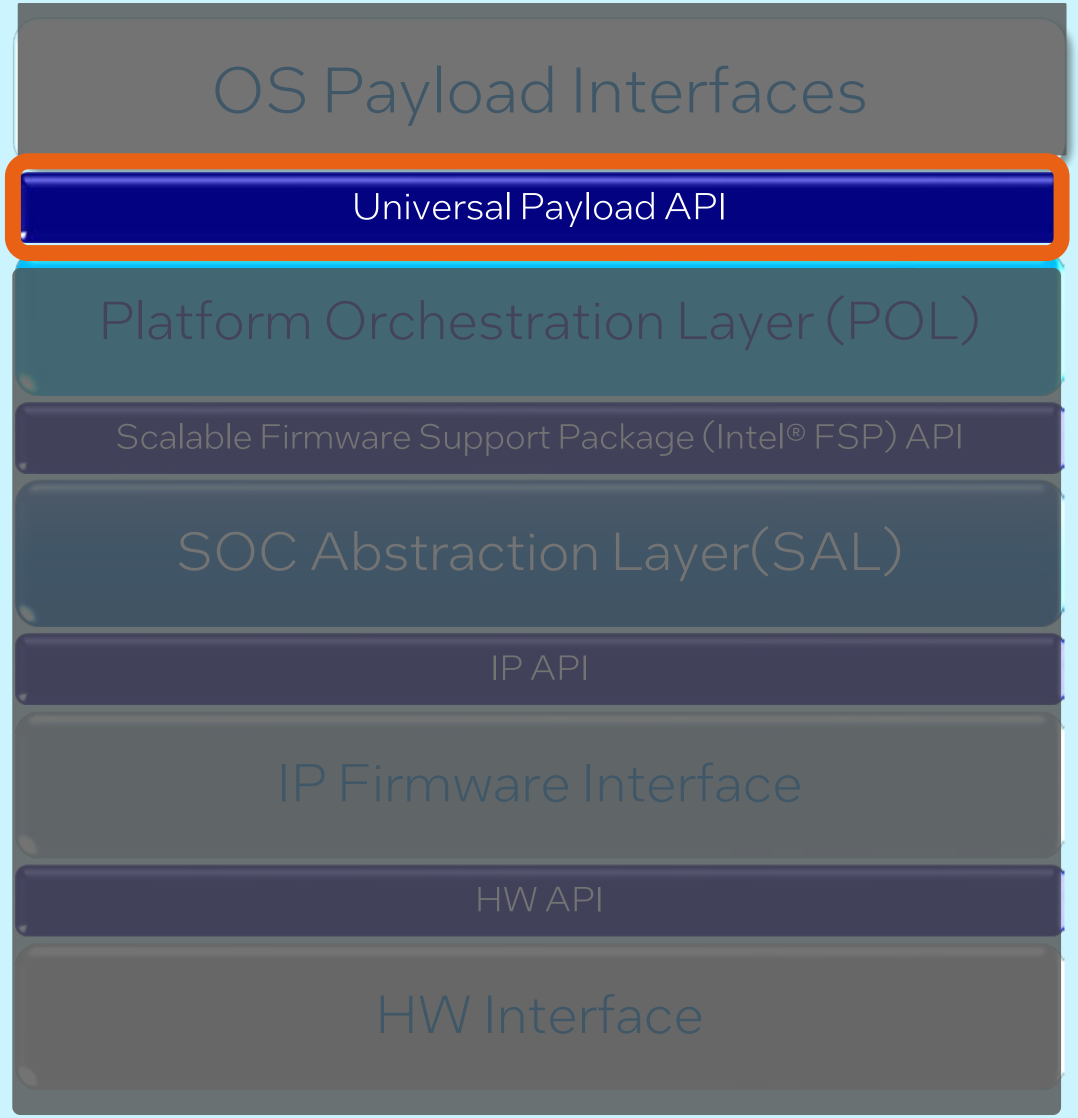
