW – DXE UEFI



UEFI Terminology

Protocols

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

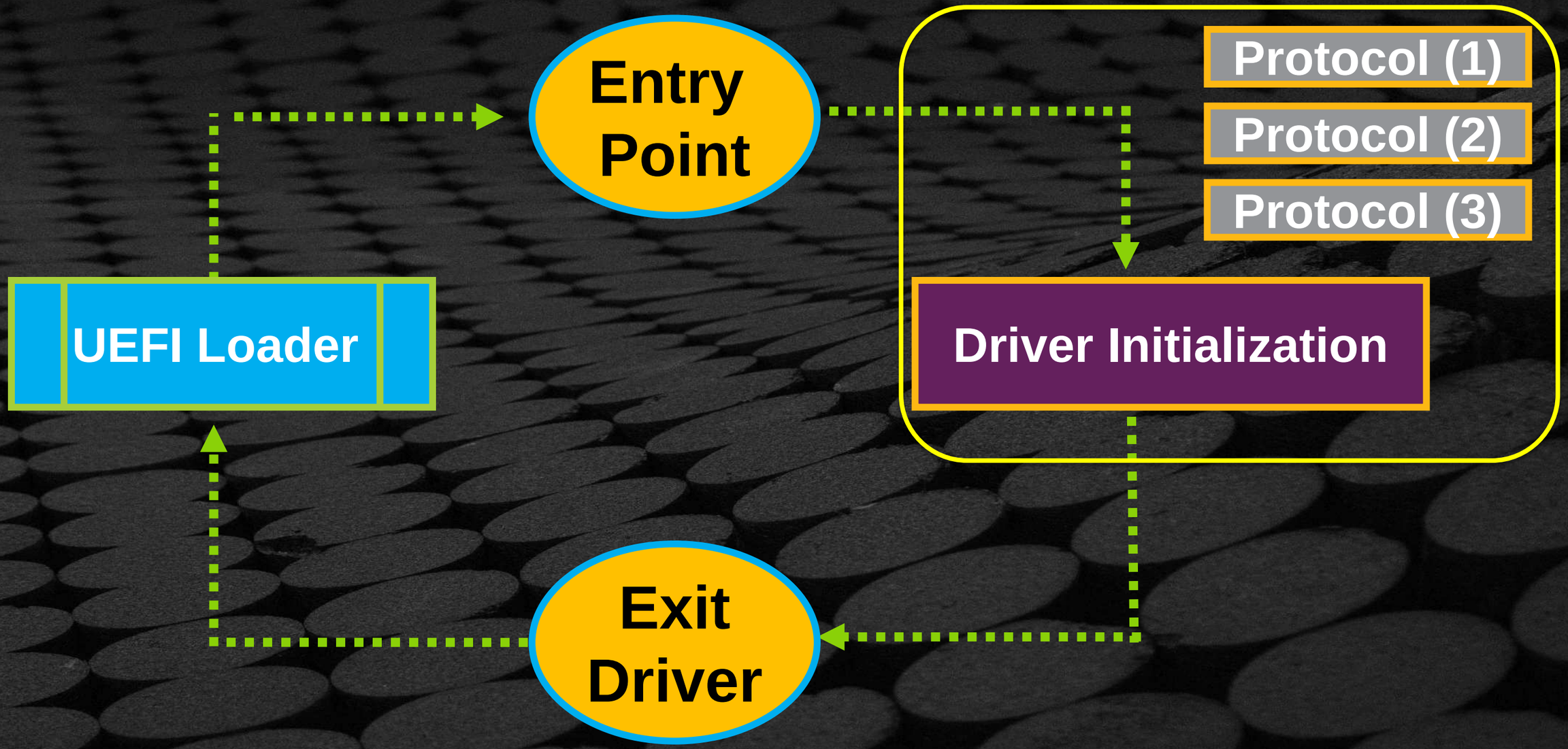
Handle Database

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

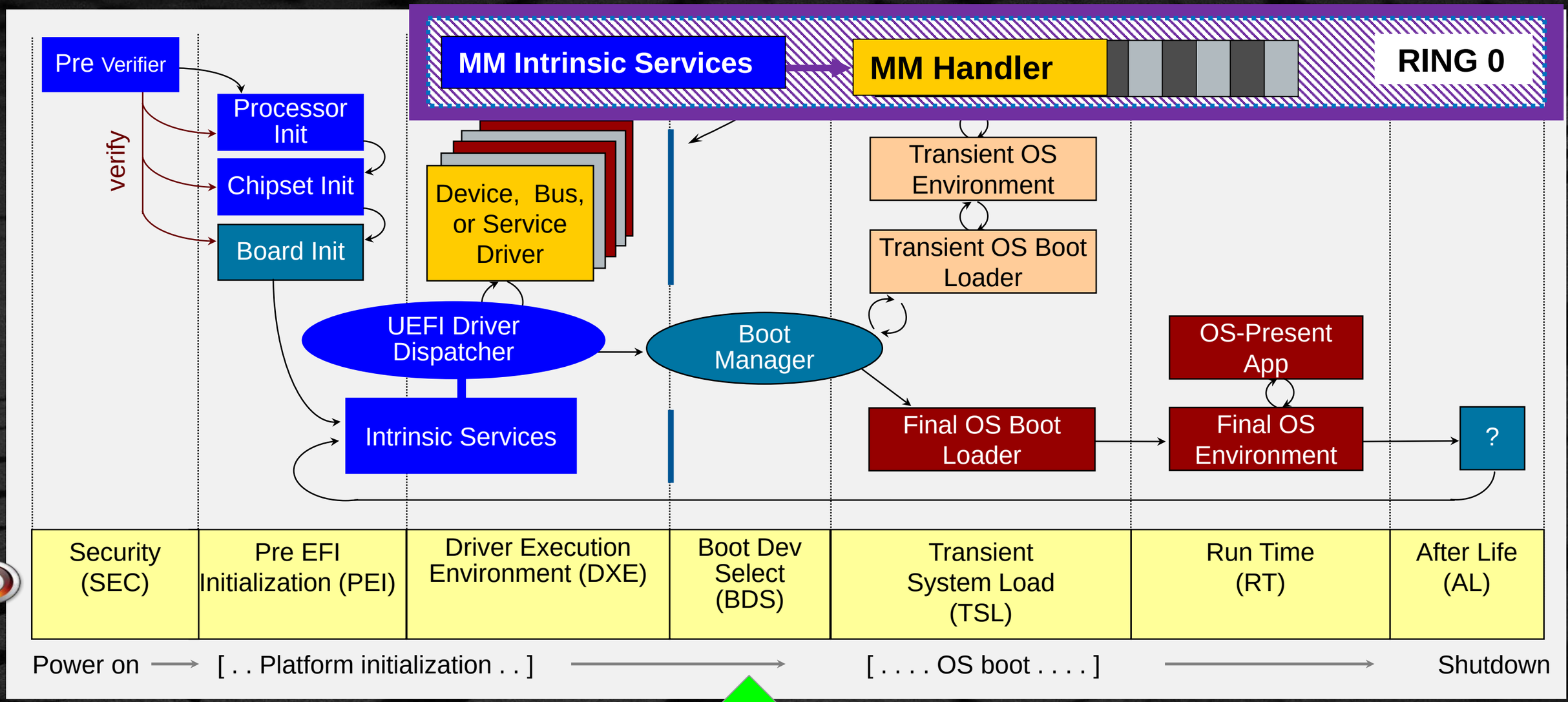
GUIDs

