

UEFI & EDK II Training

UEFI AND PLATFORM INITIALIZATION (PI) BOOT FLOW &
OVERVIEW

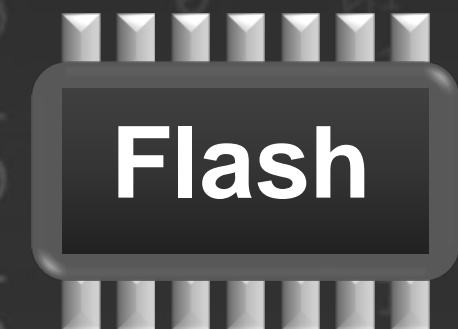
tianocore.org

LESSON OBJECTIVE

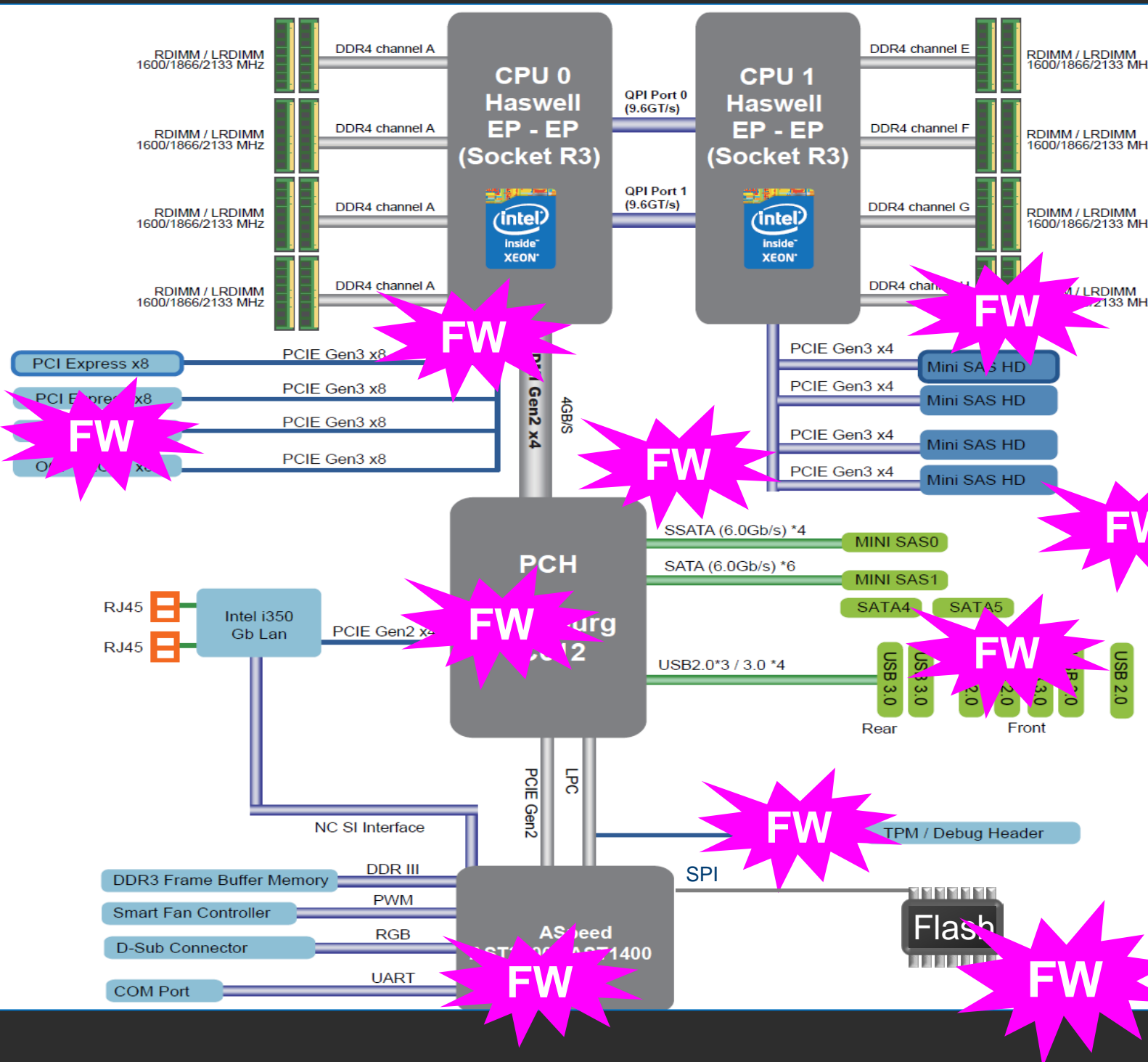
-  Where is the System Firmware
-  Review UEFI Platform Initialization Boot Flow Process
-  What is Intel[®] Firmware Support Package (Intel[®] FSP)
-  The Universal Scalable FW (USF), UEFI.org Forum & Tianocore.org

WHERE IS THE FIRMWARE

Where is the UEFI Firmware on a platform

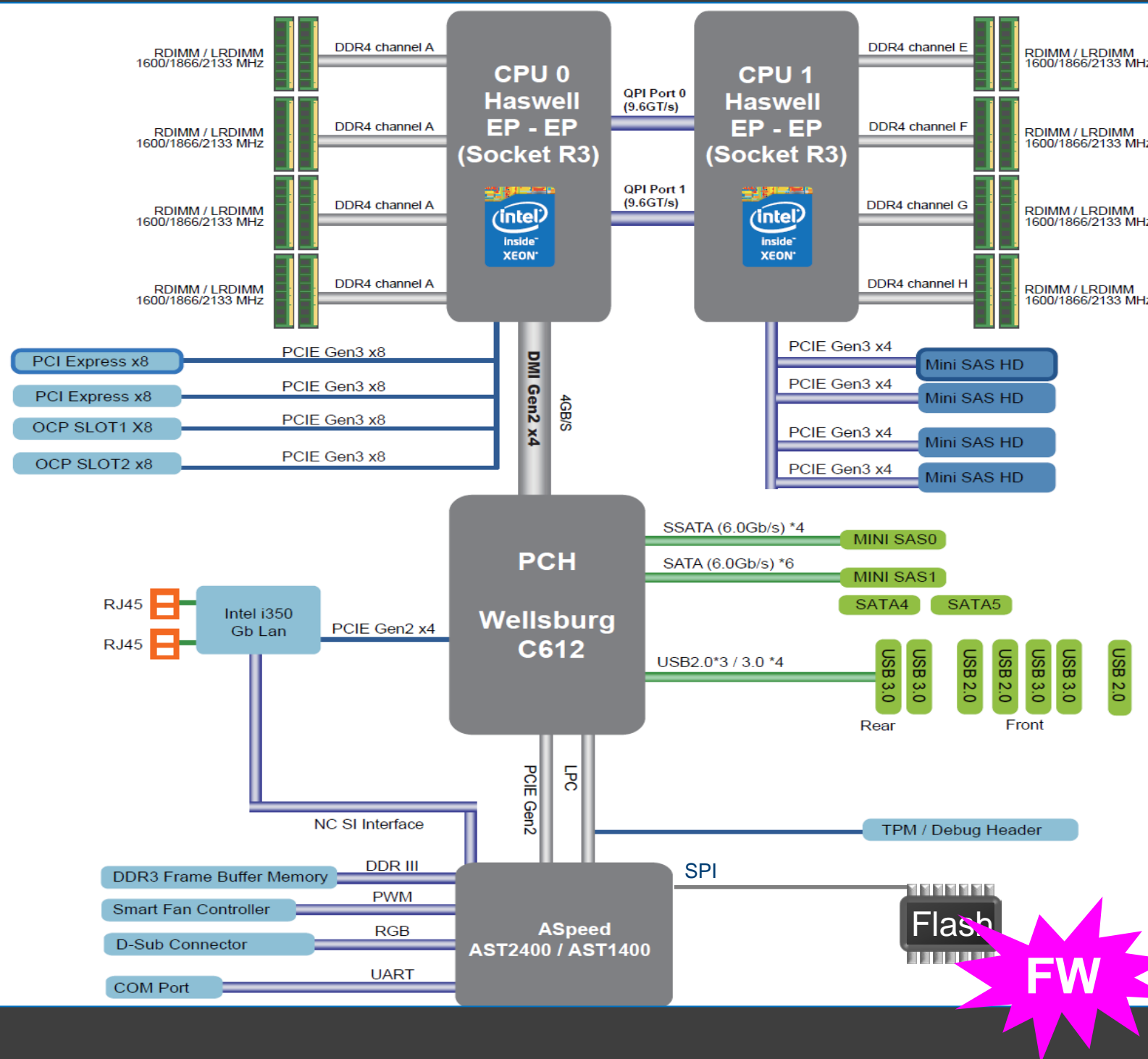


Firmware is Everywhere



- 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)

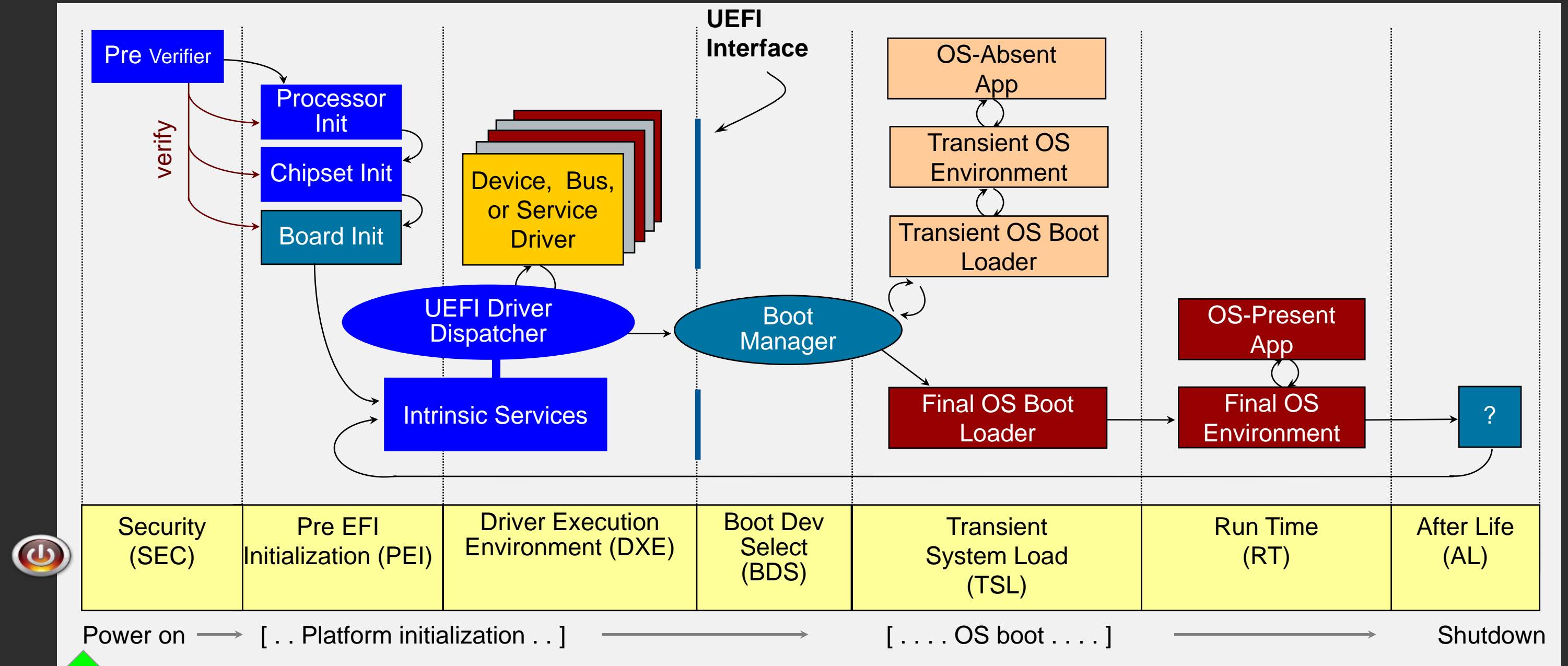



Main system firmware (BIOS, UEFI firmware, coreboot)