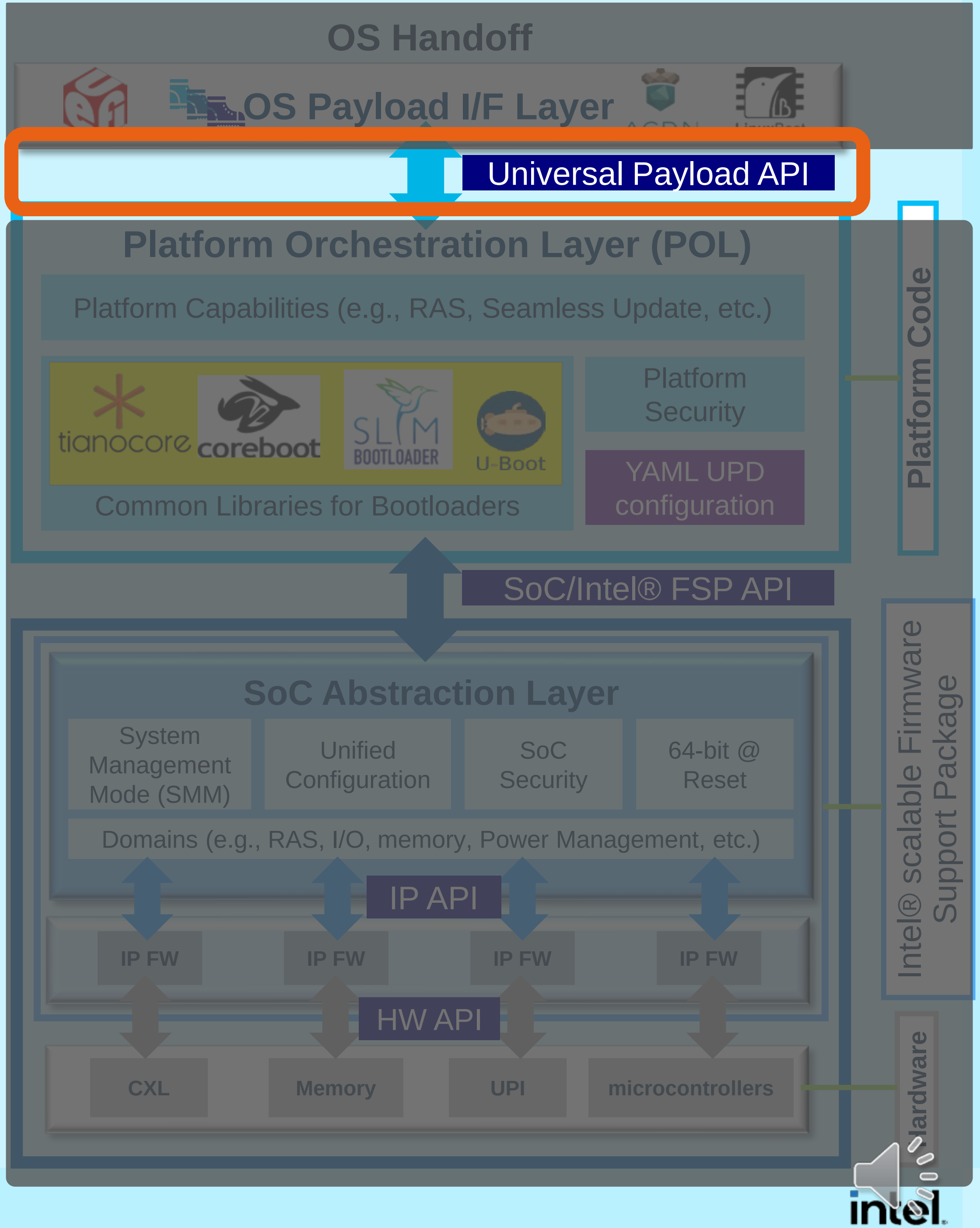
System Firmware Training