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

DXE Dispatcher Installs Drivers



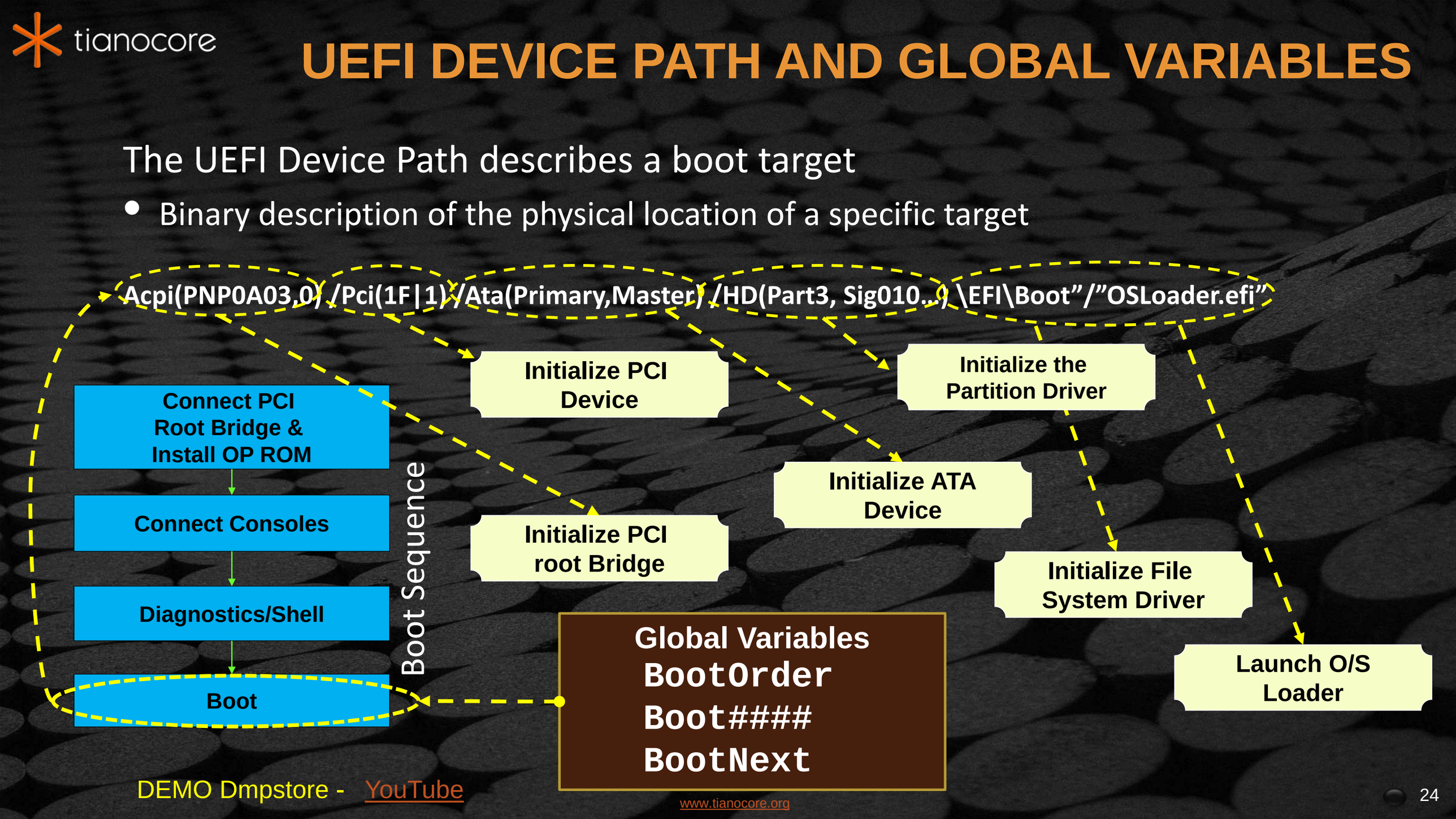
UEFI – PI & EDK II BOOT FLOW – BDS



UEFI DEVICE PATH AND GLOBAL VARIABLES

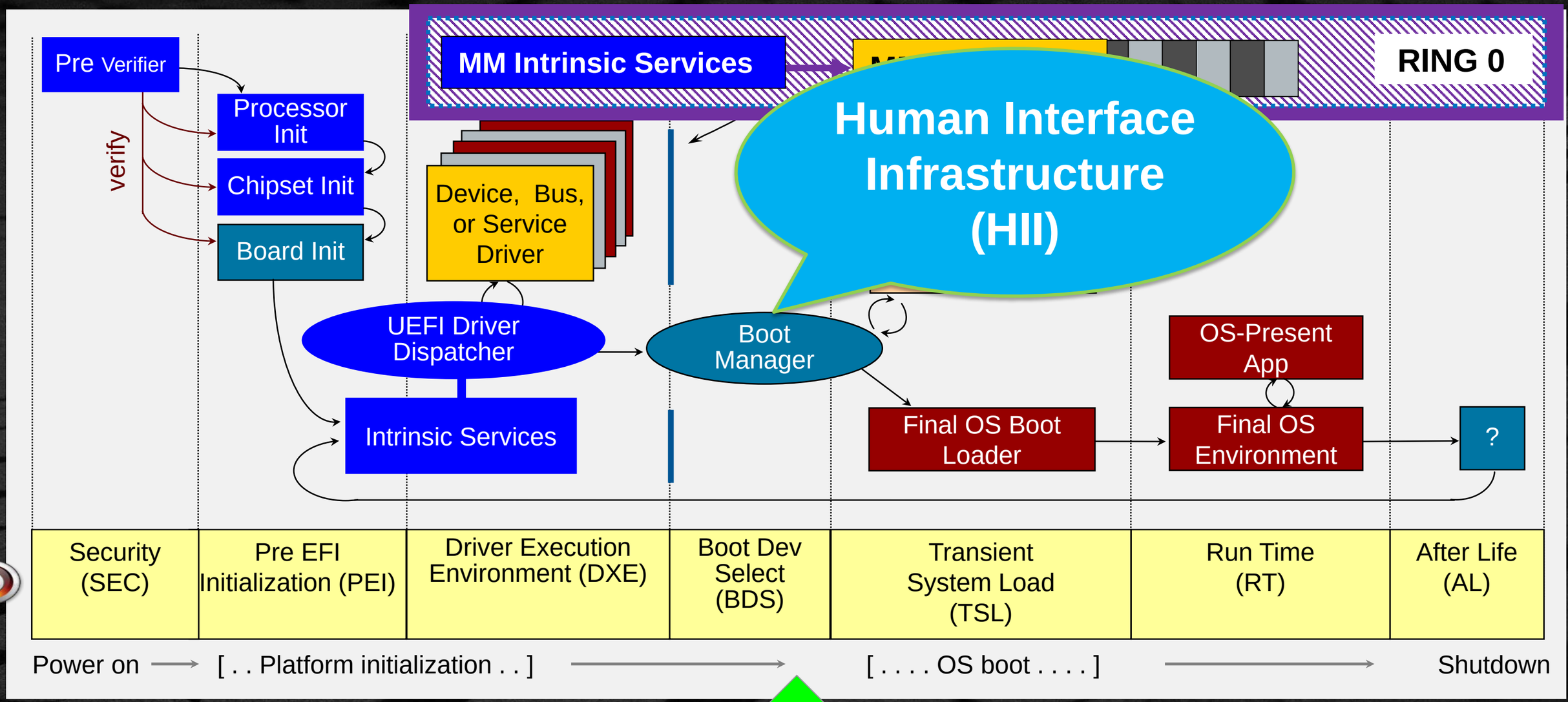
The UEFI Device Path describes a boot target