UEFI BOOT EXECUTION FLOW

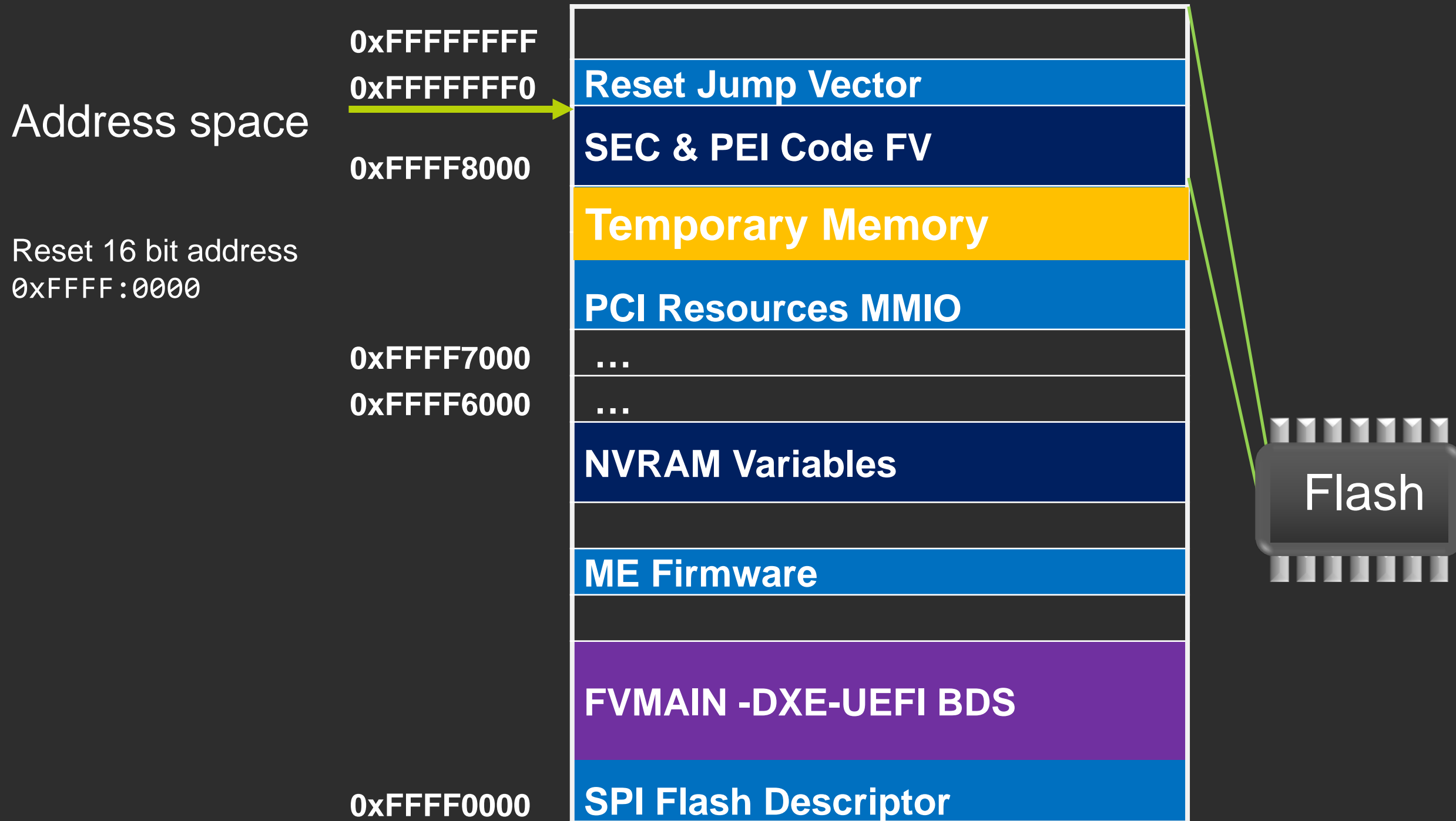
Starting at the processor reset vector

UEFI – PI & EDK II BOOT FLOW - SEC



 System Reset Vector
Stage 7 on IA

PRE-MEMORY INIT



STARTING AT THE RESET VECTOR- SEC

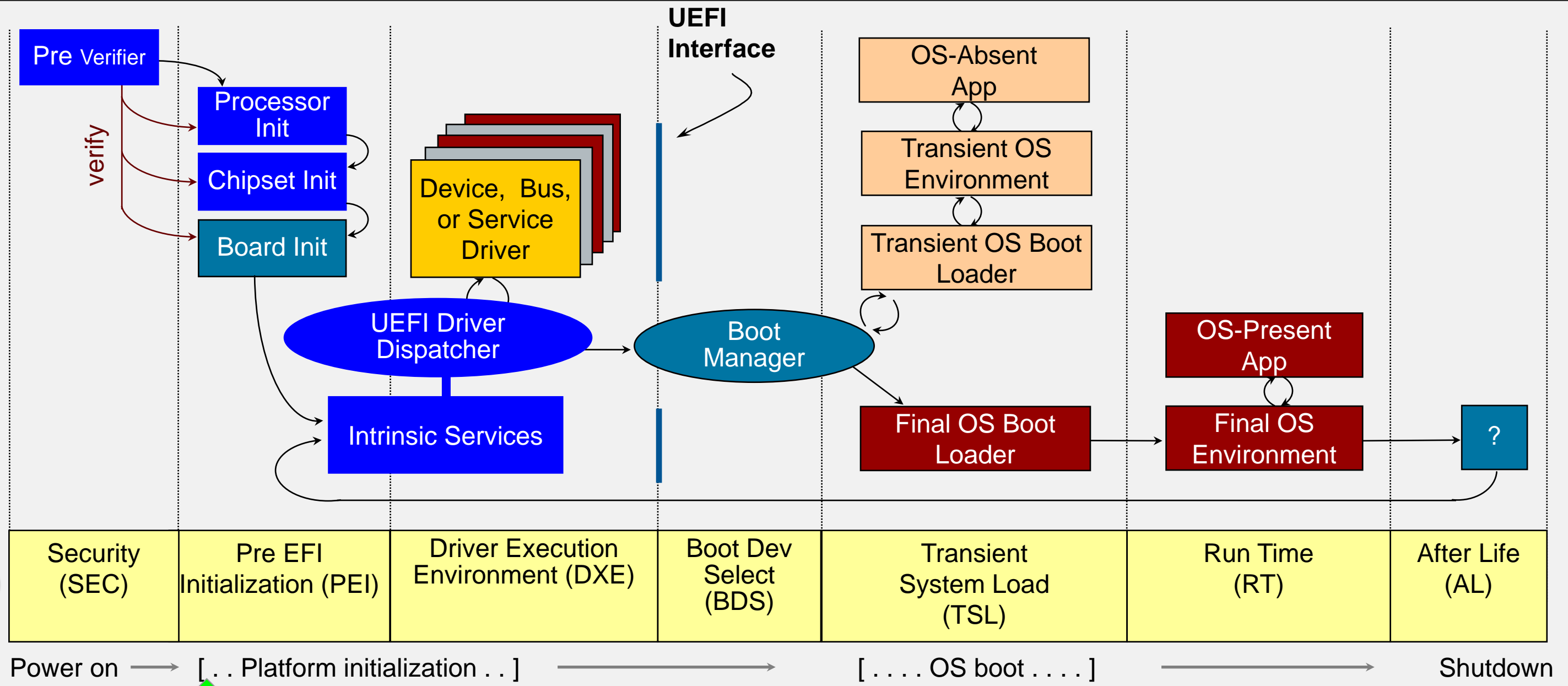
The Processor Executes SEC starting at the first fetch from the reset vector

- 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)