Universal Scalable Firmware (USF) :

Universal Payload

Intel Corporation


OS Payload Interfaces
Universal Payload API
Platform Orchestration Layer (POL)
Scalable Firmware Support Package (Intel® FSP) API
SOC Abstraction Layer(SAL)
IP API
IP Firmware Interface
HW API
HW Interface


OS Handoff
OS Payload I/F Layer
Universal Payload API
Platform Orchestration Layer (POL)
Platform Capabilities (e.g., RAS, Seamless Update, etc.)
tianocore
coreboot
SLIM BOOTLOADER
U-Boot
Common Libraries for Bootloaders
Platform Security
YAML UPD configuration
Platform Code
SoC/Intel® FSP API
SoC Abstraction Layer
System Management Mode (SMM)
Unified Configuration
SoC Security
64-bit @ Reset
Domains (e.g., RAS, I/O, memory, Power Management, etc.)
IP API
IP FW
IP FW
IP FW
IP FW
HW API
CXL
Memory
UPI
microcontrollers
Intel® scalable Firmware Support Package
Hardware

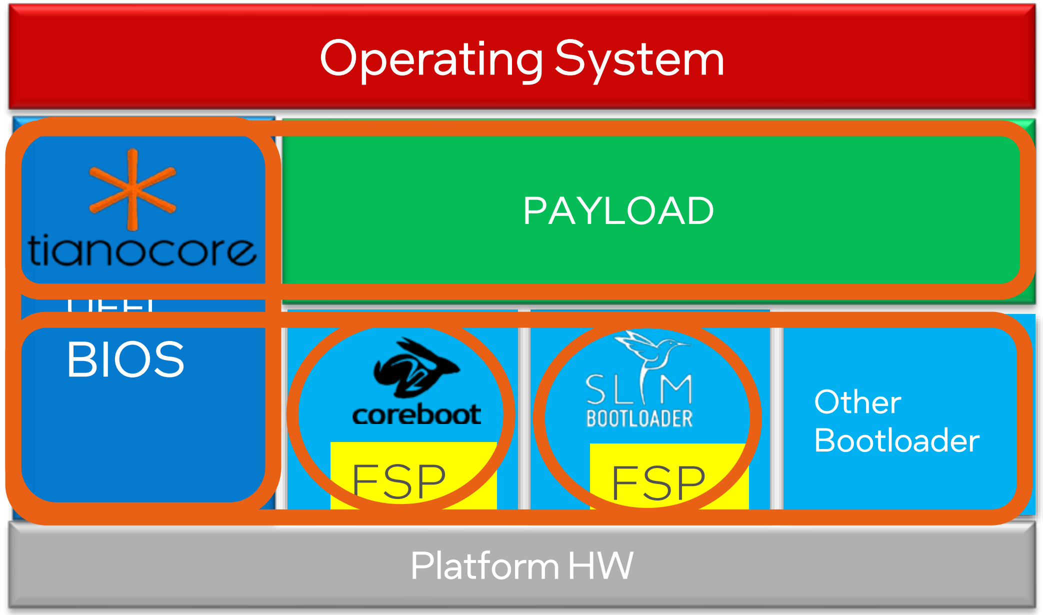
Payload overview

Bootloader

- Platform initialization including memory, silicon, GPIO, ACPI, etc.
- Coreboot, Slim Bootloader (SBL), Uboot

Payload

- Boot media initialization, file system, OS boot, etc.
- EDK II UEFI Payload, LinuxBoot, Uboot