- Binary description of the physical location of a specific target

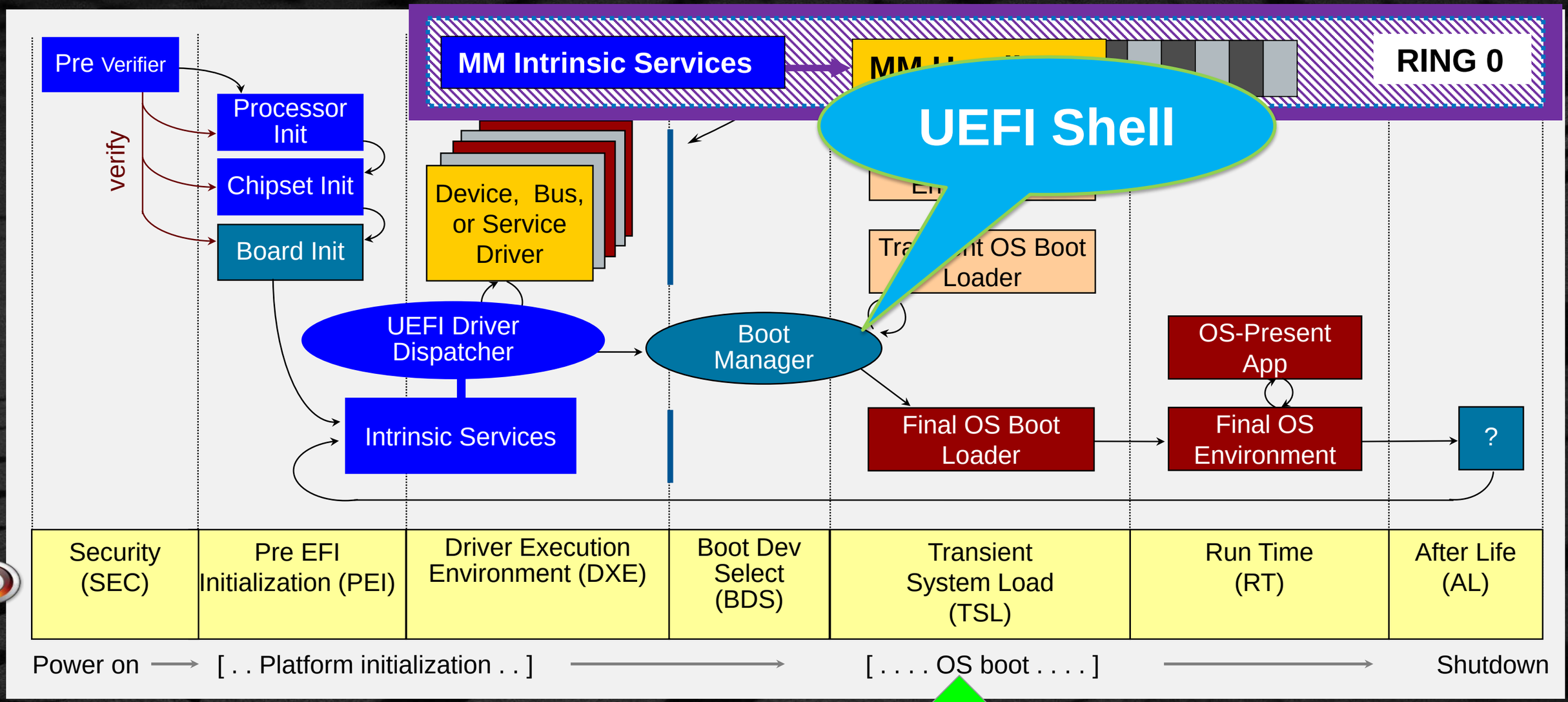


DEMO Dmpstore - YouTube

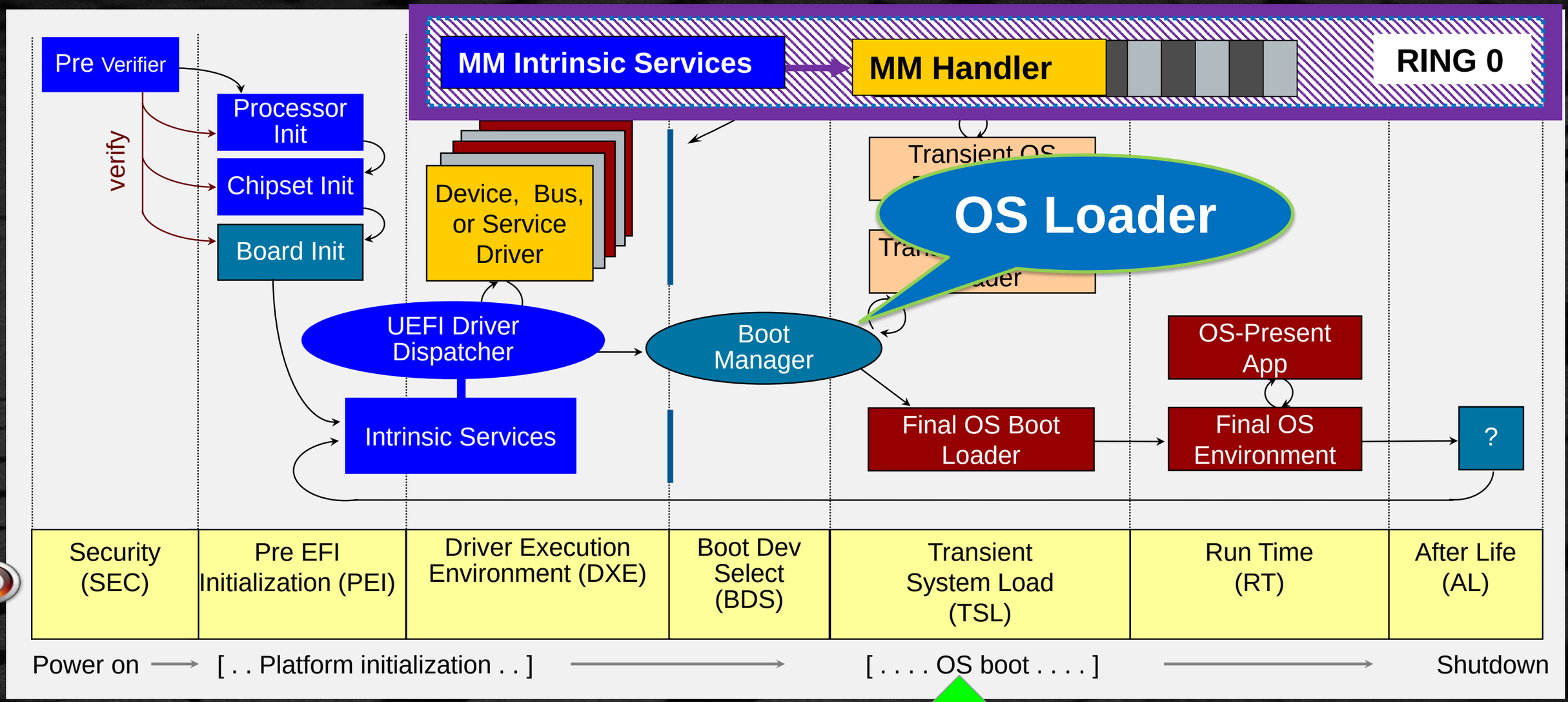
