

UEFI Tutorial

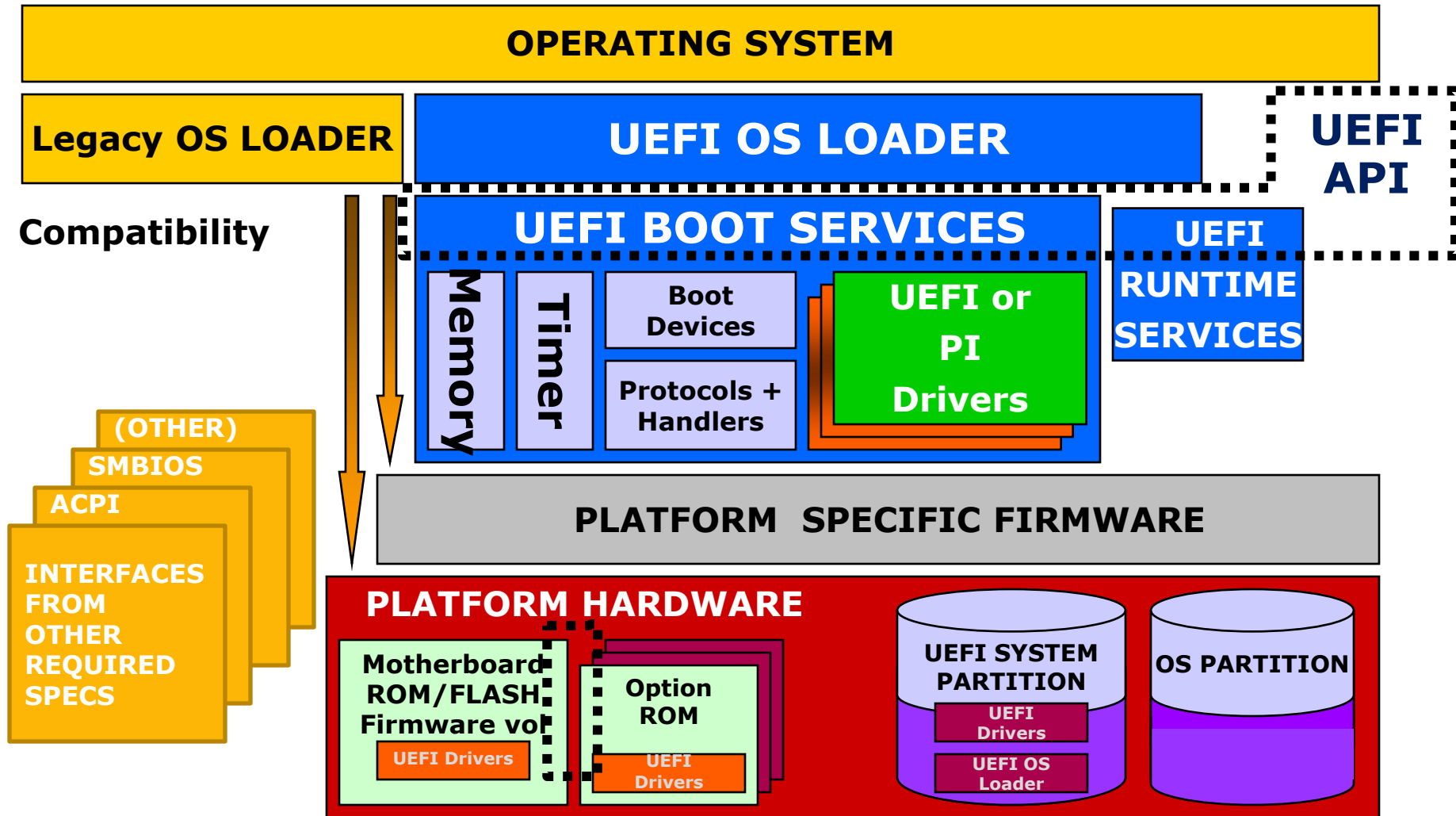
8/29/12

Harry Hsiung
Intel Corp

Agenda

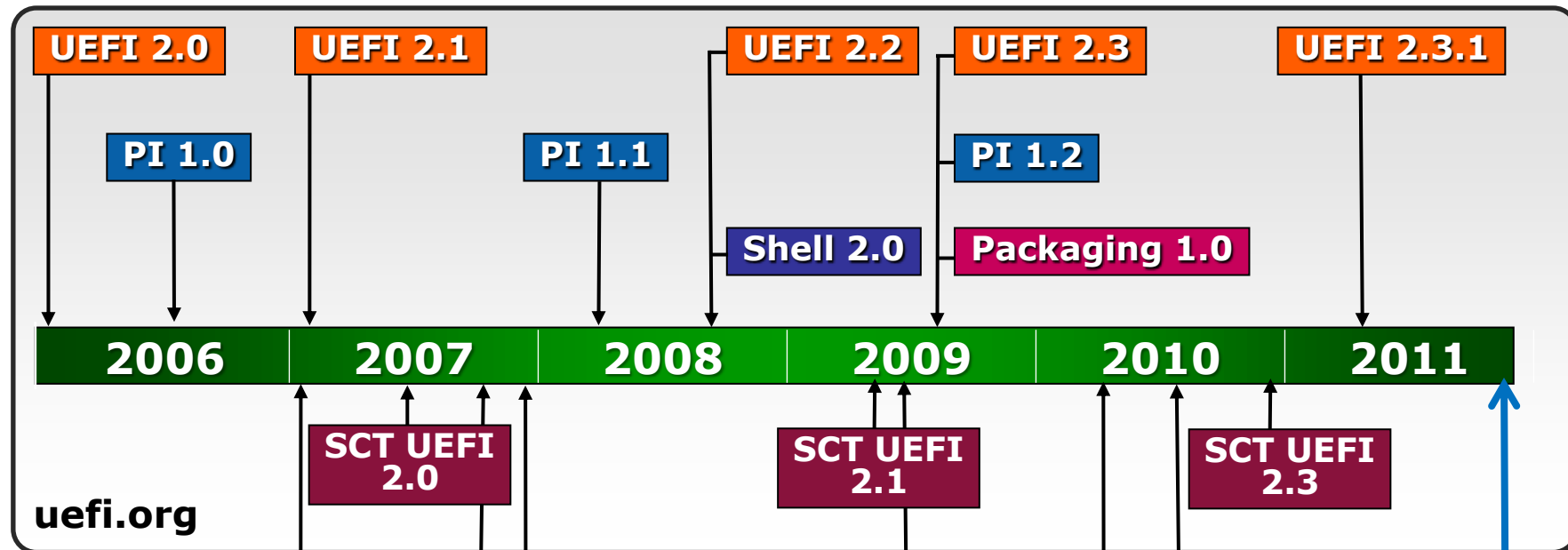
- UEFI Basics
- UEFI security features
- UEFI development platforms
- UEFI Resources
- Opens
- Backup