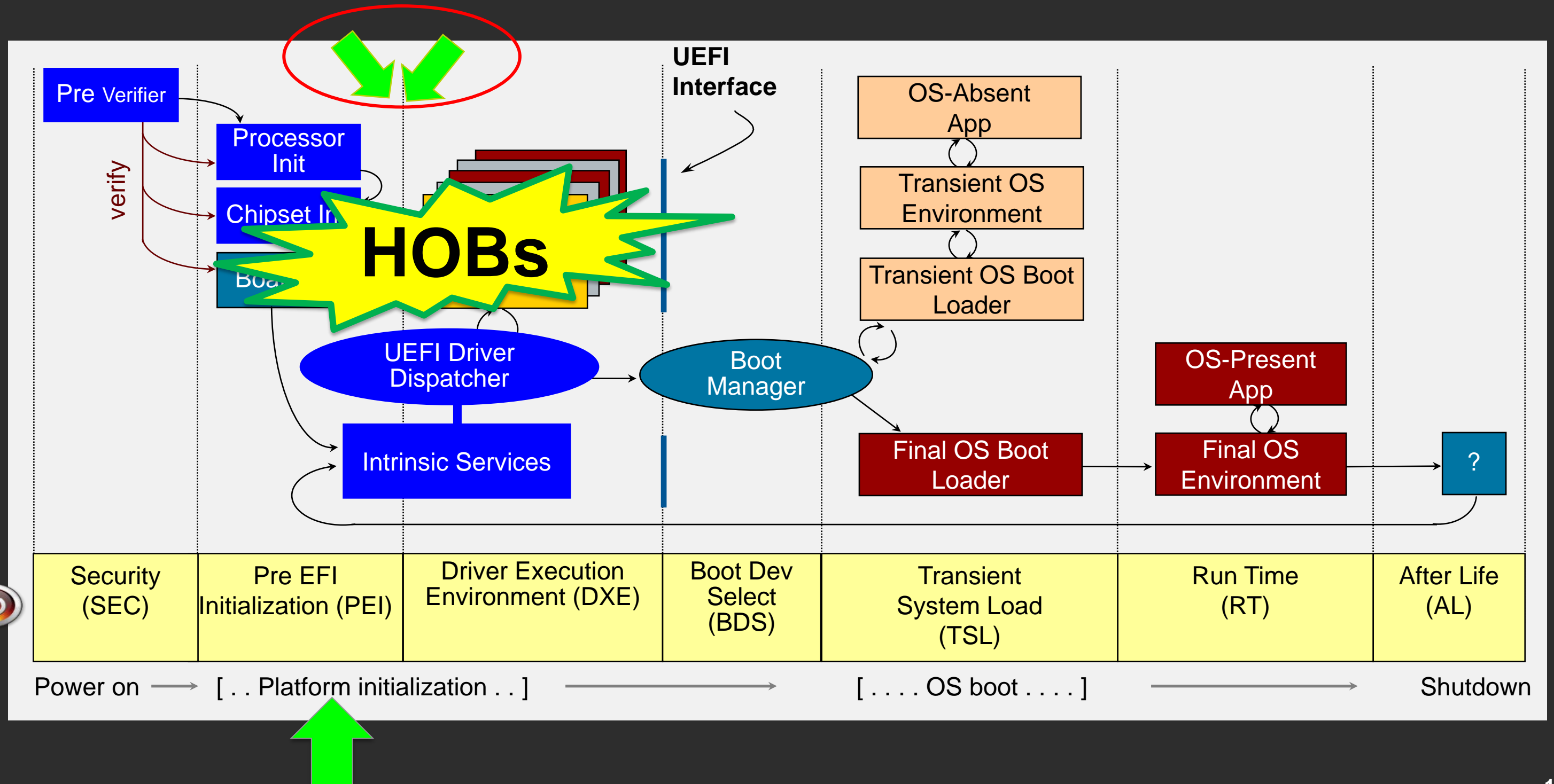
TERMS TO KNOW ABOUT THE FLASH DEVICE

- 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 - DXEIMPL

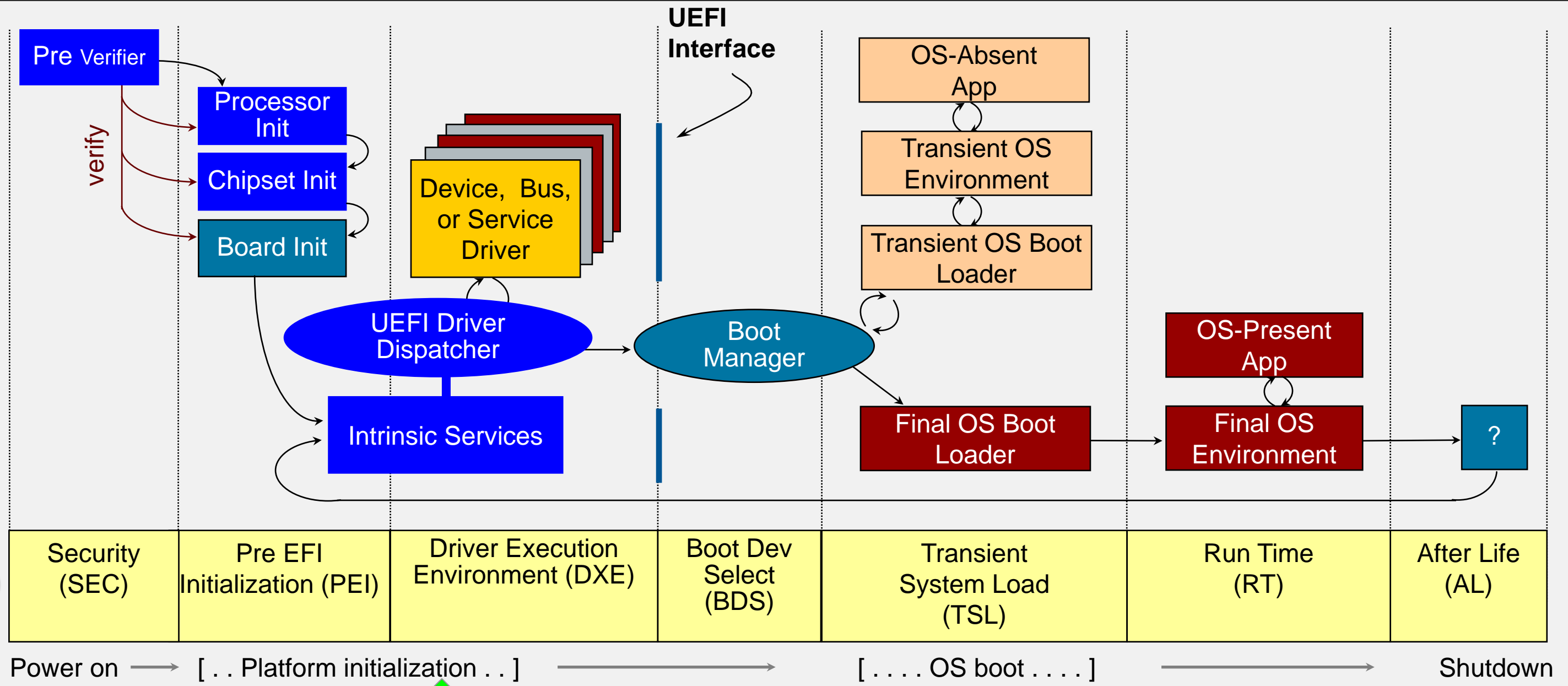


DXE IPL CHARACTERISTICS

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

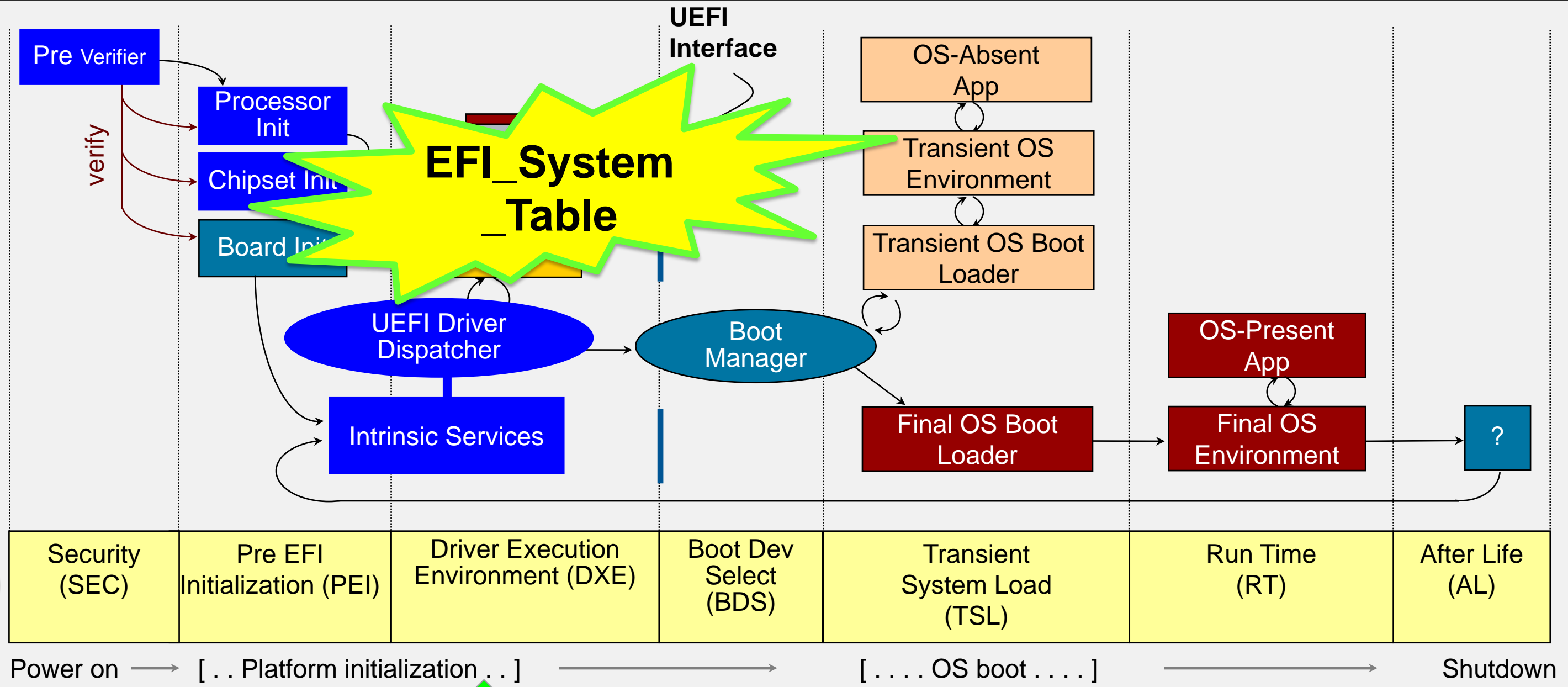
UEFI – PI & EDK II BOOT FLOW – DXE



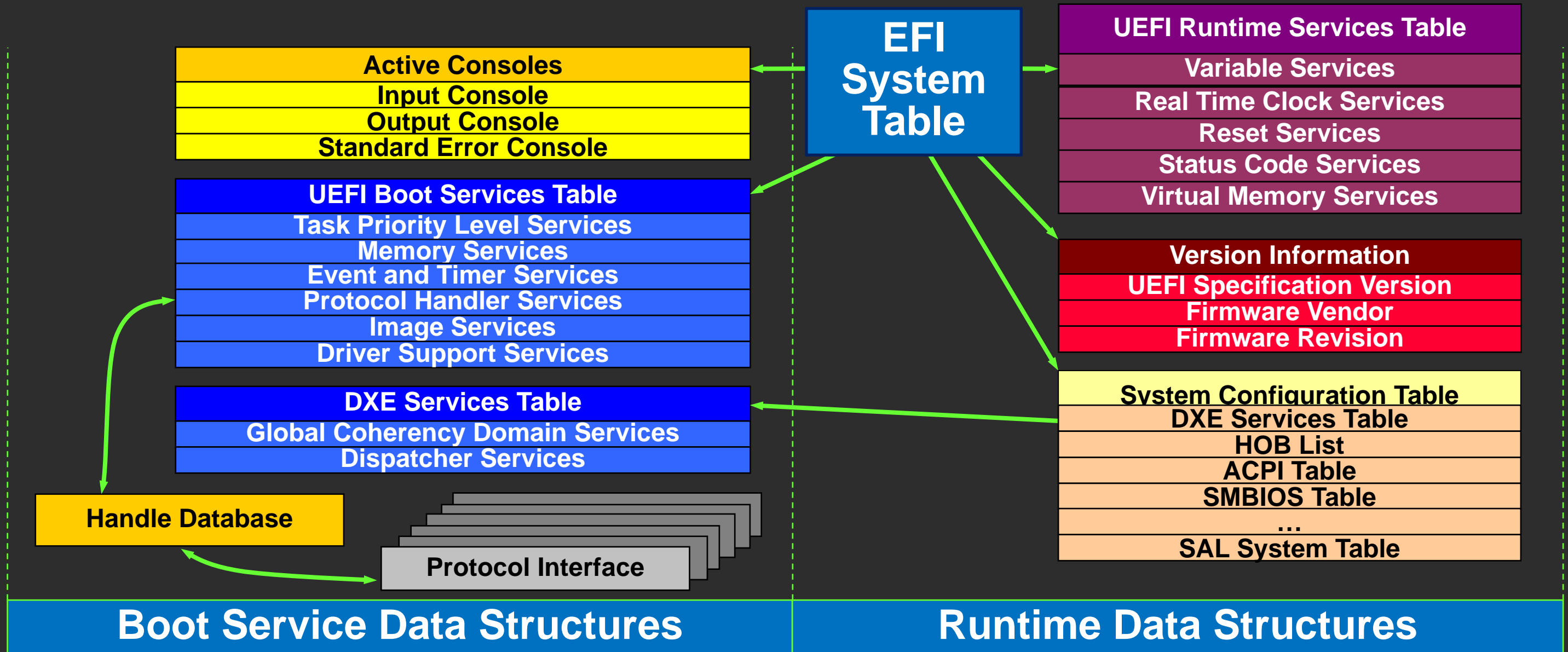
DXE CHARACTERISTICS & RESPONSIBILITIES

- 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 . . .