Payload is part of boot firmware to initialize boot media and boot OS

Goal 1: Platform independent

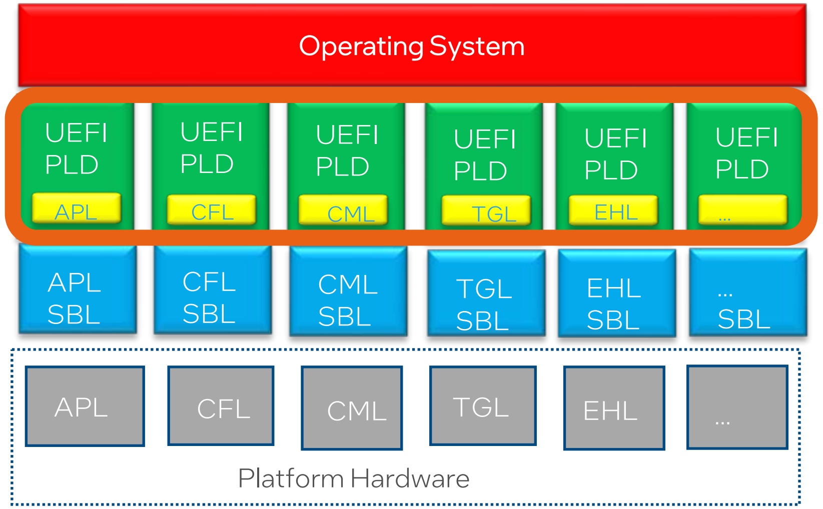


Fig1. Before Change

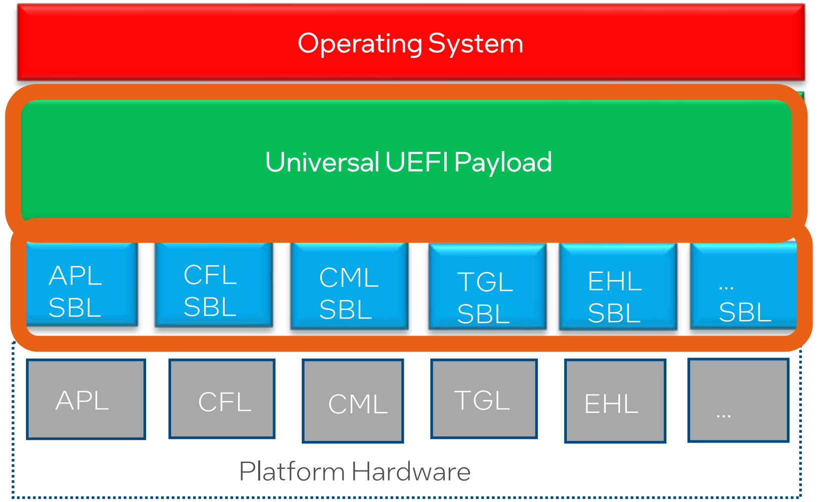


Fig2. After Change

Universal payload is platform independent

Goal 2: Bootloader independent

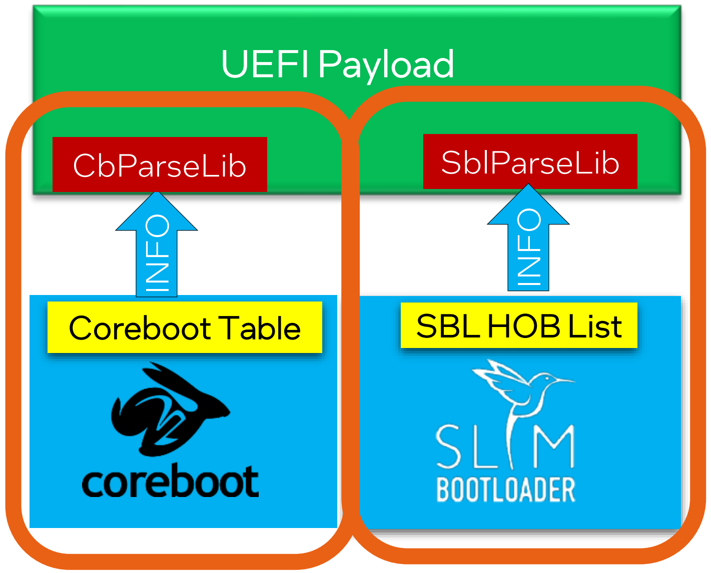


Fig1. Current status

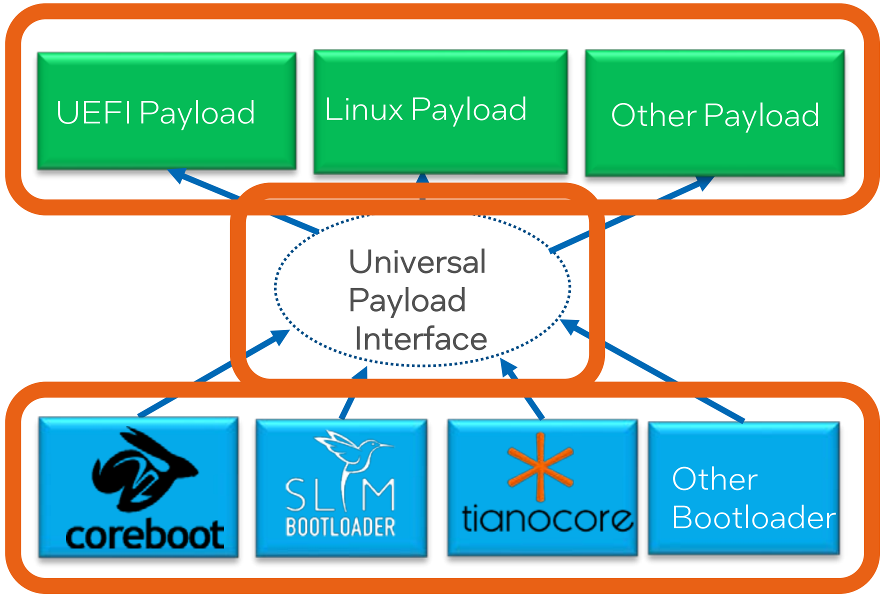


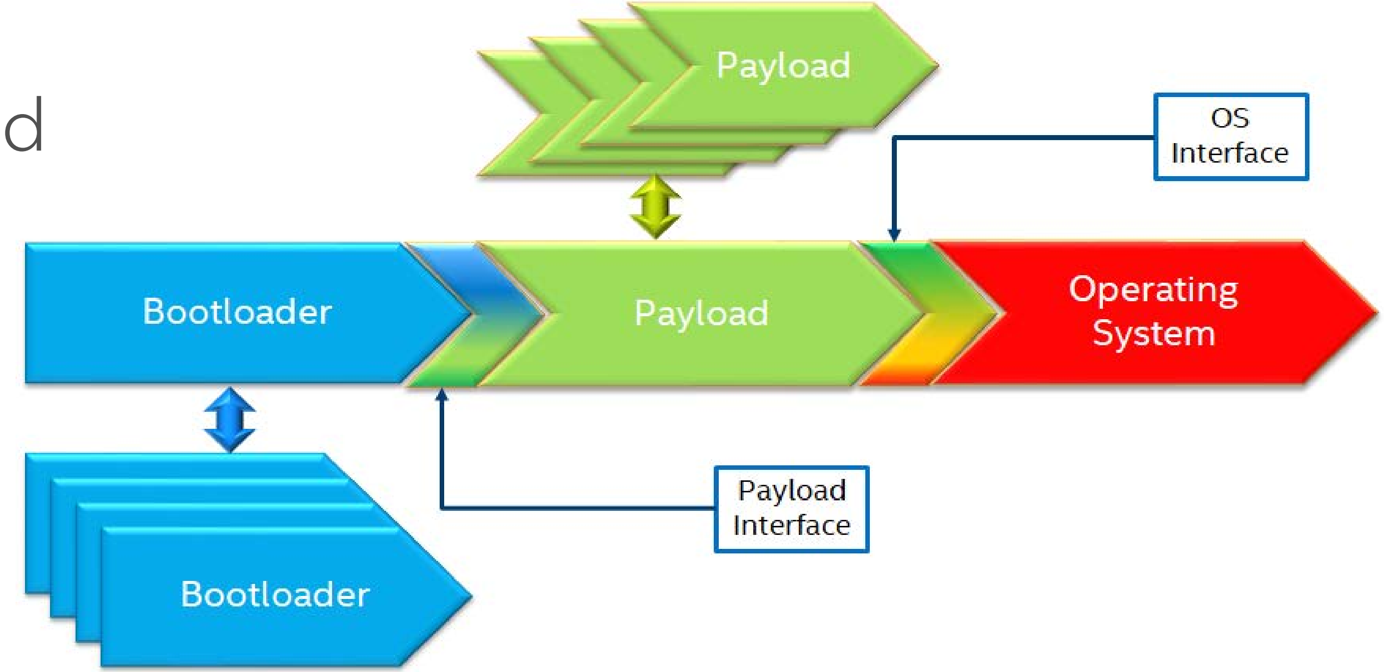
Fig2. Expected goal

Universal payload is bootloader independent

Universal Payload Interface – State of Silicon

Bootloader Transfers Control to the Payload

- Memory controller is initialized
- Processors are patched w/ Microcode update
- PCI Bus is enumerated
- Graphics controller is initialized



Universal Payload Image Format Sections

- Universal Payload Information Section
  - Have section name defined as ".upld_info"
  - Have section aligned at 4-byte boundary within the ELF image.