UEFI architecture

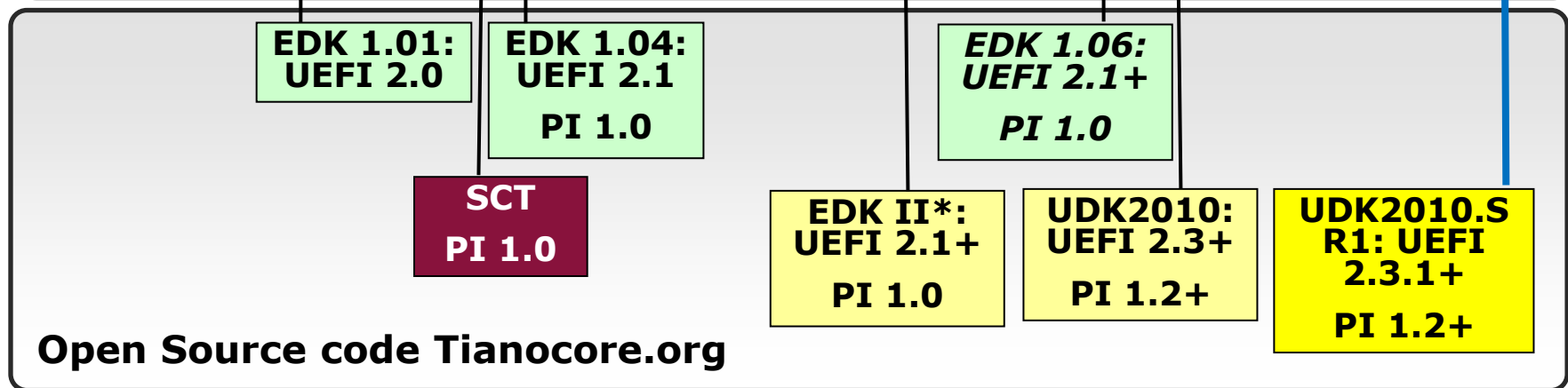


UEFI Specification Timeline

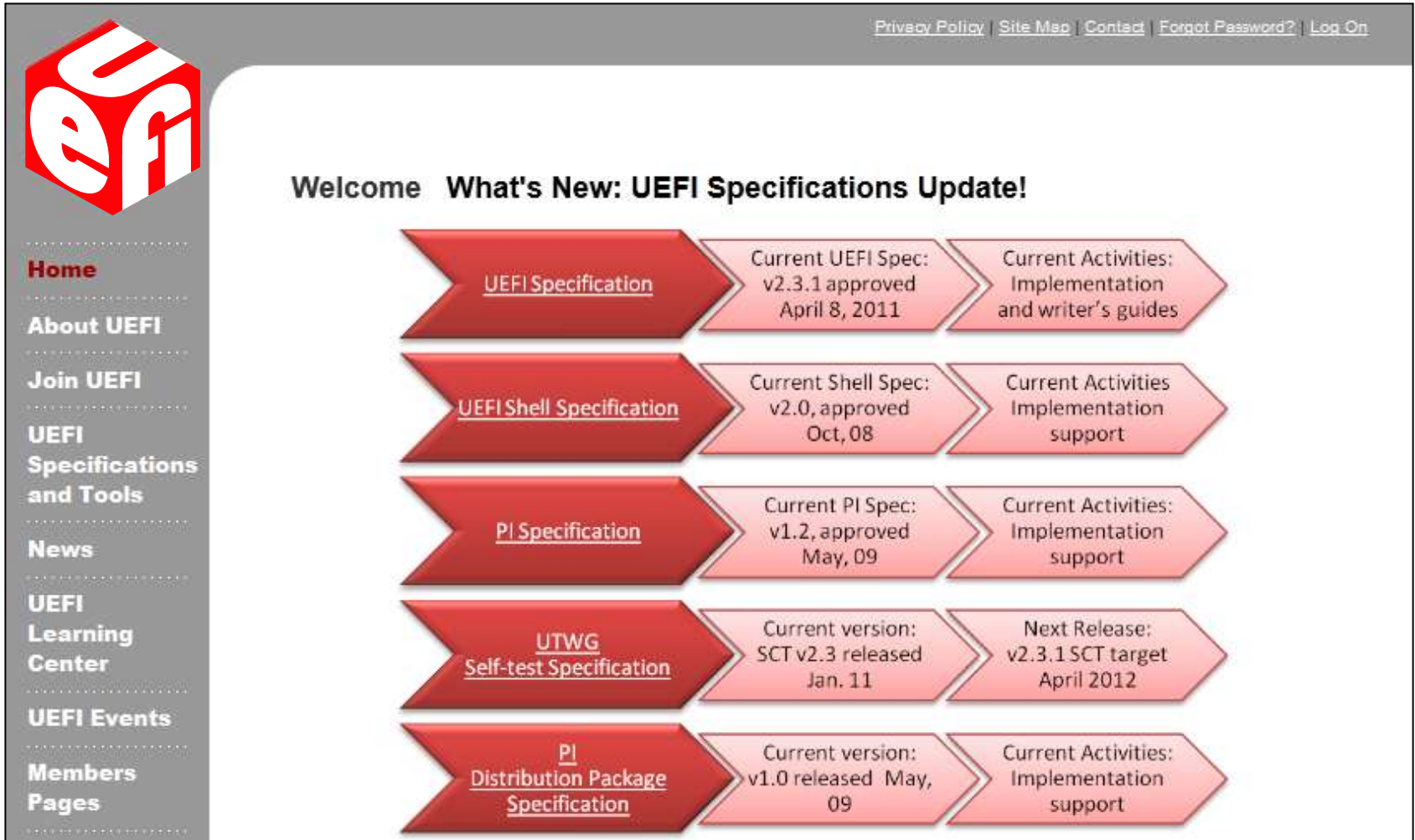
Specifications