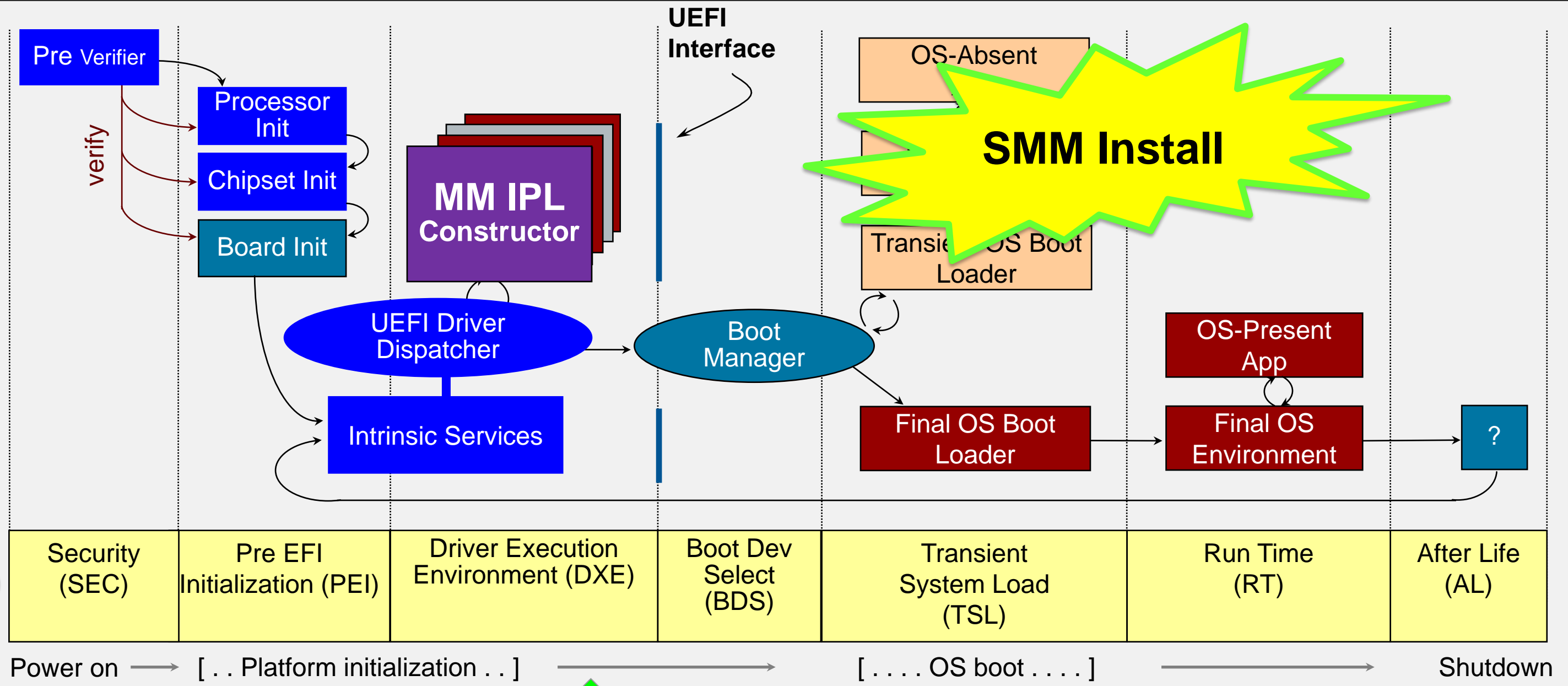
UEFI – PI & EDK II BOOT FLOW – DXE



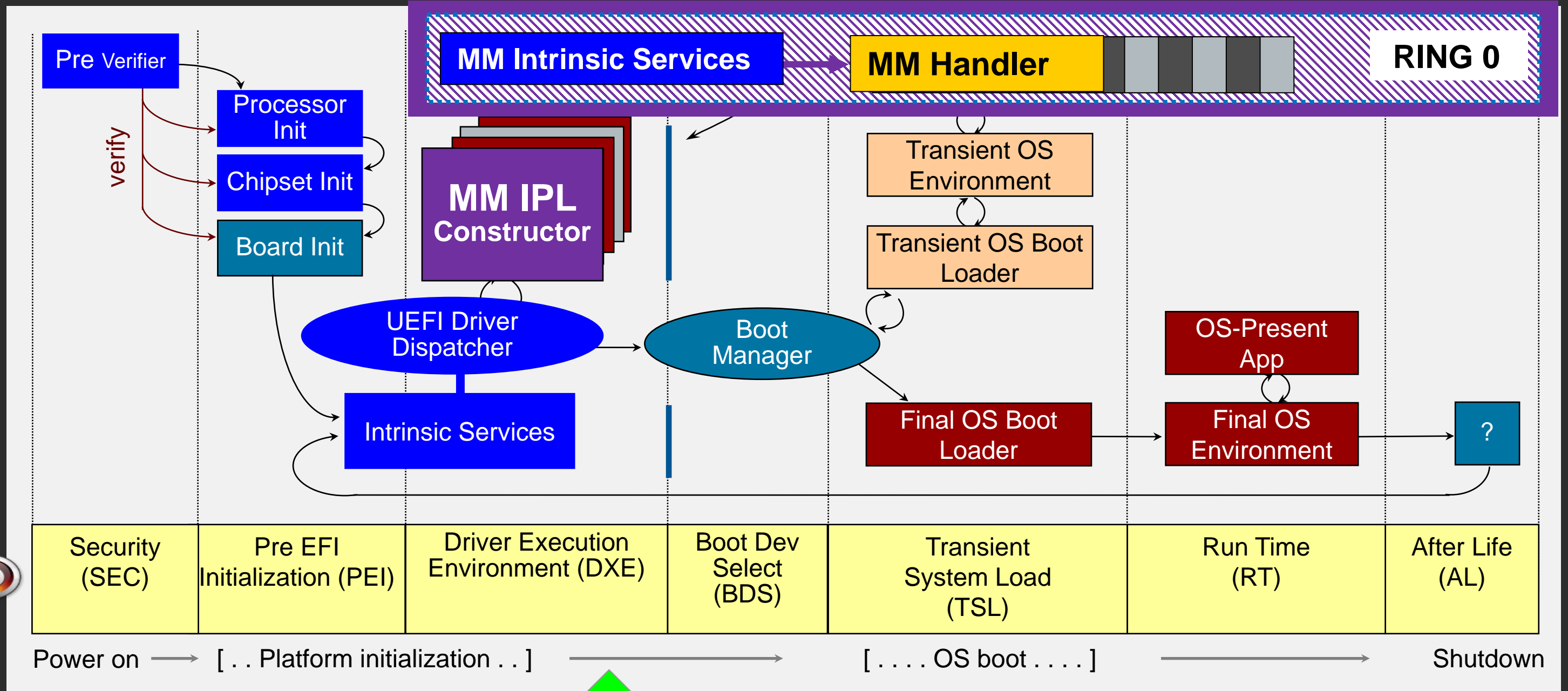
UEFI SYSTEM TABLE



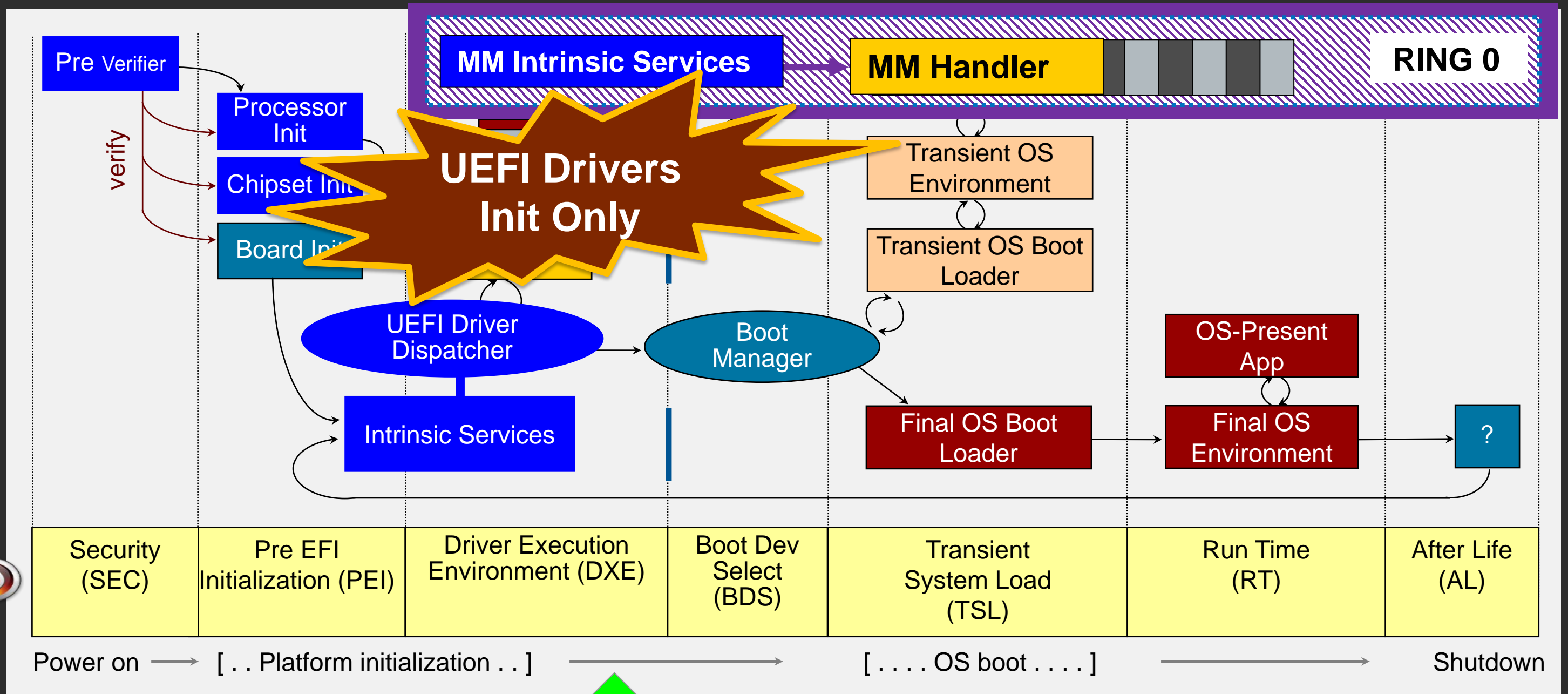
UEFI - PI & EDK II BOOT FLOW - SMM



UEFI - PI & EDK II BOOT FLOW - SMM



UEFI – PI & EDK II BOOT FLOW – DXE UEFI



Protocols

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

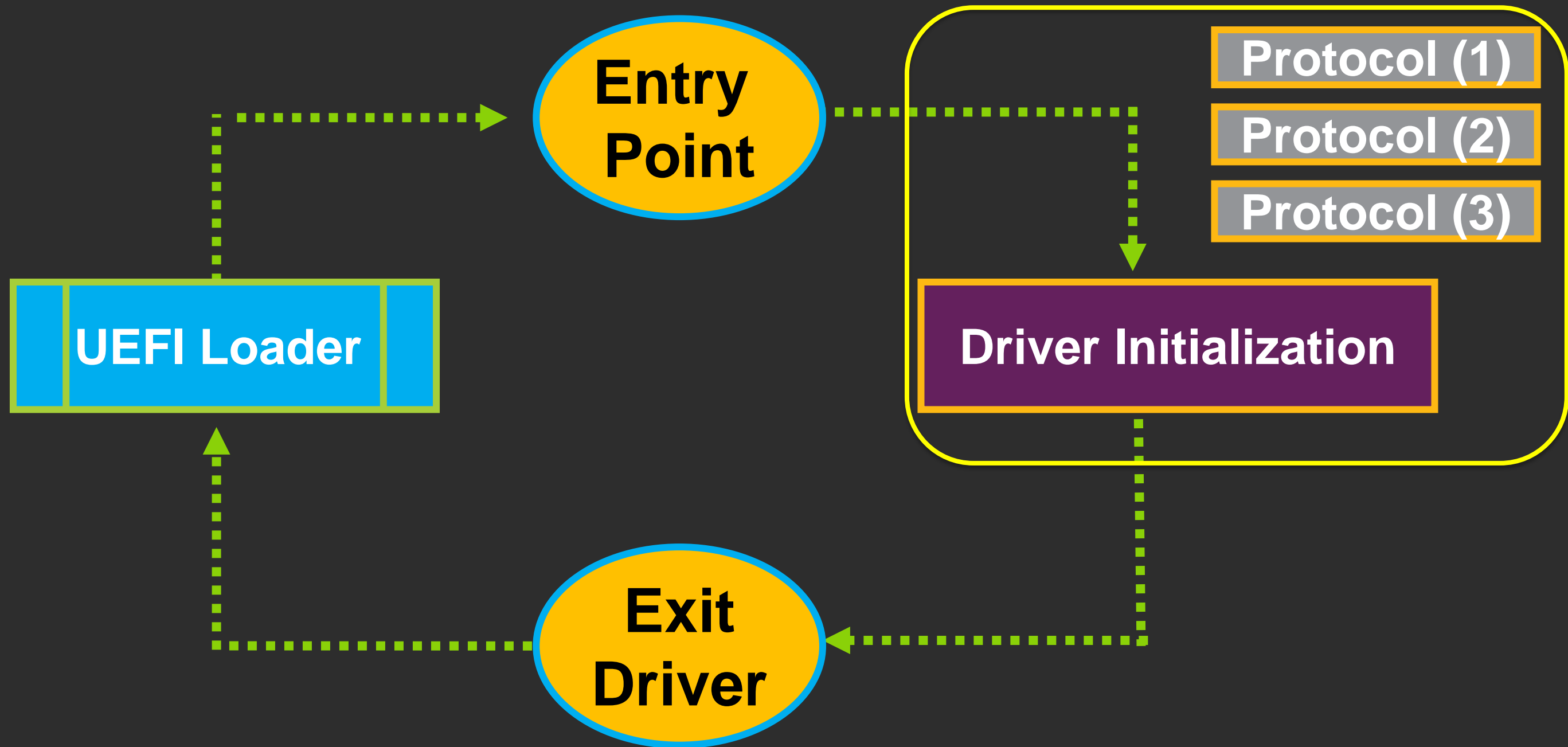
Handle Database

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

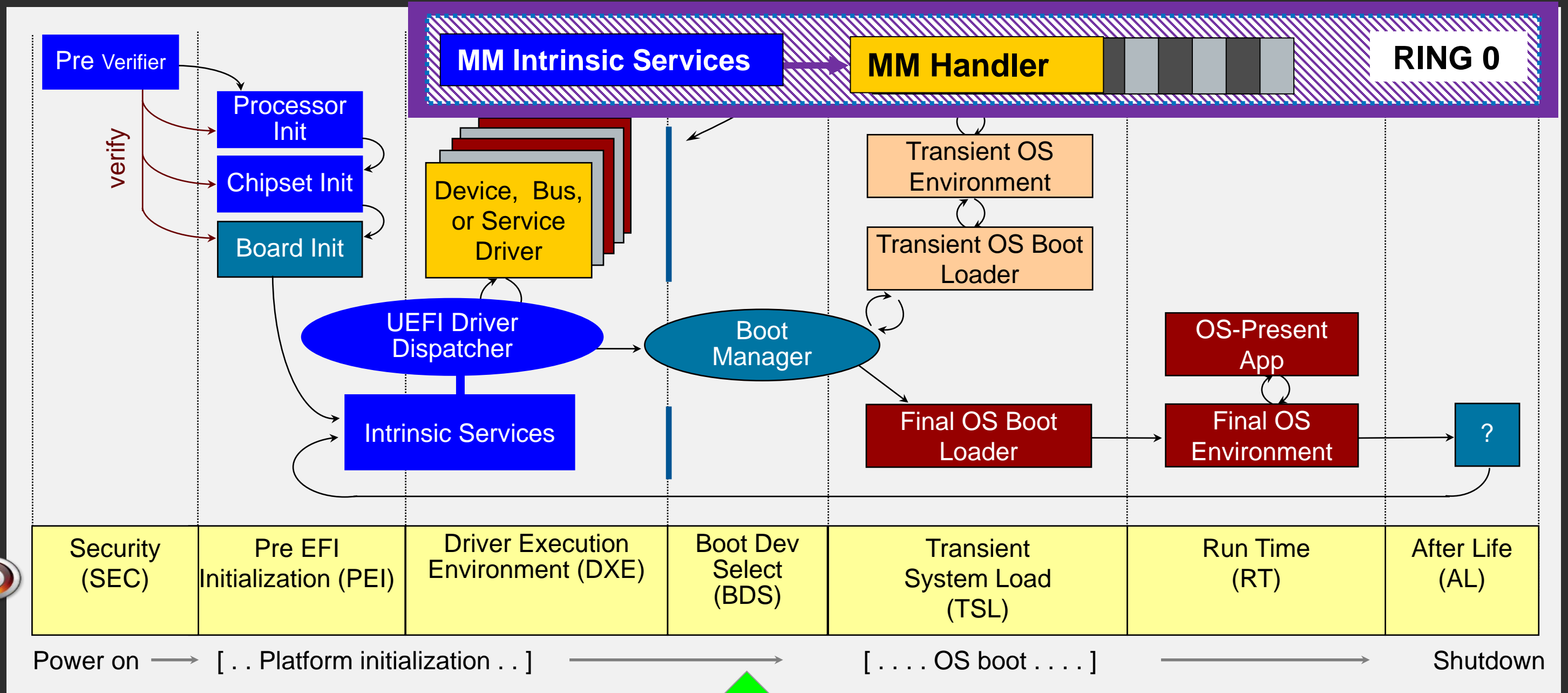
GUIDs

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

DXE Dispatcher Installs Drivers



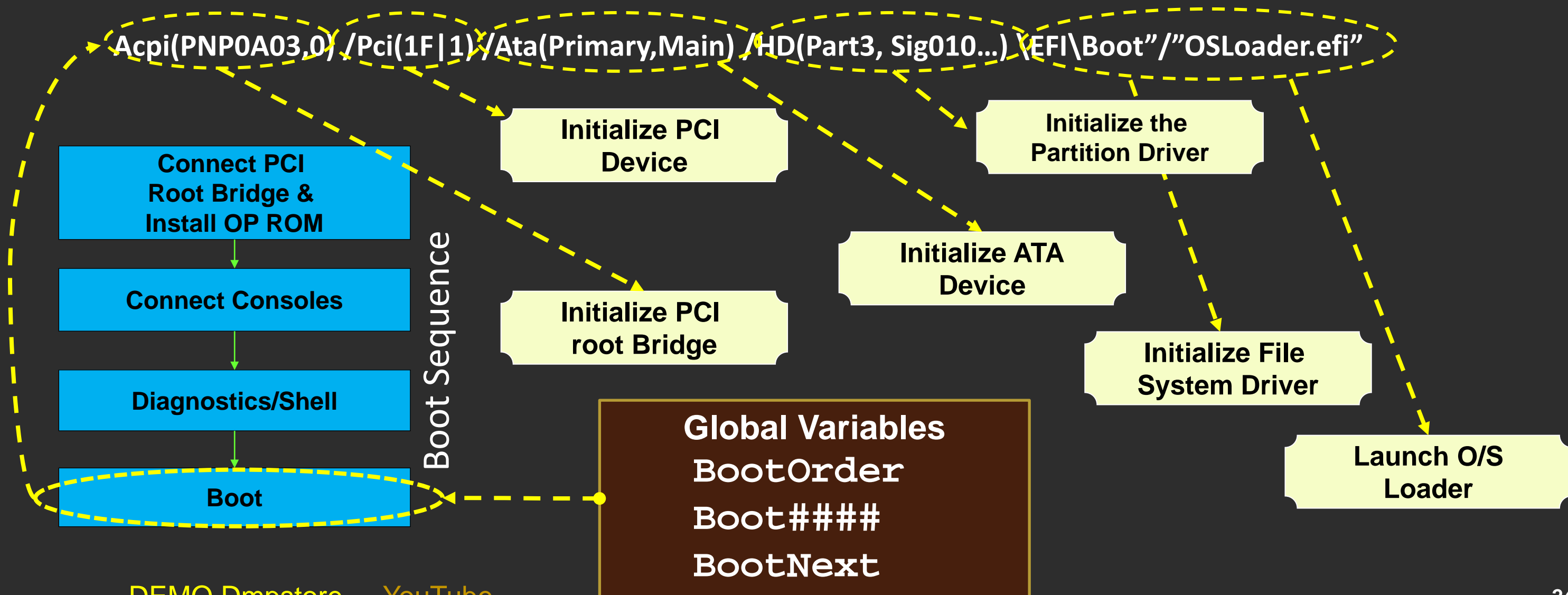
UEFI – PI & EDK II BOOT FLOW – BDS



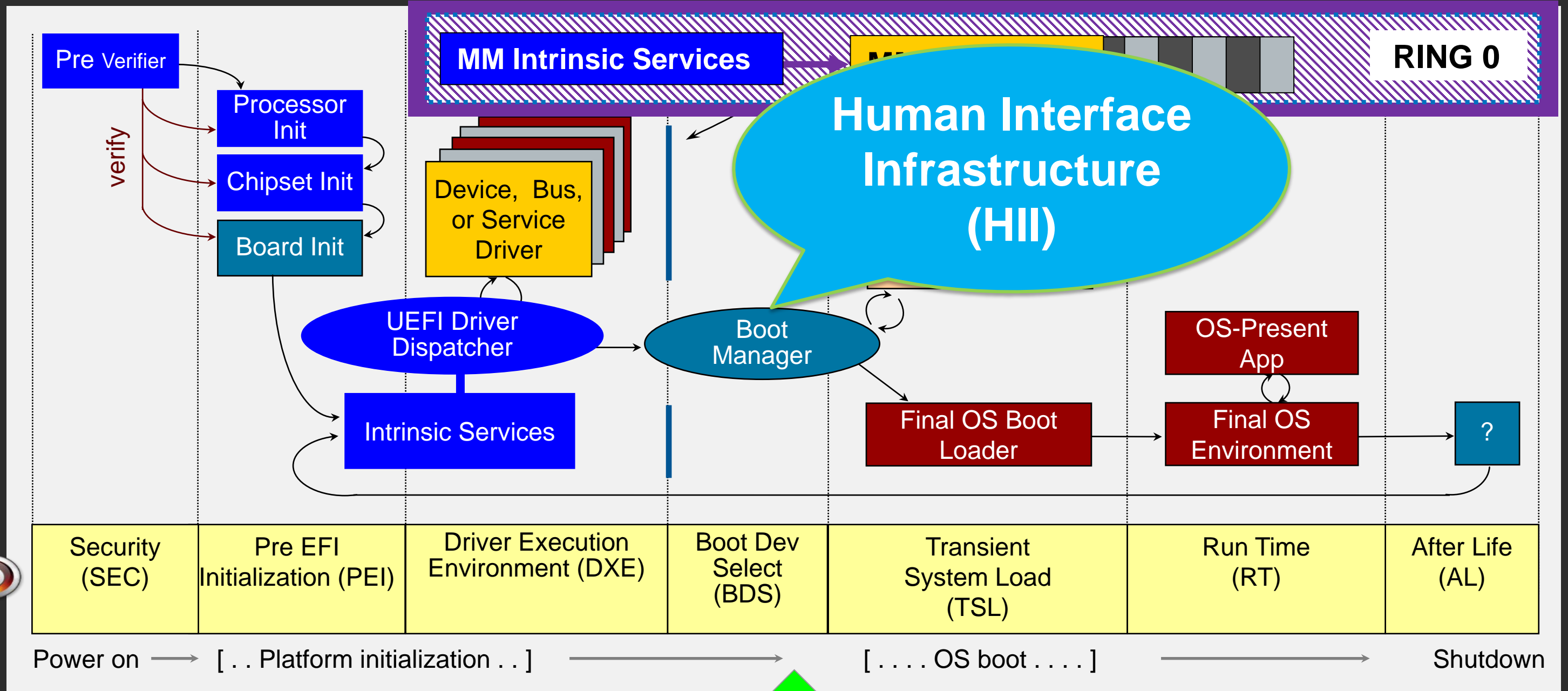
UEFI DEVICE PATH AND GLOBAL VARIABLES

The UEFI Device Path describes a boot target

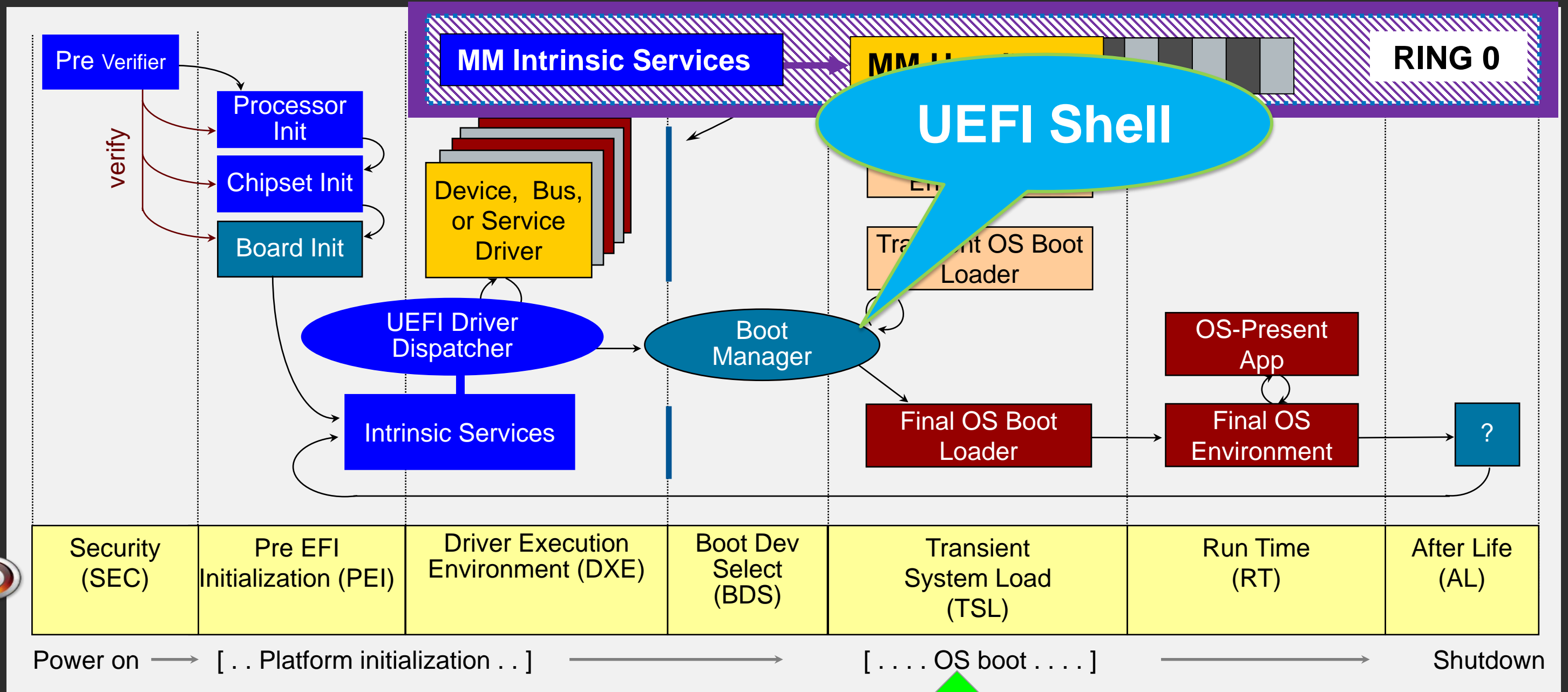
- Binary description of the physical location of a specific target