UEFI – PI & EDK II BOOT FLOW – HII



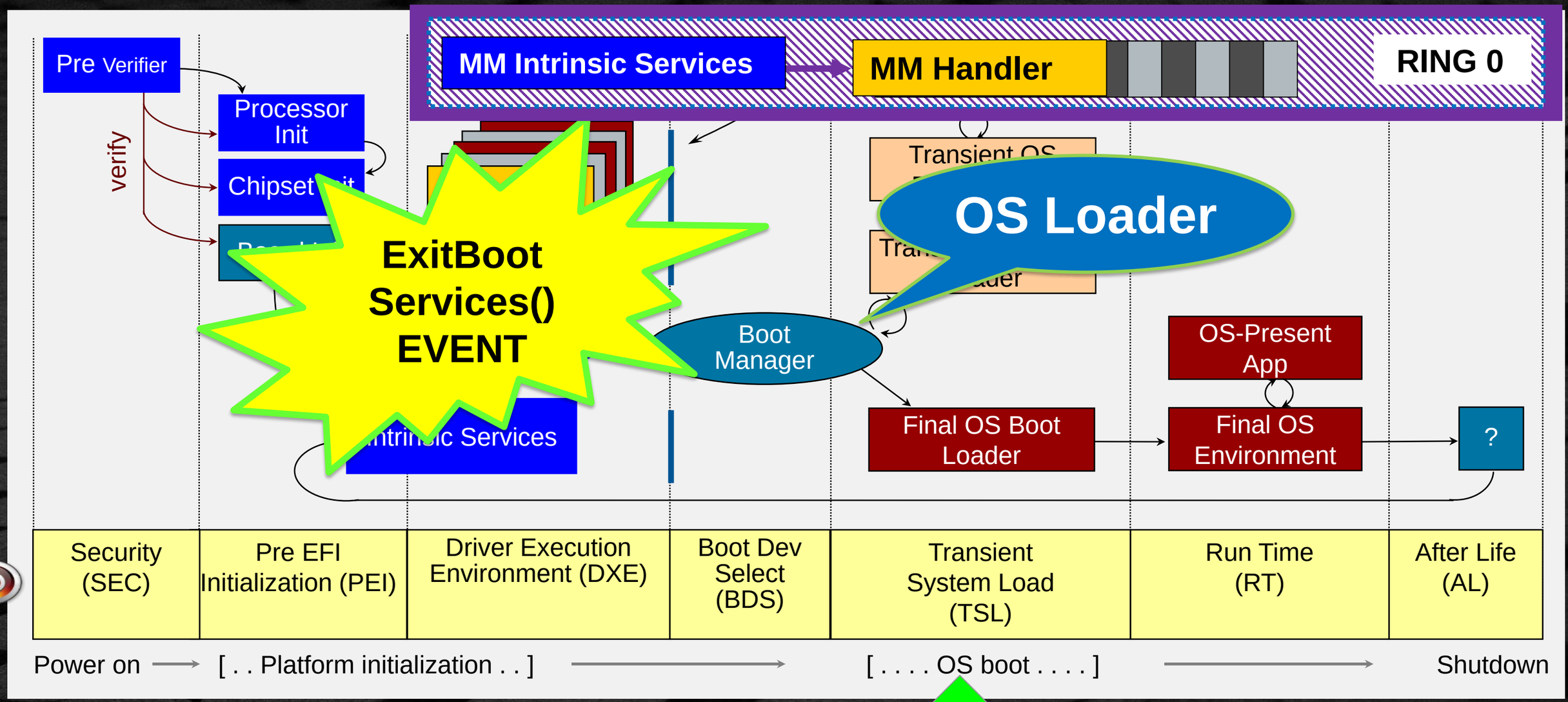
UEFI – PI & EDK II BOOT FLOW – TSL



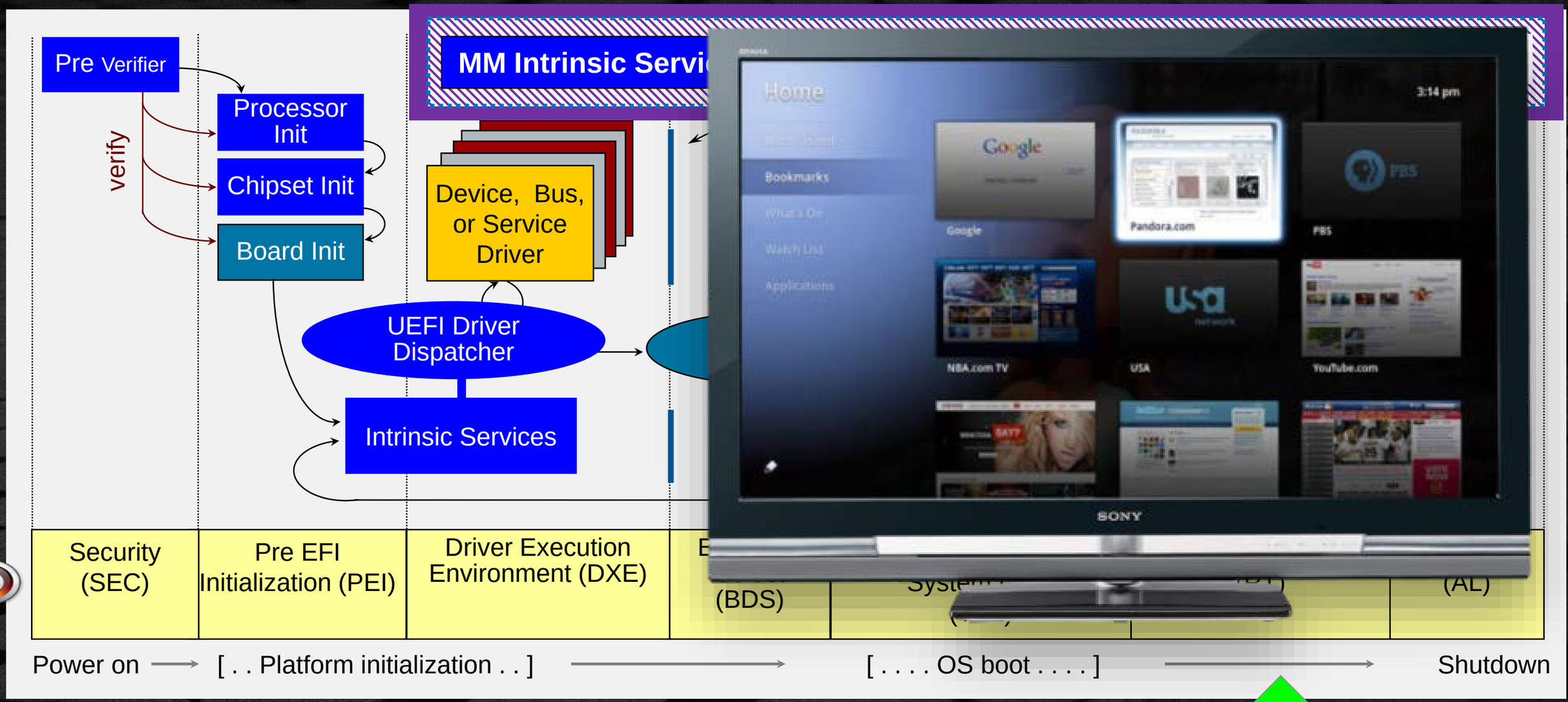
UEFI – PI & EDK II BOOT FLOW – BOOT LOADER