  - Contain `UNIVERSAL_PAYLOAD_INFO` structure in its section

Format is using ELF (Executable and Linkable Format)

1. Universal Payload Information Section
2. Universal Payload Loaded Image Section
3. Optional universal payload extra image sections with unique section name ".upld.*"

Universal Payload Entry Point

- The prototype of payload entry point

```
typedef
void
(*PAYLOAD_ENTRY) (
   EFI_HOB_HANDOFF_INFO_TABLE              *HobList
   );
```

- The Hand-Off Block (HOB) is passed to the payload entry

Universal Payload Hand-Off

Payload Hand-Off State

- Memory and silicon initialized
- PCI enumeration complete
- Stack
- Interrupt
- Registers
- Page table
- …

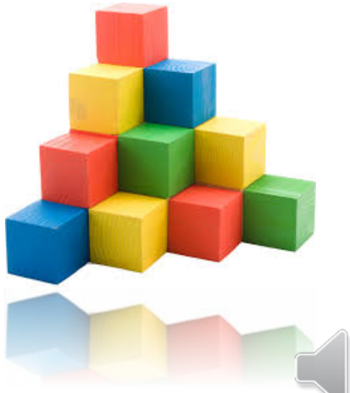
Payload Hand-Off Block (HOB) List

- HOBs in the hoblist
  - ACPI Table HOB
  - SMBIOS Table HOB
  - Device Tree HOB
  - Resource Descriptor HOB
  - Graphics Information HOB
  - Serial Information HOB
  - Cpu Information HOB
  - Would add more HOBs for advanced features.