Implementation



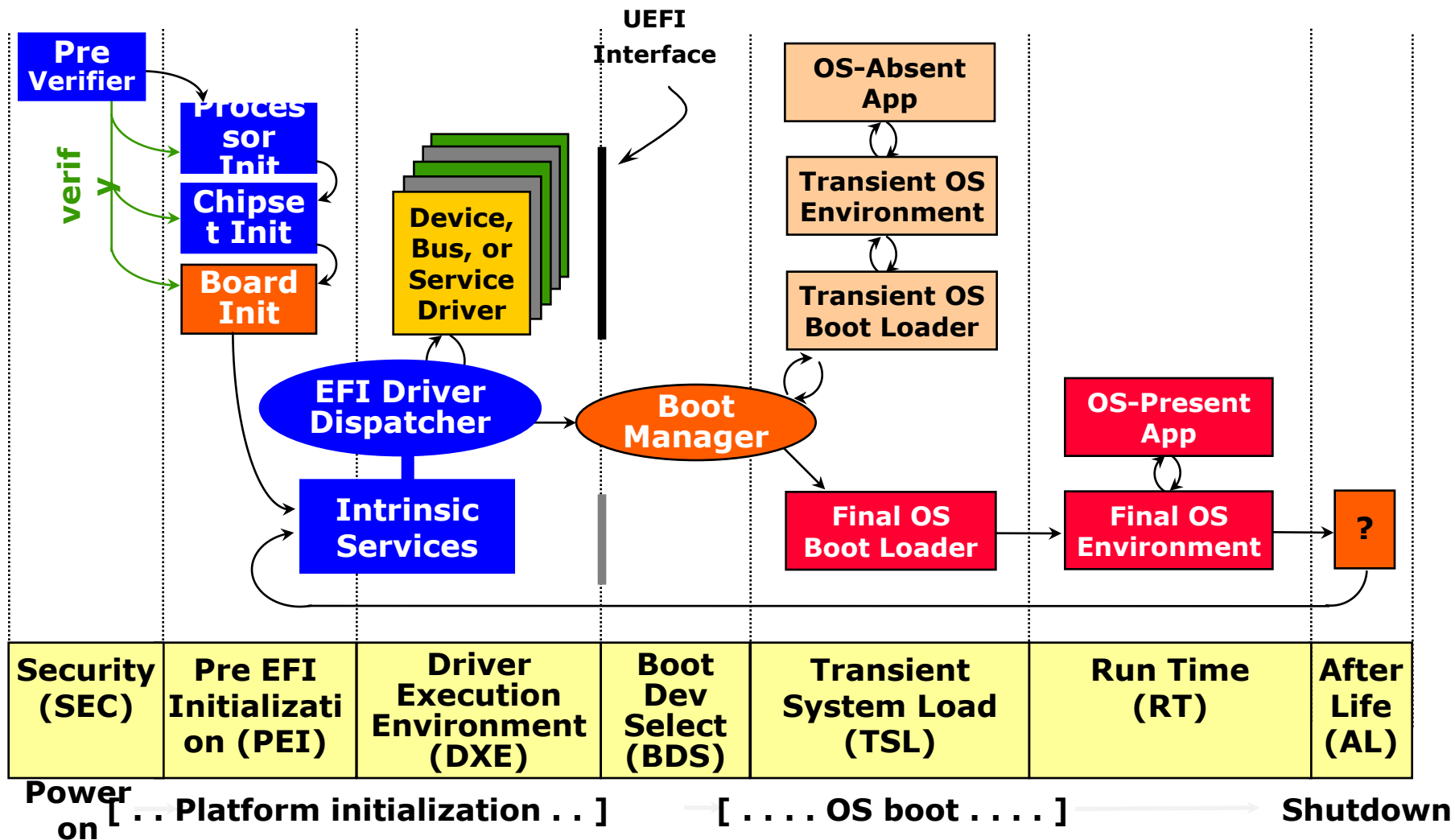
* EDK II is same code base as UDK2010 * EDK I is UEFI 2.0..UEFI 2.1(1117)