UEFI - PI & EDK II BOOT FLOW - EVENT



UEFI - PI & EDK II BOOT FLOW - BOOT UEFI OS



WHAT IS MANAGEMENT MODE (MM)

The UEFI PI Introduces the MM or System Management Mode (SMM)

Platform Initialization (PI) Specification Introduces Management Mode (MM)**

UEFI PI-standard for creating a protected execution environment using hardware resources

- Dedicated, protected memory space, entry point and hardware resources, such as timers and interrupt controllers
- Implemented using SMM (Intel® Architecture) or TrustZone(Arm)
- Highest-privilege operating mode (Ring 0) with greatest access to system memory and hardware resources

Presented at UEFI Plugfest Fall 2017: Presentation link

**Formerly known as SMM in PI specification

Why are Software MMI Vulnerabilities Dangerous?

Because . . .

Software MMIs can be asked to perform:

- Privileged operations: Flash System FW (IFWI), flash EC, write to MMIO, write to MMRAM, etc.
- Overwrite OS code/data
- Copy protected OS data to another unprotected location
- Copy protected firmware data to another unprotected location
- Overwrite System FW code/data

UEFI Platform Firmware Assumptions

- Memory protected by the OS cannot be snooped while in use by the OS application or OS driver
  - No protection from MM, VMs or hardware snooping
- Flash protected by hardware cannot be modified outside of MM after the end of DXE
  - Not worried about snooping since no secrets are stored in System FW
  - Not worried about flash-altering hardware attacks
- Software MMIs cause CPUs to enter SMM in SMRAM at a fixed location
- MMRAM cannot be altered from outside SMM

Key Points for More Secure Software MMI Handlers

- Allocate The Buffer In PEI/DXE
- Never Trust That Pointers Point To The Buffer
- Prohibit Input/Output Buffer Overlap
- Don’t Trust Structure Sizes
- Verify Variable-Length Data

THE INTEL® FIRMWARE SUPPORT PACKAGE (INTEL® FSP)

INTEL® FSP - COMPONENTS

- CPU, memory controller, and chipset initialization functions as a binary package
- Provides silicon initialization ingredients
- Plugs into existing firmware frameworks
- Integration guide, includes API documentation

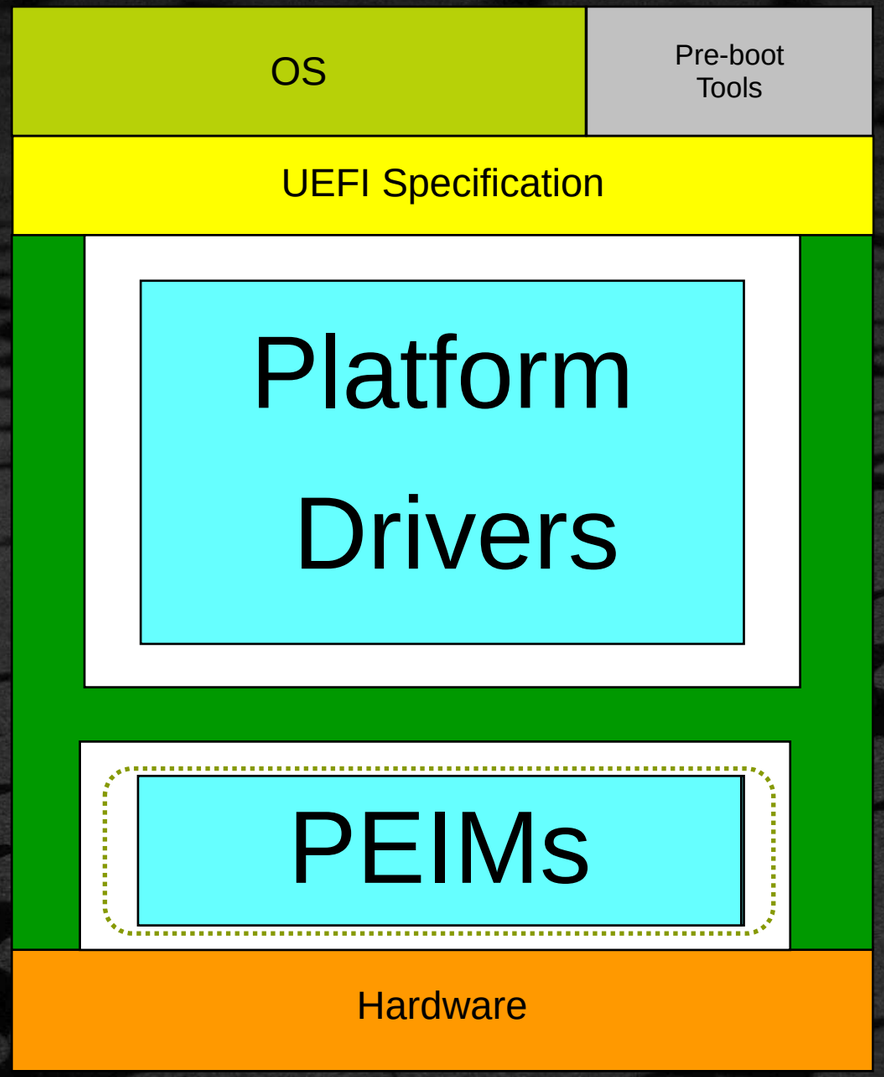
Intel FSP is currently available for the many Intel hardware-producing divisions

See: About Intel FSP (Intel® FSP 2.2 May 2020)