Hand-Off Block (HOB) List

Required HOB Type	Usage
Phase Handoff Information Table (PHIT) HOB	This HOB is required.
One or more Resource Descriptor HOB(s) describing physical system memory	The DXE Foundation will use this physical system memory for DXE.
Boot-strap processor (BSP) Stack HOB	The DXE Foundation needs to know the current stack location so that it can move it if necessary, based upon its desired memory address map. This HOB will be of type EfiConventionalMemory
One or more Resource Descriptor HOB(s) describing firmware devices	The DXE Foundation will place this into the GCD.
One or more Firmware Volume HOB(s)	The DXE Foundation needs this information to begin loading other drivers in the platform.
A Memory Allocation Module HOB	This HOB tells the DXE Foundation where it is when allocating memory into the initial system address map.

HOBs - Mechanism to discover the state of the system



Universal Payload Info Structure

```
typedef struct {
  UINT32    Identifier;
  UINT32    HeaderLength;
  UINT16    SpecRevision;
  UINT8     Reserved[2];
  UINT32    Revision;
  UINT32    Attribute;
  UINT32    Capability;
  CHAR8     ProducerId[16];
  CHAR8     ImageId[16];
} UNIVERSAL_PAYLOAD_INFO_HEADER;
```

Byte Offset	Size in Bytes	Field	Description
0	4	Identifier	'PLDH' Identifier for the universal payload info.
4	4	HeaderLength	Length of the structure in bytes.
8	2	SpecRevision	Indicates compliance with a revision of this specification in the BCD format. 7 : 0 - Minor Version 15 : 8 - Major Version For revision v0.75 the value will be 0x0075.
12	4	Revision	Revision of the Payload binary. Major.Minor .Revision.Build The ImageRevision can be decoded as follows: 7 : 0 - Build Number 15 :8 - Revision 23 :16 - Minor Version 31 :24 - Major Version
16	4	Attribute	Bit-field attribute indicator of the payload image. BIT 0: Build Type. 0: Release Build 1: Debug Build
20	4	Capability	Bit-field capability indicator that the payload image can support. BIT 0: Support SMM rebase
24	16	ProducerId	A null-terminated OEM-supplied string that identifies the payload producer.
40	16	ImageId	A null-terminated ASCII string that identifies the payload name.

Payload Execution Environment Intel@ 64 and IA-32 Architectures

- Executes on Bootstrap Processor (BSP)
- 32bit protected or 64bit long-mode
- Registers
  - HOBS Pointer in Registers:
    - ESP+4 for 32bit
    - RCX for 64bit
  - EFLAGS – Direction Flag clear
  - Floating Point Control = 0x027F
  - MMX control word = 0x1f80
    - All exceptions masked
  - CR0.EM is clear
  - CR0.TS is clear

- Interrupts disabled
- Page Table
  - Selectors set to flat
  - 32bit may have paging mode
  - 64bit Paging mode enabled
  - All memory space is identity mapped
- Stack – 4KB for payload
  - Payload may use its own stack
- Application Processors (AP)– in halt state

Universal Payloads & Bootloader Payload Interfaces

tianocore

UEFI / EDK II Payload

- Provides UEFI Architectural Protocols
- Uses UEFI HOB
- POC: https://github.com/universalpayload/edk2/tree/universal_payload

SLIM BOOTLOADER

Slim Bootloader OS Loader

- Supports Linux Boot Protocol, ELF, PE and Multi-Boot
- uses UEFI HOB
- POC: https://github.com/universalpayload/slimbootloader/tree/universal_payload