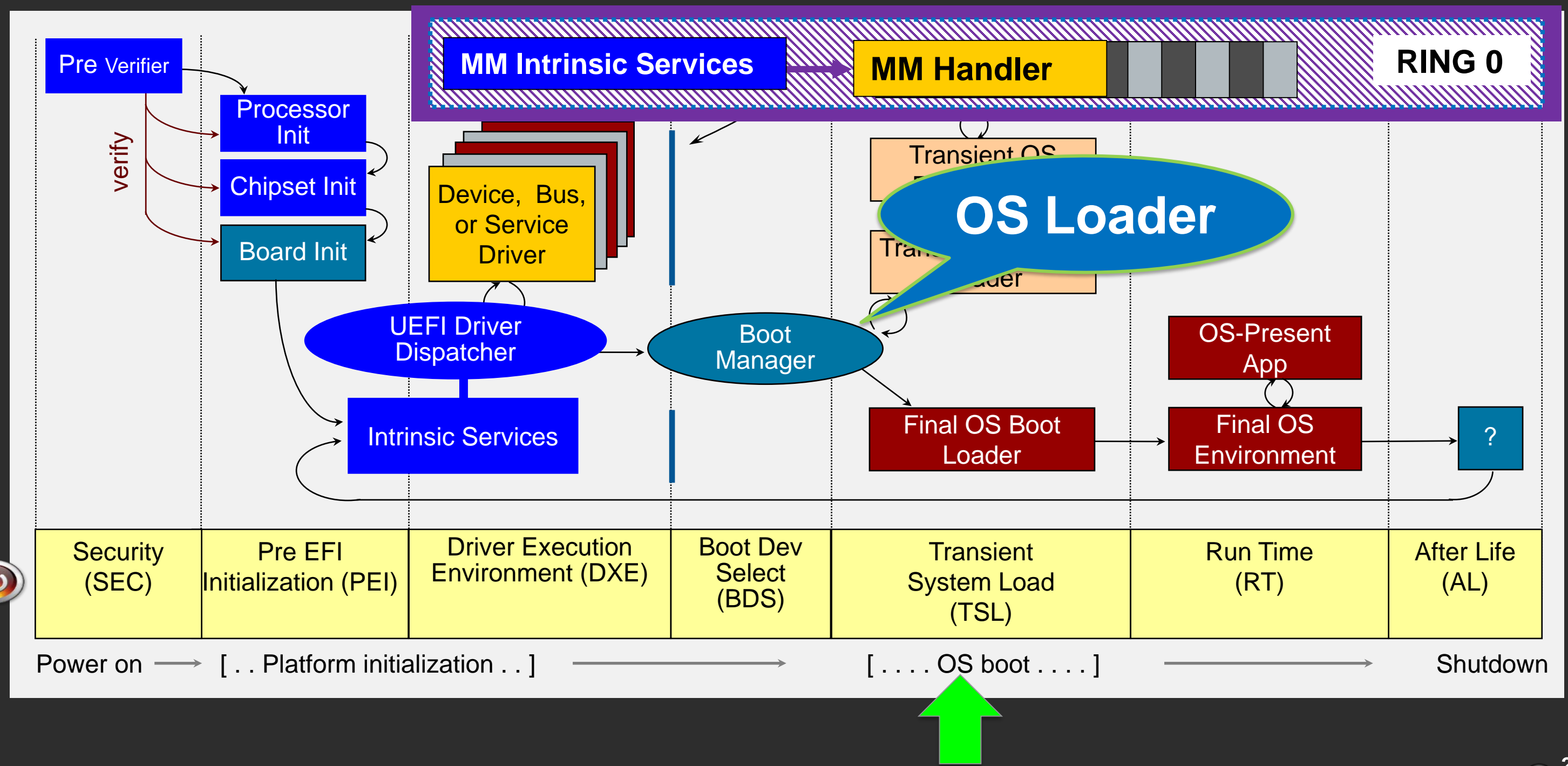
UEFI – PI & EDK II BOOT FLOW – HII



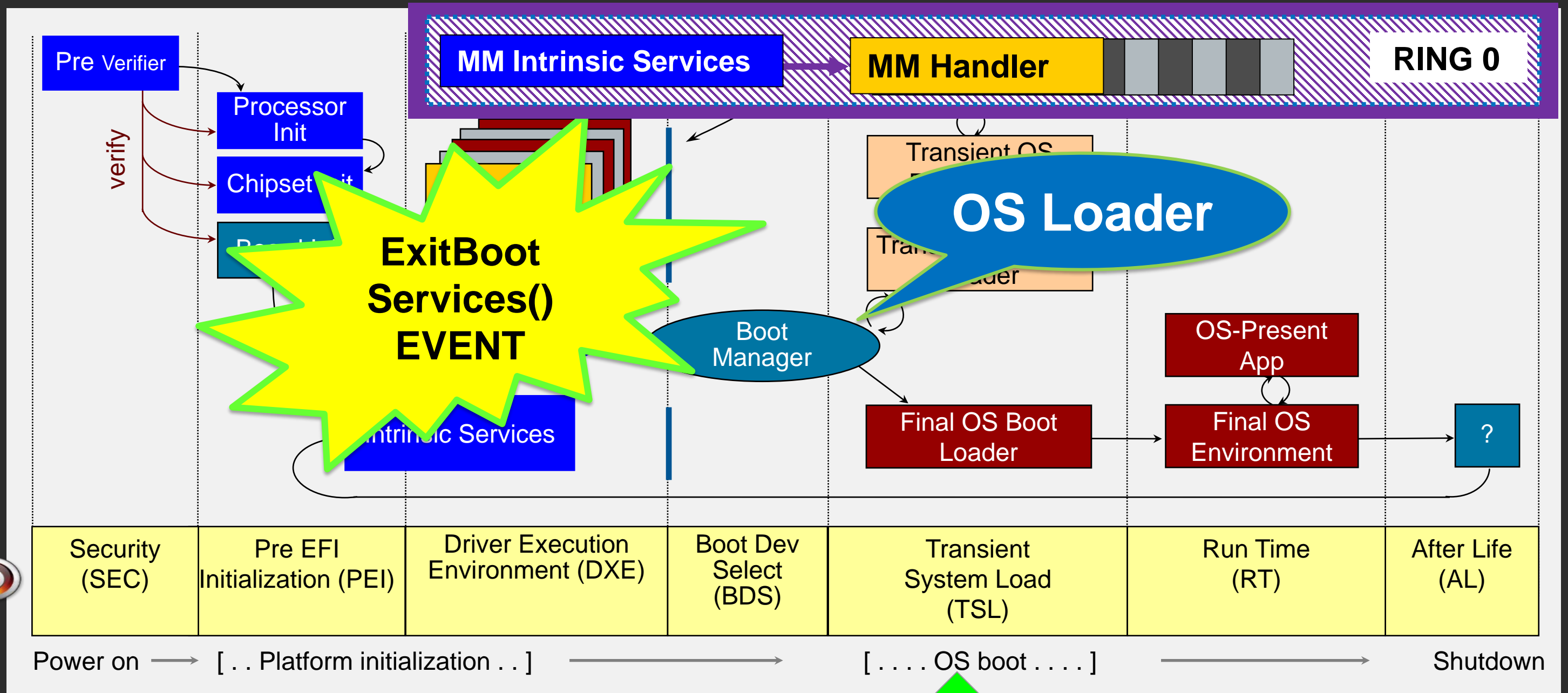
UEFI – PI & EDK II BOOT FLOW – TSL



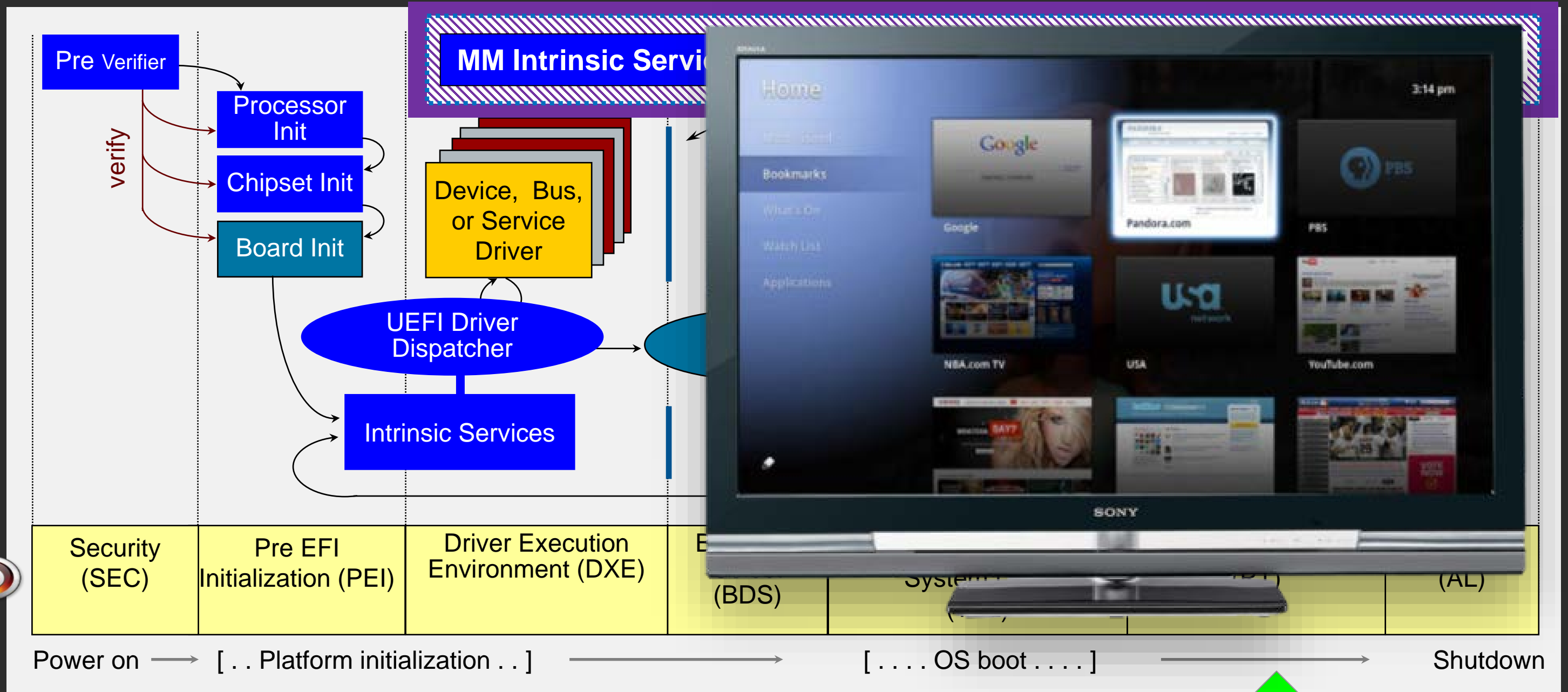
tianocore **UEFI – PI & EDK II BOOT FLOW – BOOT LOADER**



UEFI - PI & EDK II BOOT FLOW - EVENT



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



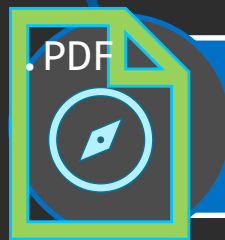
The Intel[®] Firmware Support Package (Intel[®] FSP)

What is Intel® Firmware Support Package?

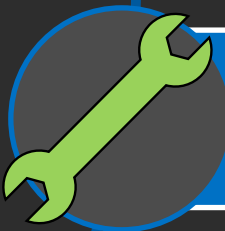
Includes:



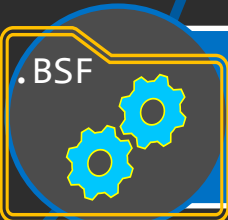
A binary firmware device (FD) file - contains multiple FSP Modules



An integration guide



A rebasing tool



A Boot Setting File (BSF) or YAML file for Configuration of the Updatable Product Data (UPD)

What Does Intel® FSP Provide?