White Paper Example: Open Braswell - Design and Porting Guide

Intel® FSP is NOT a stand-alone boot-loader

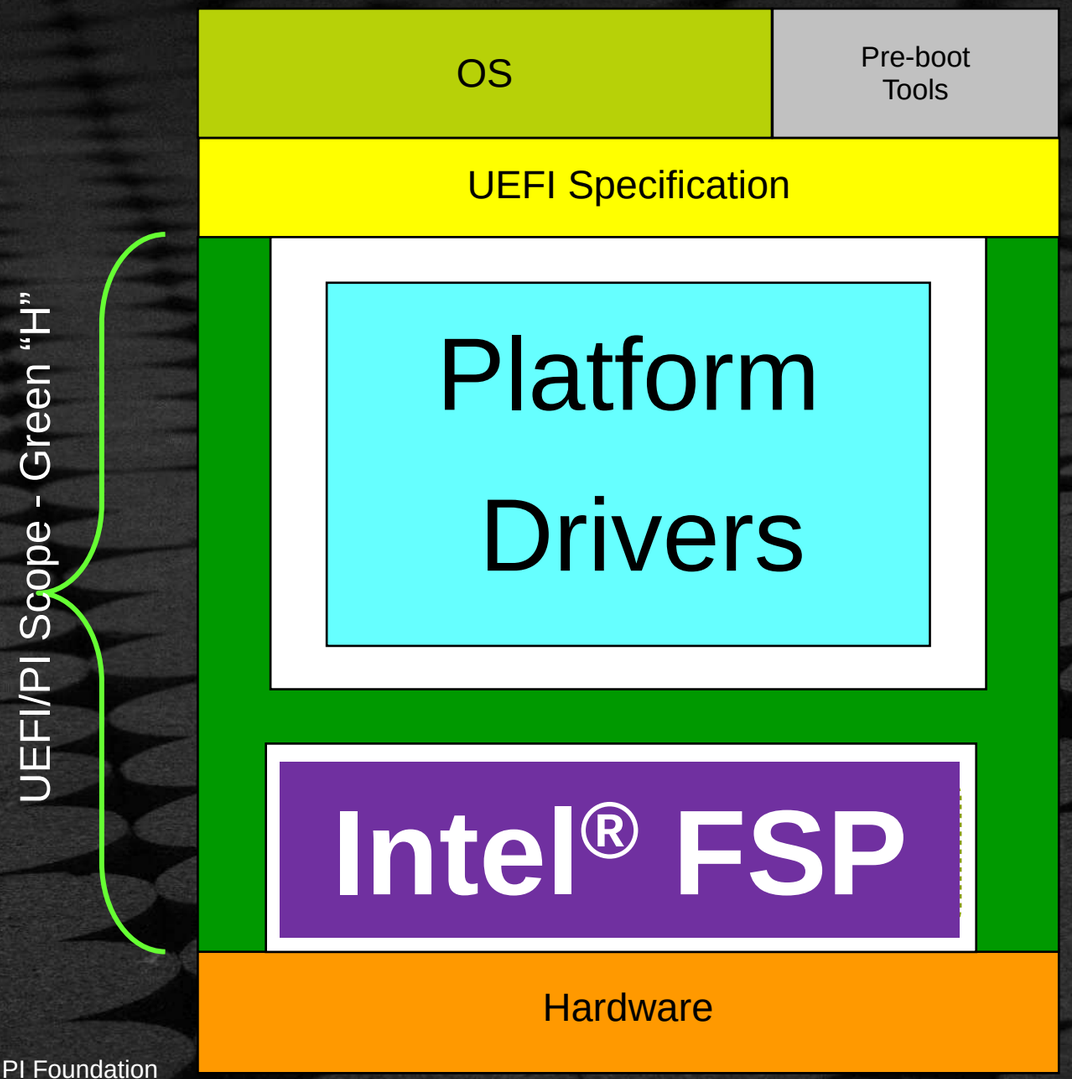
INTEL® FSP TO OPEN SOURCE EDK II



EDK II provides the framework ("Green H")

Intel® Firmware Support Package (Intel® FSP) provides low level of silicon initialization

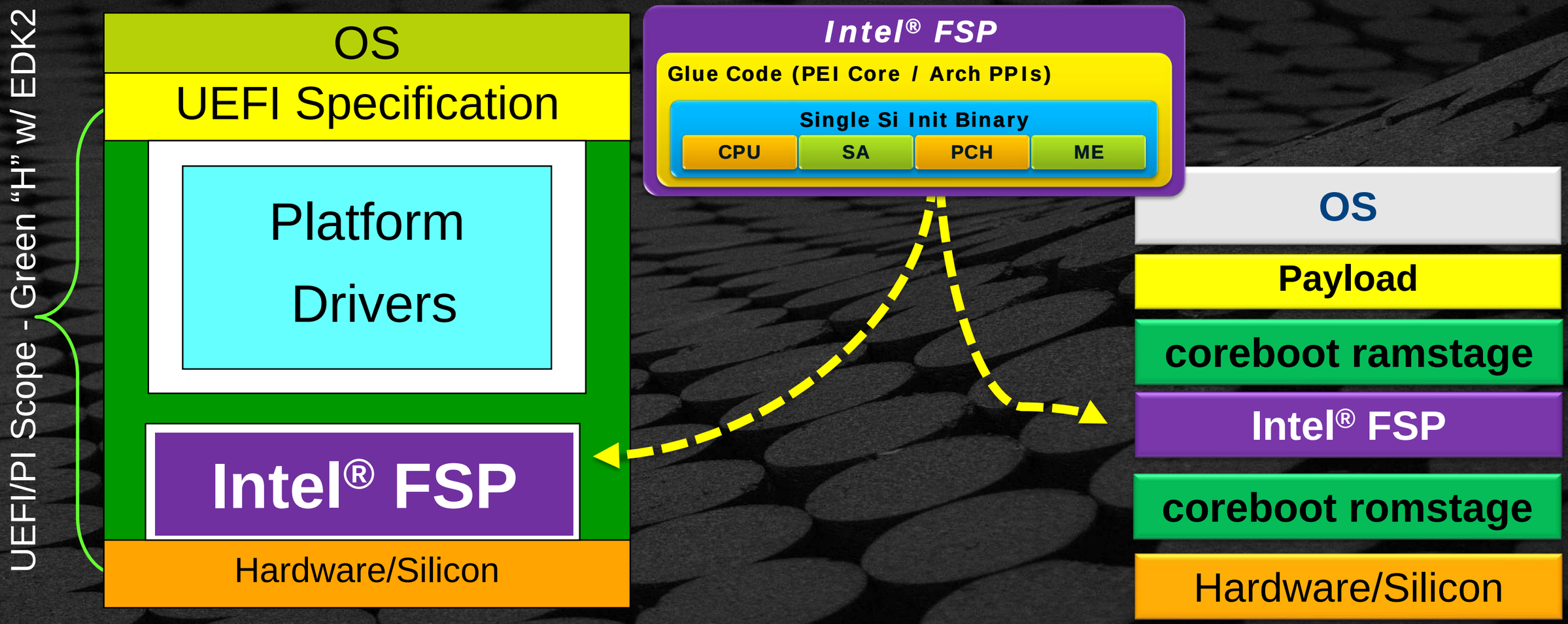
INTEL® FSP TO OPEN SOURCE EDK II



EDK II provides the framework ("Green H")