LinuxBoot

Linux Payload

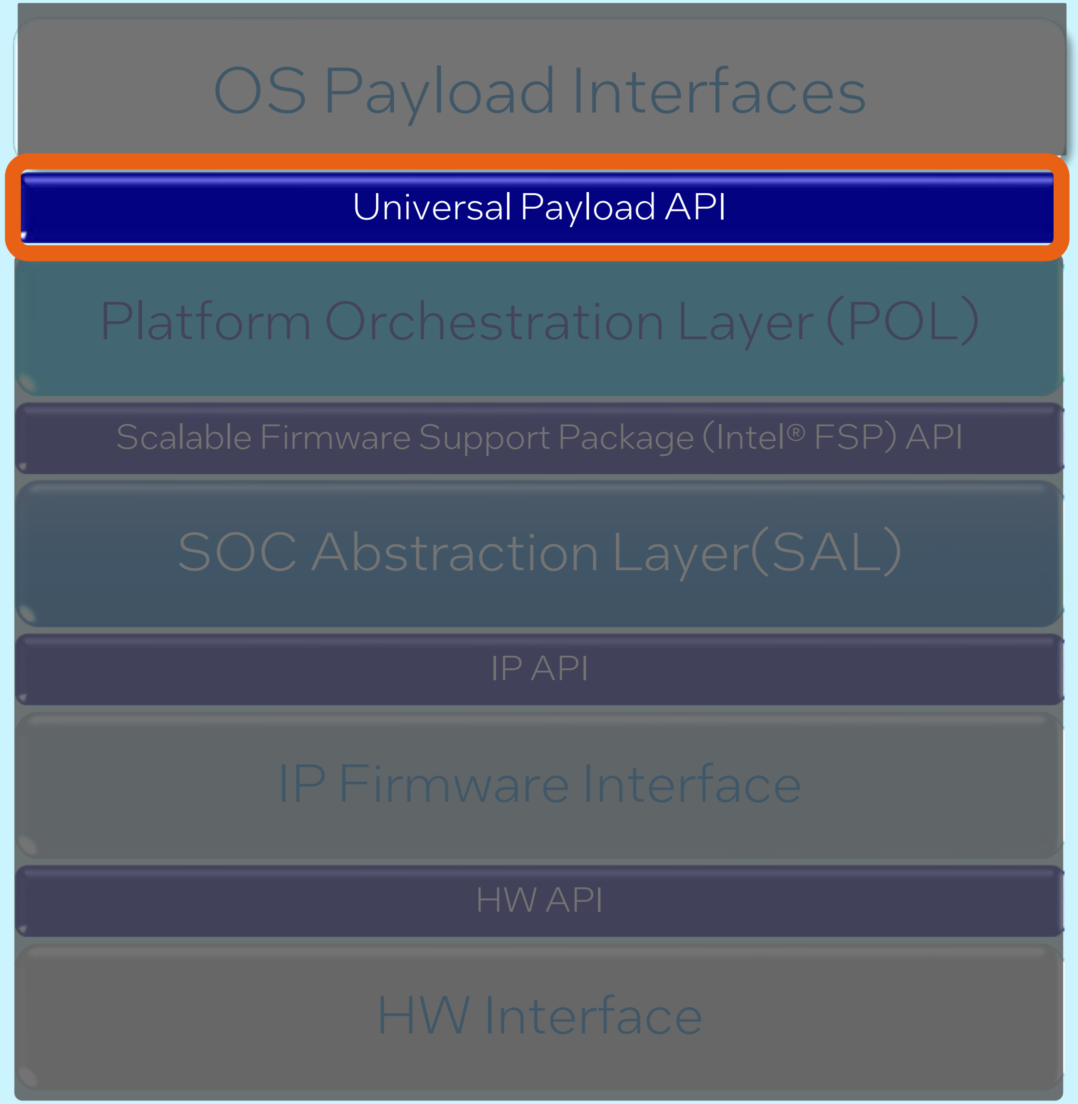
- Like UEFI DXE with Linux kernel https://www.linuxboot.org
- POC: https://github.com/universalpayload/linuxpayload