- Provides 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

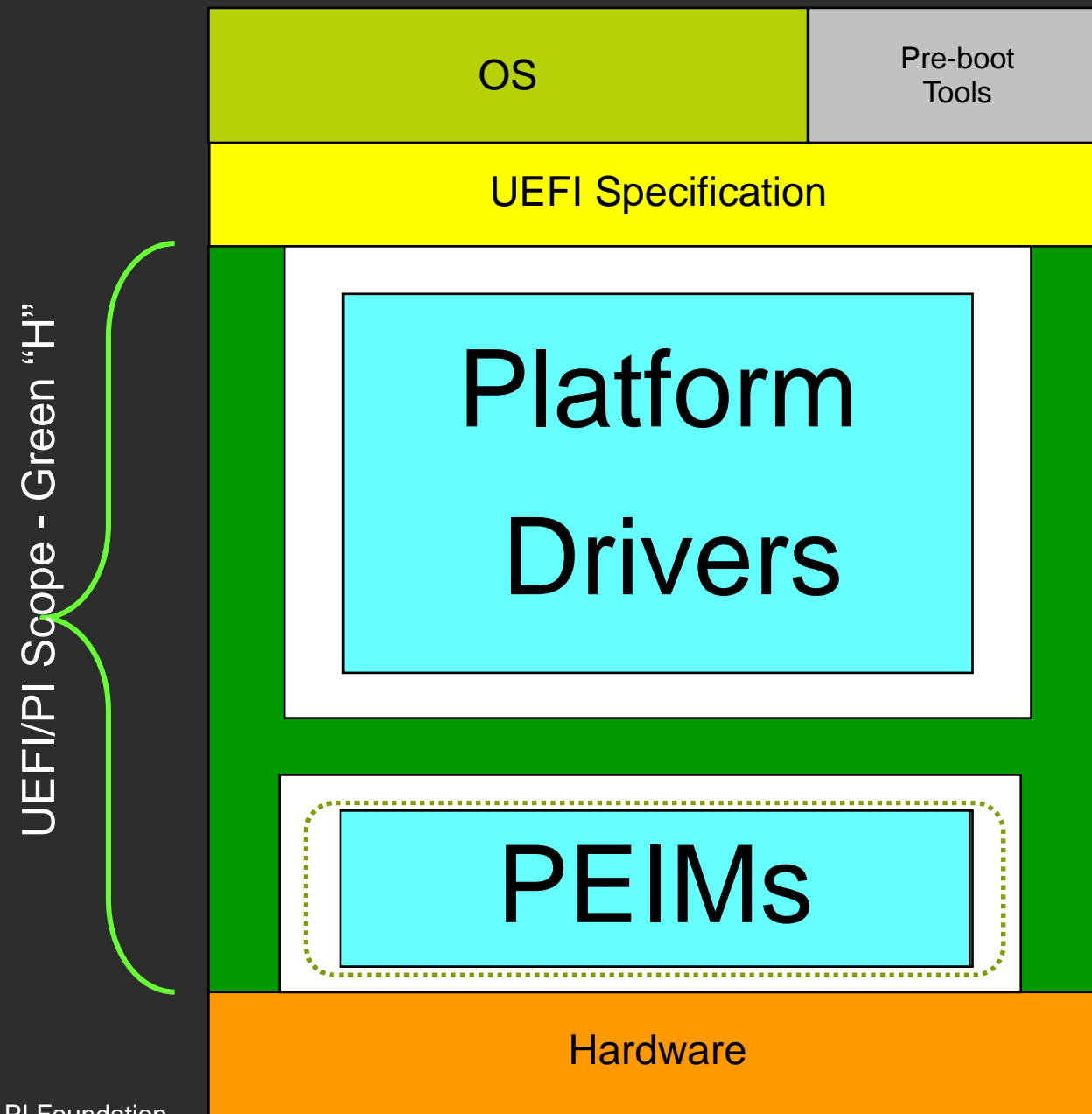
Intel FSP is currently available for the many Intel hardware-producing divisions

See: [About Intel FSP](#) (Intel® FSP 2.3 July 2021) ([2.4 Draft](#) February 2022)

White Paper Example: [Open Braswell - Design and Porting Guide](#)

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

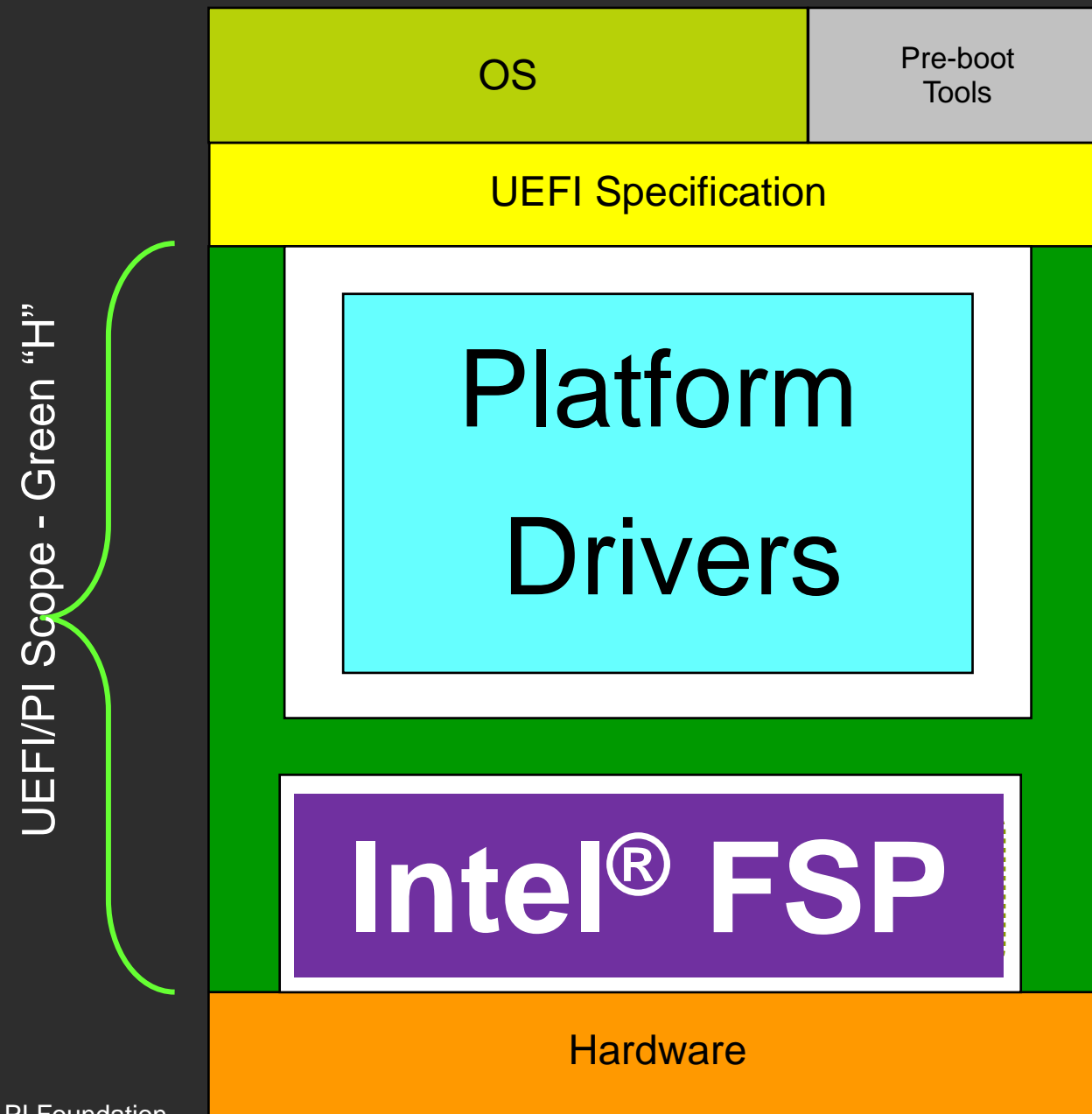
Intel® FSP to Open Source EDK II



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

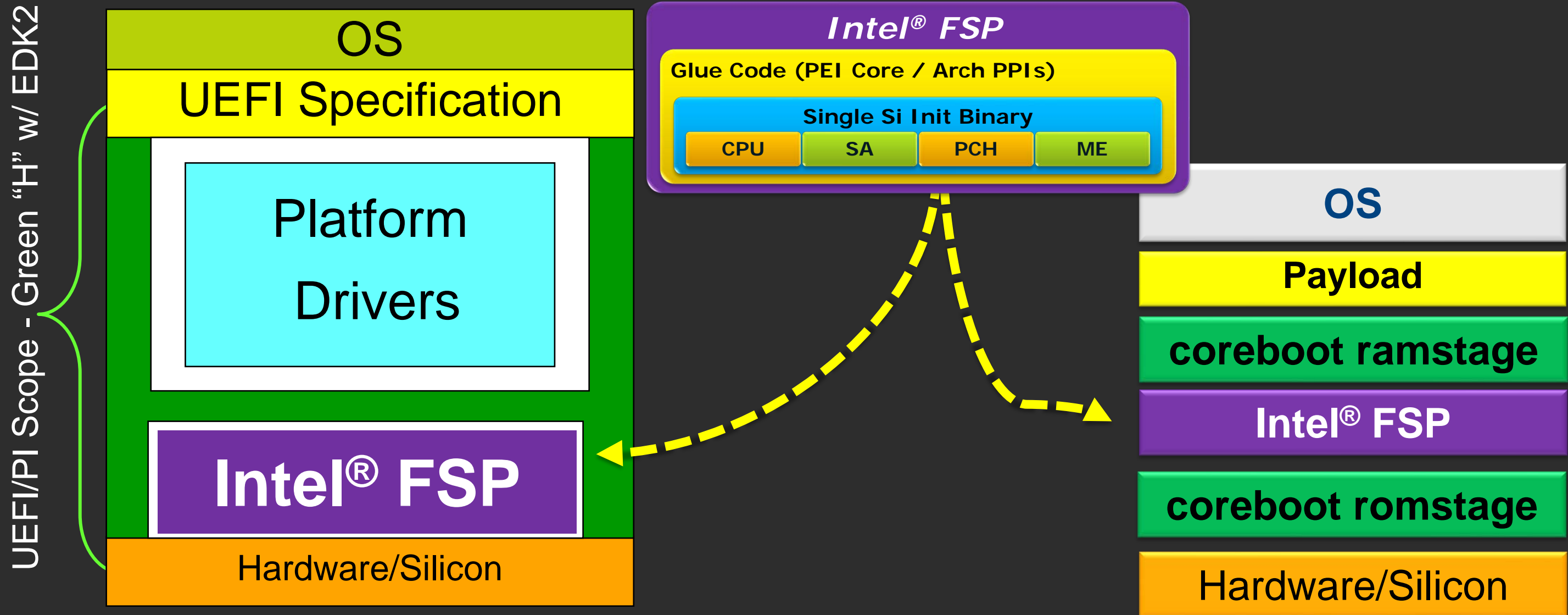


■ PEI/DXE PI Foundation
■ Modular Components

EDK II provides the framework (“Green H”)

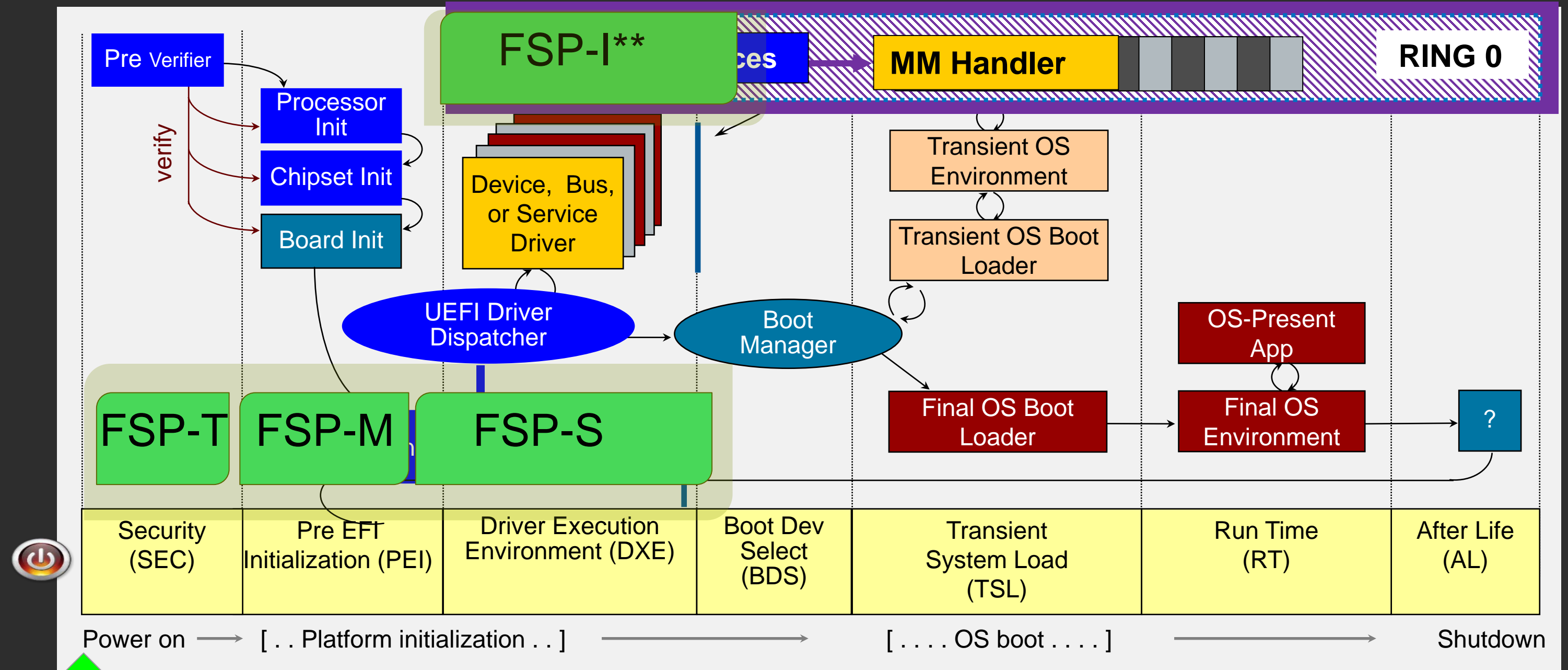
Intel® Firmware Support Package (Intel® FSP) provides low level of silicon initialization