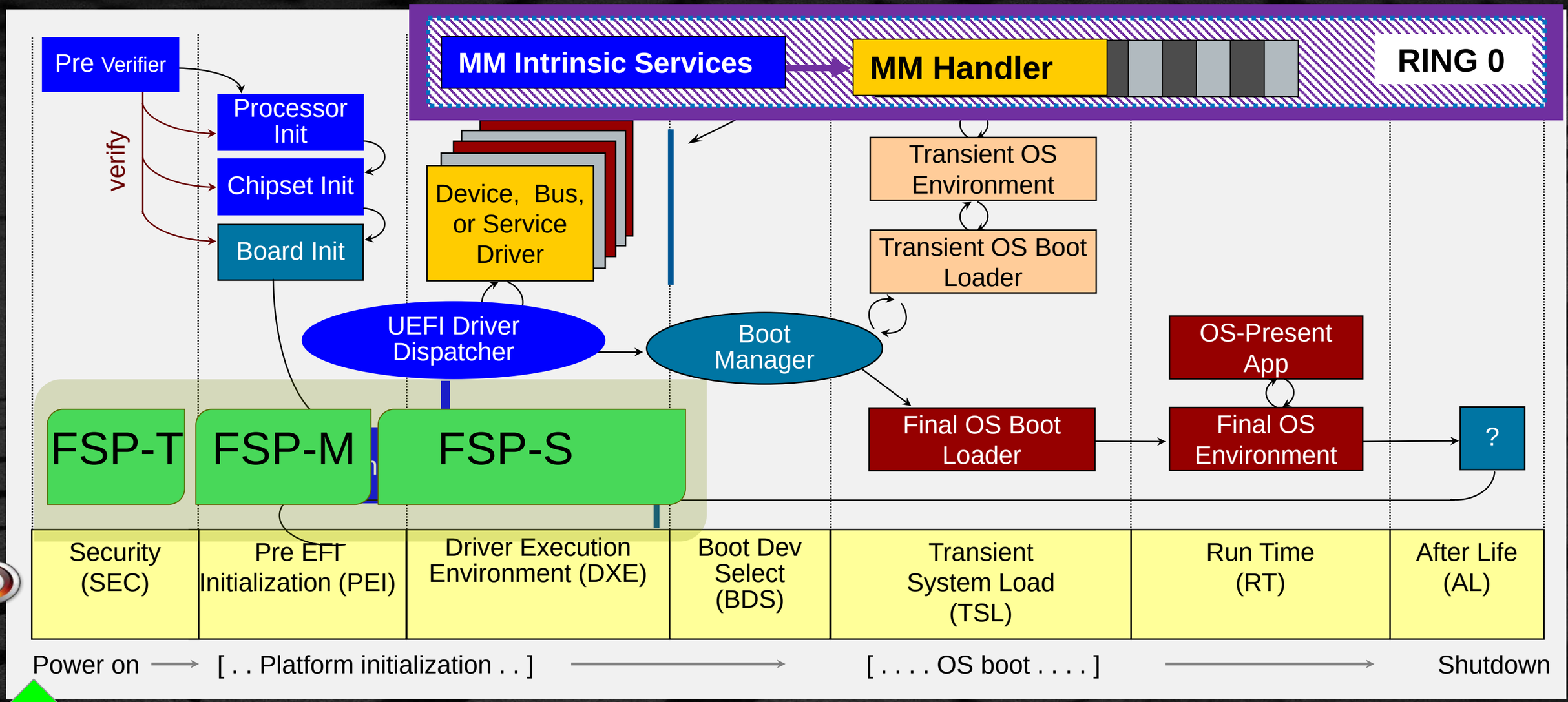
Intel® Firmware Support Package (Intel® FSP) provides low level of silicon initialization

Intel® FSP "Produced" to "Consuming" Intel® Architecture Firmware

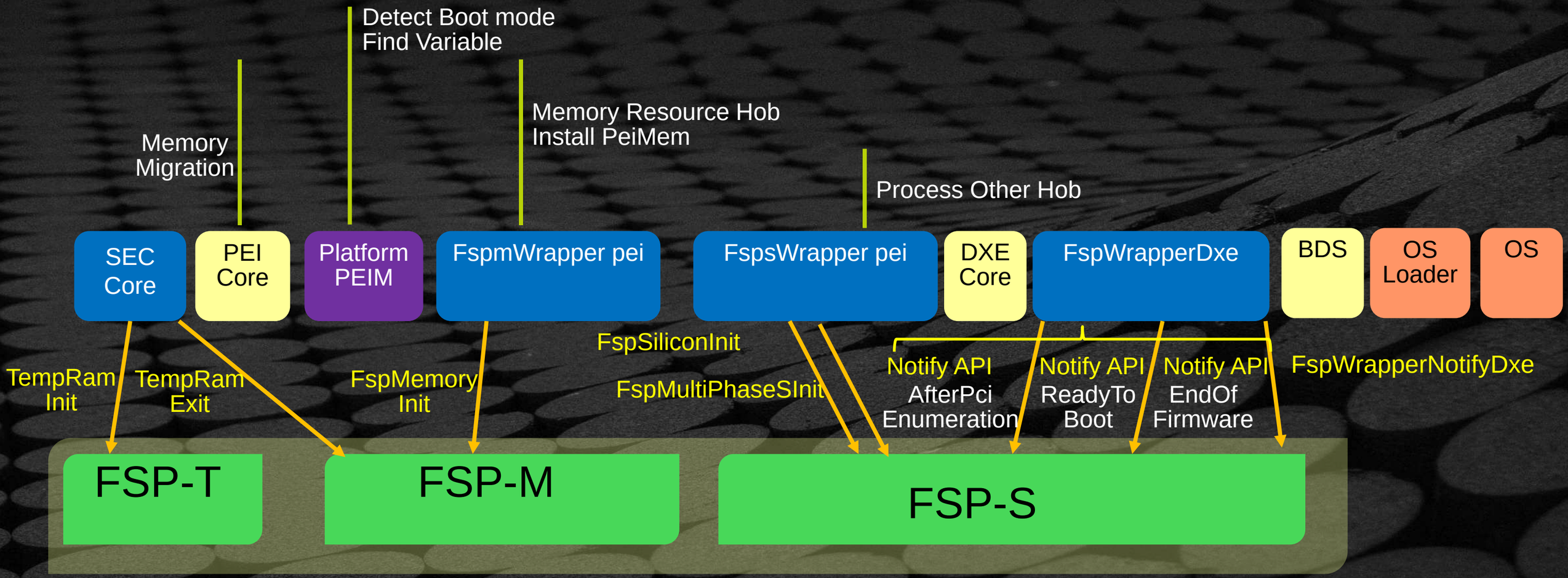


Intel FSP is independent of the bootloader solutions

UEFI – PI & EDK II BOOT FLOW – FSP



Boot Flow with UEFI & Intel® FSP



Original Source: Using the Intel® FSP with EDK II (2.0) Fig 4. – This now shows a 6th API added in FSP 2.2

INTEL® FSP PRODUCER

- Examples of binary instances on http://www.intel.com/fsp with integration guides
  - This includes hardware initialization code that is EDK II based PEI Modules (PEIM's)
- Modules are encapsulated as a UEFI PI firmware volume w/ extra header
- Configure w/Vital Product Data (VPD)-style Platform Configuration Data (PCD) externalized from the modules
- Resultant output state reported via UEFI Platform Initialization (PI) Hand Off Block (HOB)

Intel® Firmware Support Package (Intel® FSP) External Architecture Specification (EAS) v2.2 Link v2.0

Resource: https://software.intel.com/content/www/us/en/develop/articles/intel-firmware-support-package.html

SOURCE FOR INTEL® FSP PRODUCER CODE

- CPU and chipset-specific code for PEIM's inside of the Intel FSP can be open or closed, code at Intel FSP-repo
- PEI core and infrastructure code at tianocore.org/edk2
  - /MdePkg
  - /MdeModulePkg
- Code to interface Intel FSP to EDK II can be found at :
  - /IntelFsp2Pkg and Wrapper at: /IntelFsp2WrapperPkg

Intel FSP can encapsulate IP protected initialization code PRODUCED by Intel business units

WHAT'S NEW IN THE UEFI SPECIFICATIONS?