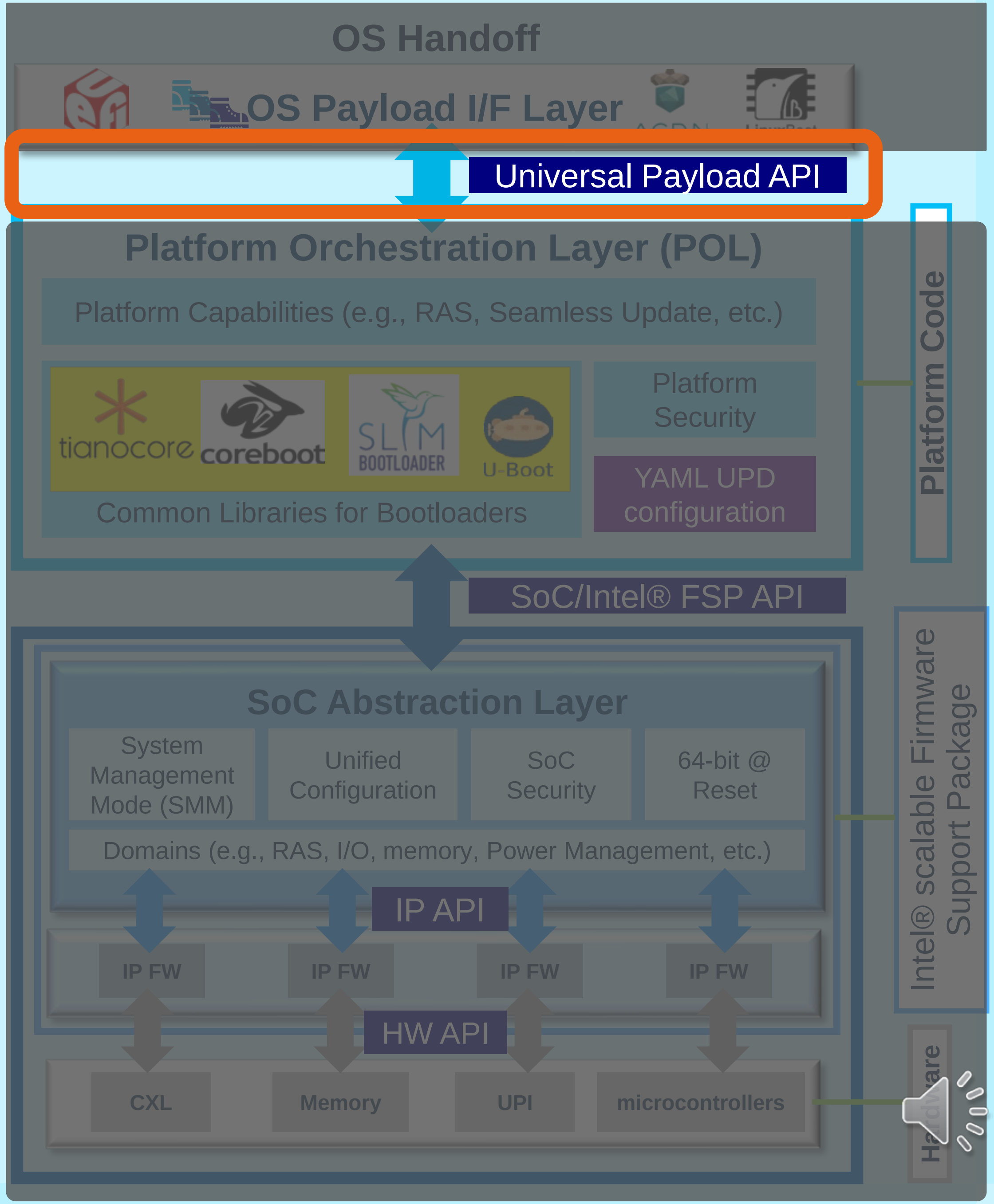
coreboot

Coreboot

- Coreboot Tables – comparable to ACPI RSDT or MP Tables
- Similar to UEFI HOB
- POC: https://github.com/universalpayload/coreboot/tree/universal_payload

- Universal payload specification is open sourced
  Current version 0.75
  https://github.com/universalpayload/documentation
- Specification HTML version
  https://universalpayload.github.io/documentation/
- Tools
  https://github.com/universalpayload/tools


OS Payload Interfaces
Universal Payload API
Platform Orchestration Layer (POL)
Scalable Firmware Support Package (Intel® FSP) API
SOC Abstraction Layer(SAL)
IP API
IP Firmware Interface
HW API
HW Interface


OS Handoff
OS Payload I/F Layer
Universal Payload API
Platform Orchestration Layer (POL)
Platform Capabilities (e.g., RAS, Seamless Update, etc.)
tianocore
coreboot
SLIM BOOTLOADER
U-Boot
Common Libraries for Bootloaders
Platform Security
YAML UPD configuration
Platform Code
SoC/Intel® FSP API
SoC Abstraction Layer
System Management Mode (SMM)
Unified Configuration
SoC Security
64-bit @ Reset
Domains (e.g., RAS, I/O, memory, Power Management, etc.)
IP API
IP FW
IP FW
IP FW
IP FW
HW API
CXL
Memory
UPI
microcontrollers
Intel® scalable Firmware Support Package

intel®