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

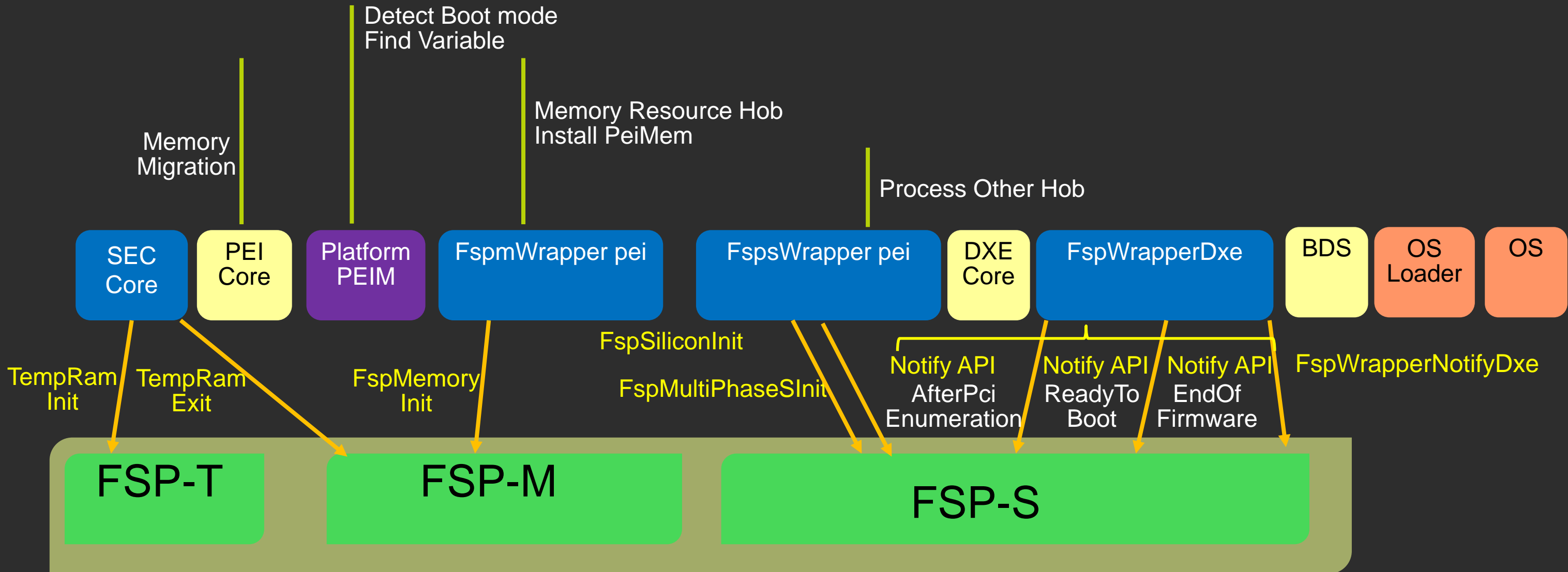


Intel FSP is independent of the bootloader solutions

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 v2.3 Boot Flow

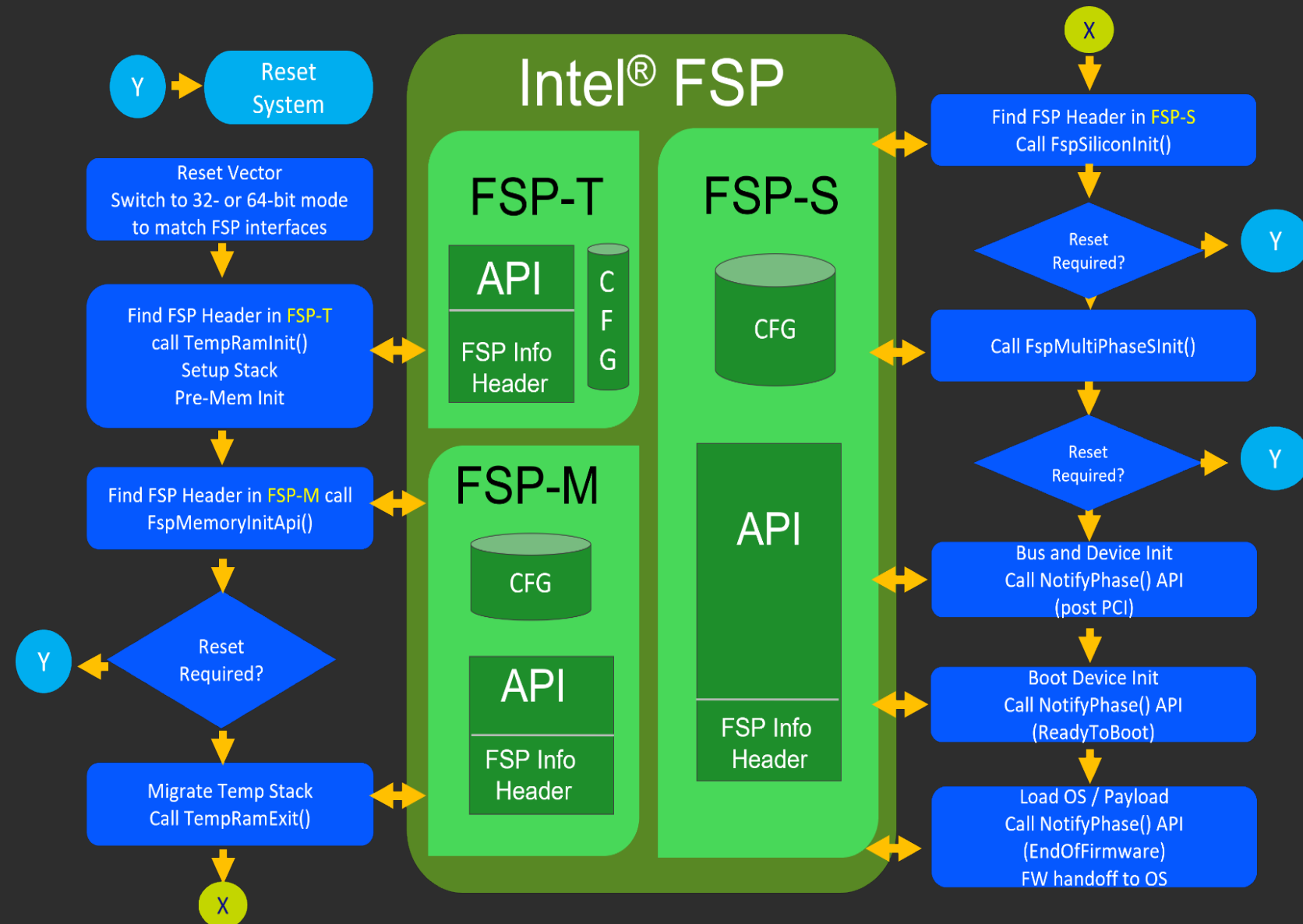
- Intel FSP 2.4 draft Out Soon

- Intel FSP 2.3 Spec

- Intel FSP version w/ Intel platforms

Version 2.4 Highlights:

- Extended FSP API calling convention to support both 32 and 64-bit interfaces
- System Management Mode (SMM) core and stand-alone handlers, such as server RAS
- Support Non-Volatile writes



Intel® FSP Spec 2.3

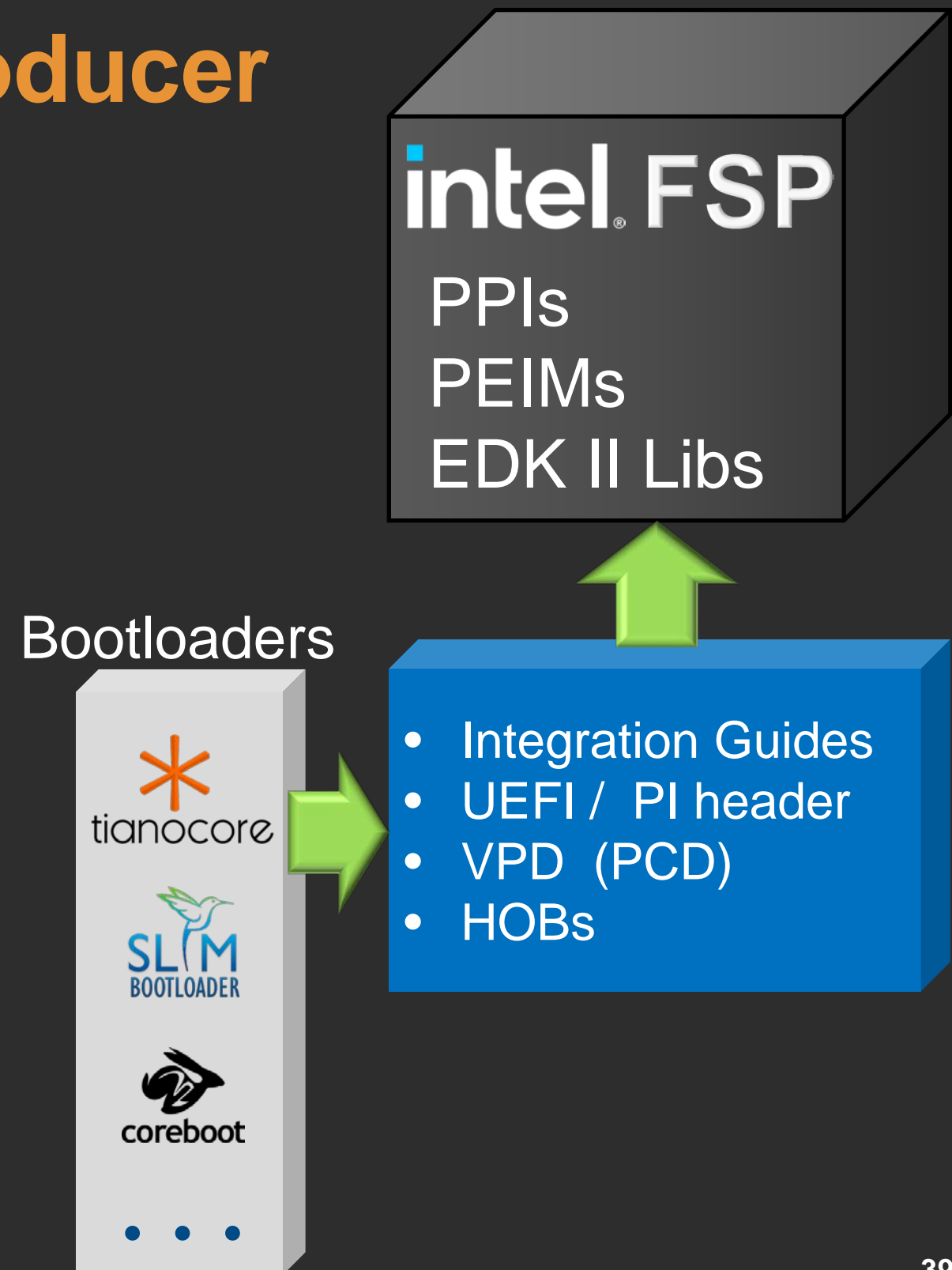
Intel® FSP Producer

- Examples of binary instances on <http://www.intel.com/fsp> w/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 \(EAS\) v2.3](#) [Link v2.0](#)

Resource:

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



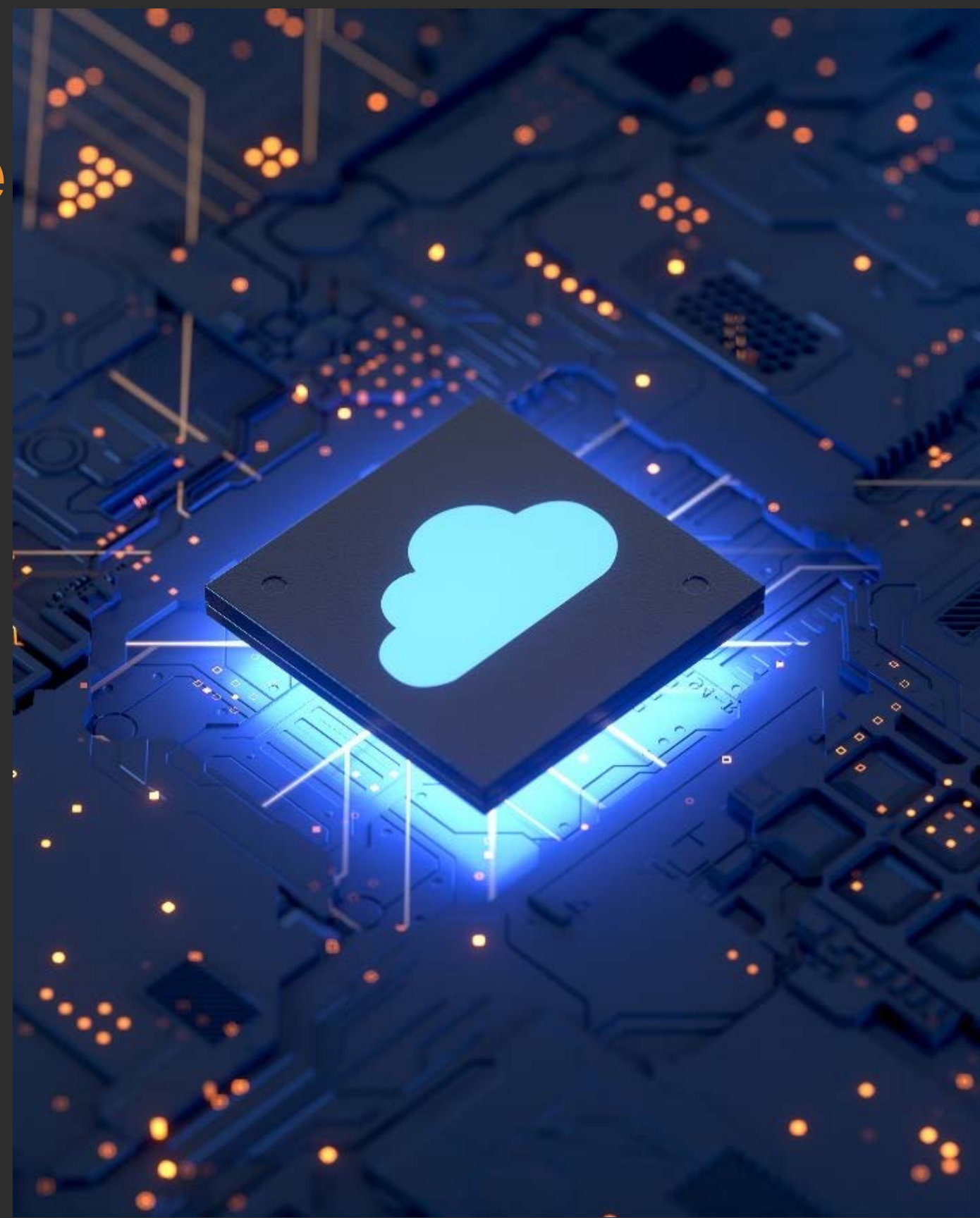
What's New in the Firmware Specifications and Open Source?