LATEST UEFI SPECIFICATIONS

Http://uefi.org

Unified Extensible Firmware Interface Forum

UEFI Specification	UEFI Shell Specification	UEFI PI Specification	Self Certification Test	PI Distro Package Specification	ACPI Specification
Current v2.8B June 2020	Current v2.2 January 2016	Current v1.7A April 2020	Current v2.7B April 2015	Current v1.1 January 2016	Current v6.3A October 2020

http://www.uefi.org/specsandtesttools

EDK II - Open Source

Community Development

- Stable Tag Releases- cycle of releasing stable versions of EDK II Firmware
- Adding UEFI Spec updates and new key features and bug fixes
- Three phases of development
  - Development phase
  - Soft Feature Freeze
  - Hard Feature Freeze

More Information on Stable Tag Releases:
TianoCore Wiki

Tag: edk2-stable202102 Features:
edk2 releases Stable tag

Report a bug on Bugzilla

Create a user account https://bugzilla.tianocore.org/

Search if bug “already” reported

File New Report – Pick a product – fill out form for the bug

tianocore

SUMMARY

- The System Firmware is a binary image that starts execution as the reset vector & is typically a SPI device
- UEFI & PI Boot Flow Process, SEC, PEI, DXE, BDS, TSL, OS
- System Management Mode is in Ring 0 in the System FW
- Intel® FSP will initialize the processor, chipset and memory
- The UEFI.org & Tianocore.org for Specs and Open source

tianocore

Questions?

Return to Main Training Page

Return to Training Table of contents for next presentation link

tianocore

tianocore

BACKUP

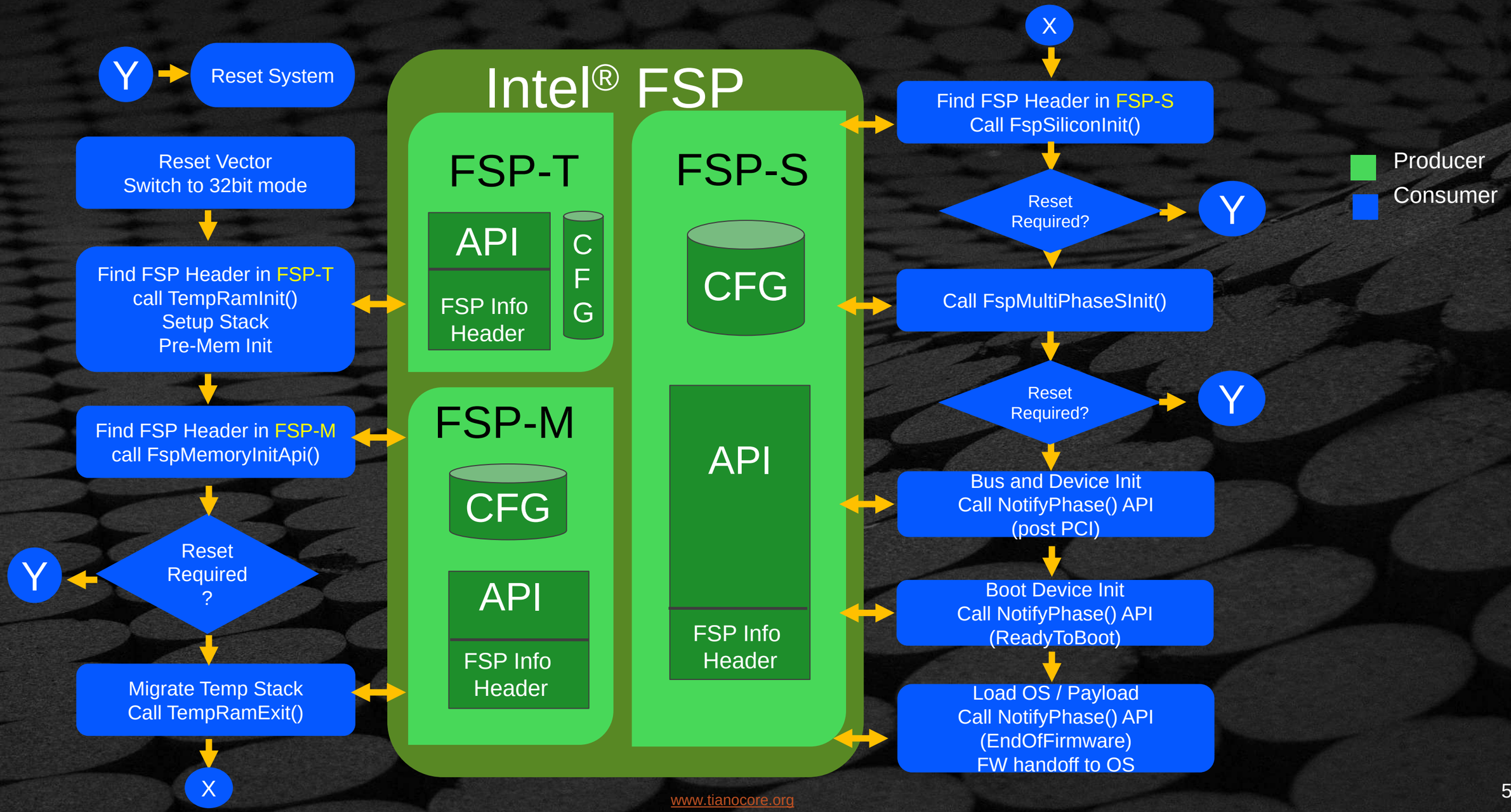
What is Intel® Firmware Support Package?

- Intel® Firmware Support Package (Intel® FSP) includes:
  - A binary file
  - An integration guide
  - A rebasing tool
  - An IDE configuration tool / Boot Setting File (BSF)
- Provide silicon initialization code:
  - Initializes processor core, chipset as explained in BIOS Writers' Guide
  - Is relocatable in ROM
  - Can be configured for platform customization
- Boot loader agnostic and can be easily integrated with many options:
  - Open source boot loaders: UEFI –EDK II, Coreboot, U-boot, etc.
  - RTOS
  - Others

tianocore

Using Intel® FSP w/ EDK II: PDF

Intel® FSP V2.2 Boot Flow



Intel® FSP Integration

Placement:

Once the Intel FSP binary is ready for integration, the bootloader build process needs to configured to place the Intel FSP binary at the proper base address.

Rebase:

The Intel FSP is not Position Independent Code (PIC) and the whole Intel FAP has to be rebased with the Binary Configuration Tool (BCT) if it is placed at a location that is different form the default base address of the Intel FSP.

Interface:

The bootloader needs to add code to setup the Operating environment for the Intel FSP, call Intel FSP with the correct parameters and parse the Intel FSP output to retrieve the necessary information returned by the Intel FSP.

Customization:

The static Intel FSP configuration parameters/features are part of the Intel FSP binary and can be customized with BCT

https://www.intel.com/content/www/us/en/intelligent-systems/intel-firmware-support-package/fsp-firmware-solutions-iot-video.html - at -41:00 secs into video

tianocore

ACKNOWLEDGEMENTS

tianocore

WAY – WAY - BACK - BACKUP

UEFI Specification & EDK II Reference Implementation Timeline

UEFI Specification -top & EDK II Open Source -bottom