Universal Scalable Firmware (USF), UEFI & Tianocore.org

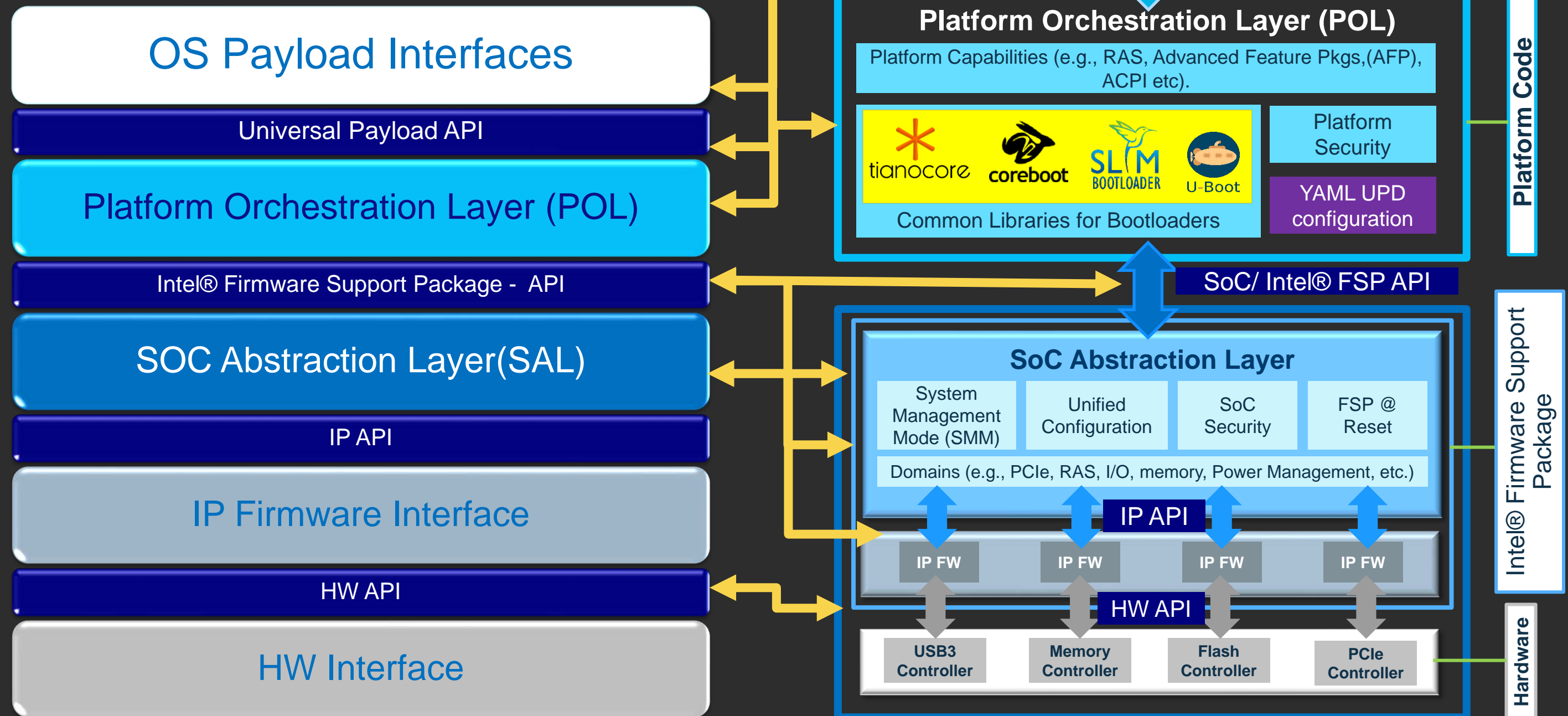
Universal Scalable Firmware (USF)

Next Gen firmware architecture to simplify and scale firmware development from edge to cloud

- Reduce firmware development complexity
- Scale development across CPU and XPU's
- Support industry bootloaders and OS payloads across market segments
- Improve firmware quality & security
- Accelerate innovation & time to value
 - <https://github.com/UniversalScalableFirmware>
- Specification:
 - <https://universalscalablefirmware.github.io/documentation/>
 - 0.7 released in Q1'22
 - Training: [YouTube Video Link](#)



Layers of Universal Scalable Firmware (USF)



Get Involved GitHub

Your feedback is welcome – join the community

USF on GitHub: <https://github.com/universalscalablefirmware>

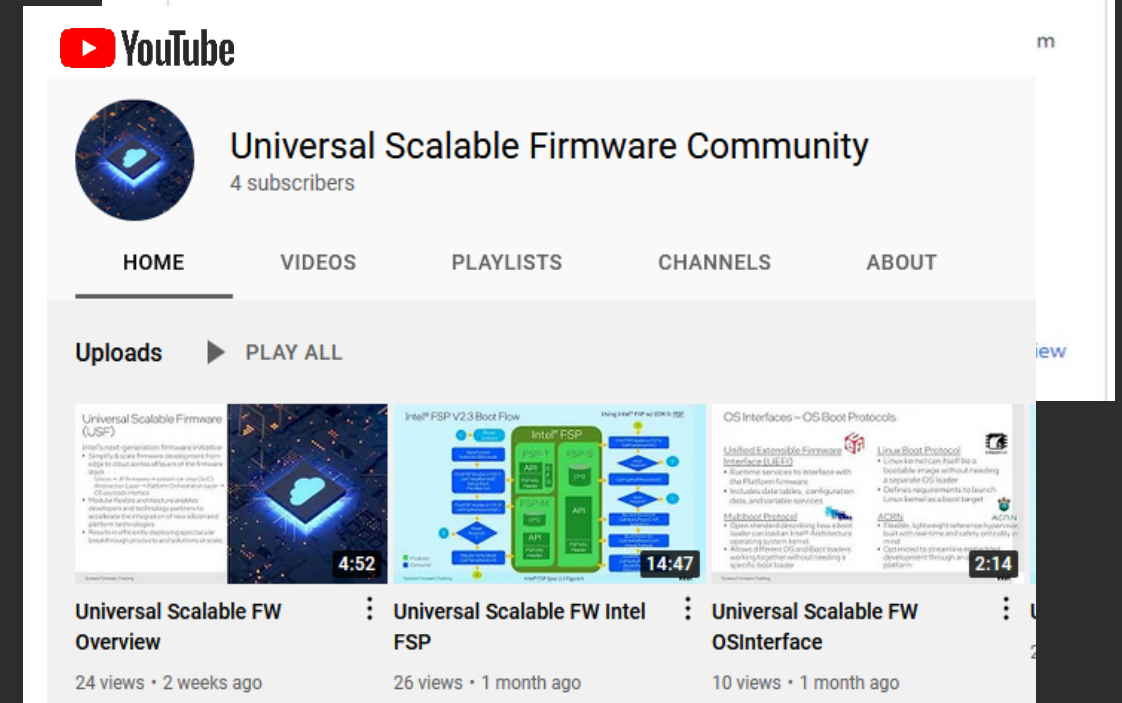
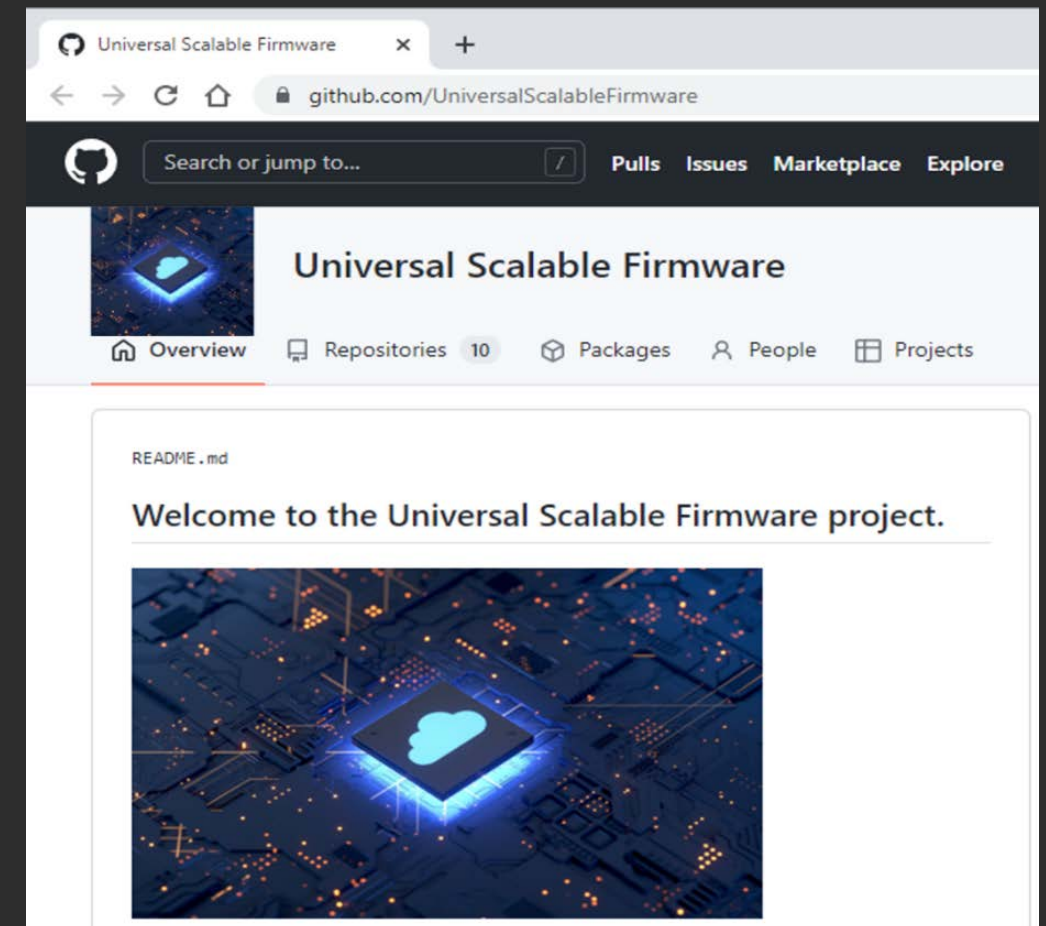
USF Specification:
<https://universalscalablefirmware.github.io/documentation/>

Subscription for email discussions & feedback:
<https://universalscalablefirmware.groups.io/g/discussion>

USF Point of Concept development, (GitHub pull requests Welcome)
https://github.com/UniversalScalableFirmware/fspsdk/tree/sbl_qemu_fsp_x64_fspt64

Training : [YouTube Video Link](#)

USF will be deployed over several product generations to ensure the latest technology advancements are quickly available to technology partners as they are ready at each layer.



Latest UEFI Specifications



Unified Extensible Firmware Interface Forum



UEFI Forum Release UEFI 2.10 and ACPI 6.5 August 29, 2022

- Introducing UEFI Conformance Profiles, allows support for more types of platforms & implementation codebases
- Addition of Confidential Computing extension to support - [Intel® Trust Domain Extensions \(Intel® TDX\)](#)
- Adding support Crypto Agility and support for enabling SHA 384/SHA-512 signing of Authenticated Variables
- Updates to protocols relating to memory protection
- Updates to the CXL memory support
- “Code First” Process - acknowledges contributions of open-source communities

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](https://www.tianocore.org/wiki/)



Tag: edk2-stable202208 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







The screenshot shows the Tianocore Bugzilla Main Page in a web browser. The browser's address bar displays <https://bugzilla.tianocore.org>. The page title is "TianoCore Bugzilla - Main Page" with a version number of 5.0.4. The navigation bar includes links for Home, New, Browse, Search, Reports, Preferences, Administration, and a Log out button for the user laurie.jarlstrom@intel.com. The main content area features a "Welcome to TianoCore Bugzilla" message and four large icons: "File a Bug" (document icon), "Search" (magnifying glass icon), "User Preferences" (person icon), and "Documentation" (question mark icon). At the bottom, there is a search bar with the placeholder text "Enter a bug # or some search terms" and a "Quick Search" button.



Bugzilla

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
-  Intel® FSP will initialize the processor, chipset and memory
-  The USF, UEFI.org & Tianocore.org for Specs and Open source

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.



BACKUP

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

SMM - Platform Runtime Mechanism (PRM)

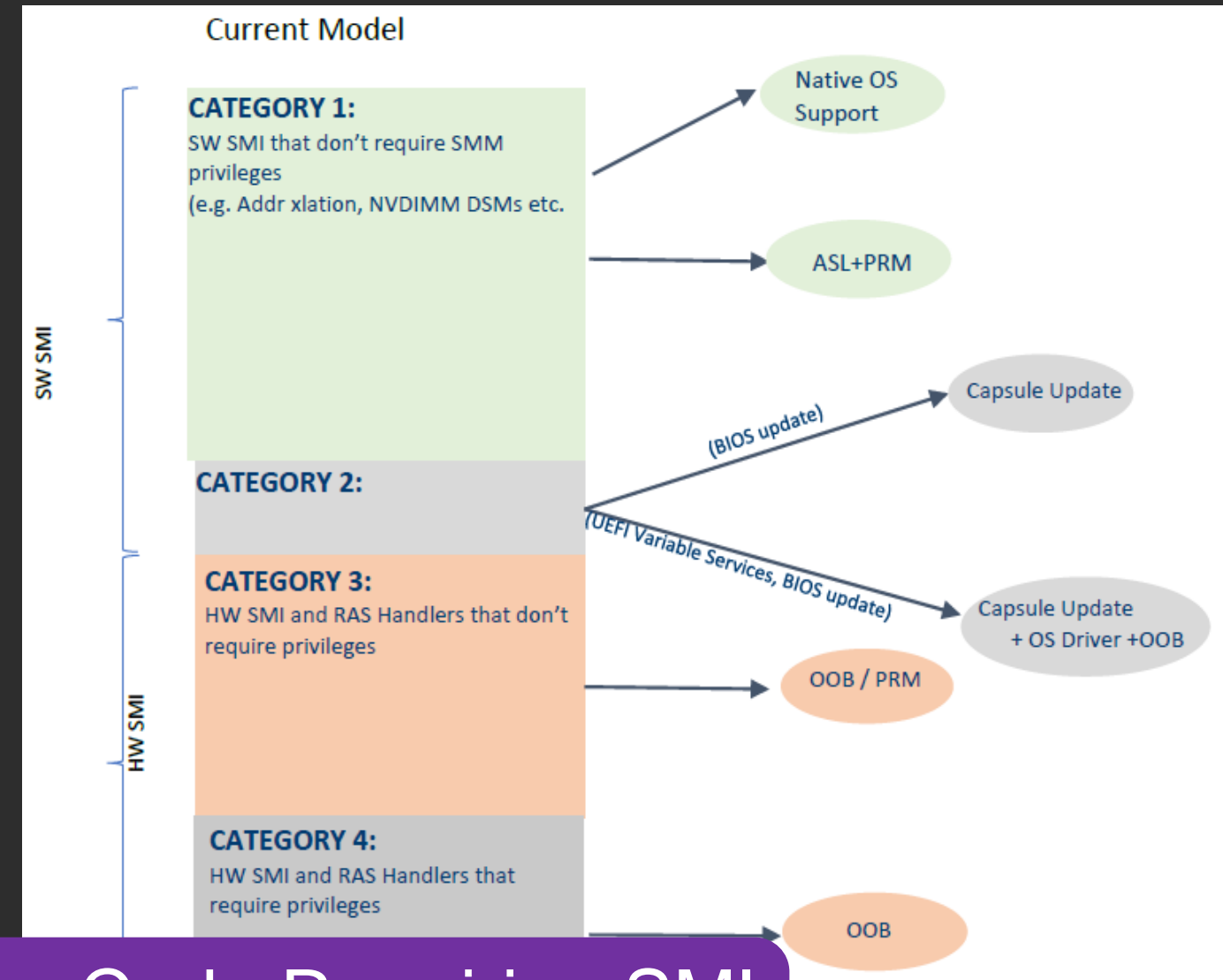
ACPI Spec added a Table for Platforms to have SMI handlers [ACPI PRM Spec](#)

Category 1 SMM handler will be migrated to use PRM.

Category 2 SMM Handlers are mainly related to UEFI authenticated variable services. Not in scope for PRM Specification

Category 3 Certain SMM Handlers can be handled by PRM as explained in PRM Spec.

Category 4 SMM Handlers are mainly related to Uncorrectable Hardware Errors and advanced RAS features. Not in scope for PRM Specification



Alternate Means for Invoking Platform Code Requiring SMI
Execution Moved to ACPI Spec

es of SMI Handlers

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