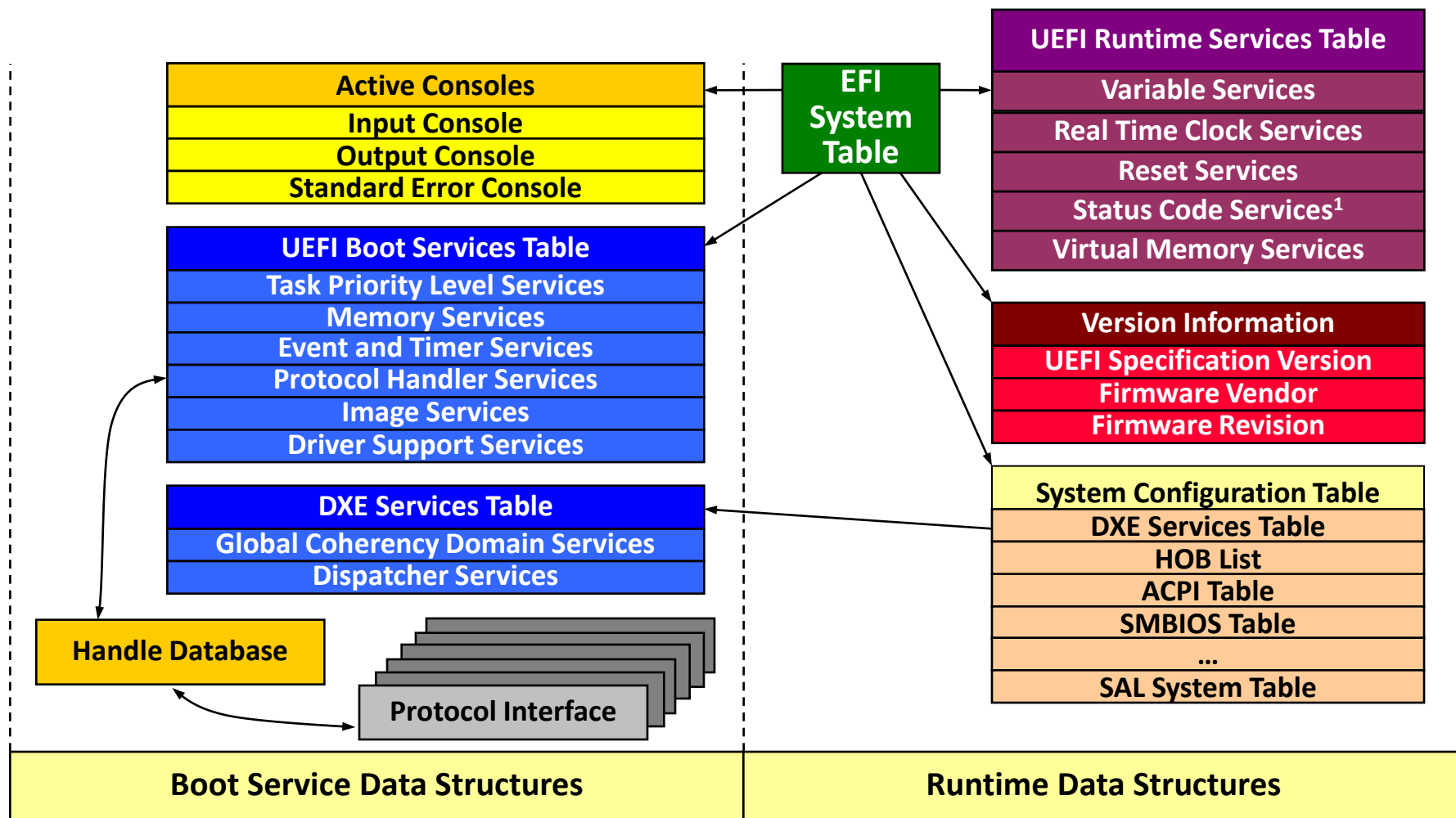
The screenshot shows the uefi.org website. At the top right, there are links for [Privacy Policy](#), [Site Map](#), [Contact](#), [Forgot Password?](#), and [Log On](#). On the left side, there is a vertical navigation menu with the following items: **Home**, **About UEFI**, **Join UEFI**, **UEFI Specifications and Tools**, **News**, **UEFI Learning Center**, **UEFI Events**, **Members**, and **Pages**. The main content area is titled "Welcome What's New: UEFI Specifications Update!". It features five rows of information, each with a red arrow pointing right containing a specification name, followed by two light red boxes containing version and activity details.

Specification	Current Version / Status	Current Activities
UEFI Specification	Current UEFI Spec: v2.3.1 approved April 8, 2011	Current Activities: Implementation and writer's guides
UEFI Shell Specification	Current Shell Spec: v2.0, approved Oct, 08	Current Activities: Implementation support
PI Specification	Current PI Spec: v1.2, approved May, 09	Current Activities: Implementation support
UTWG Self-test Specification	Current version: SCT v2.3 released Jan. 11	Next Release: v2.3.1 SCT target April 2012
PI Distribution Package Specification	Current version: v1.0 released May, 09	Current Activities: Implementation support

Architecture Execution Flow



DXE Foundation Data Structures



Boot Service Data Structures

Runtime Data Structures

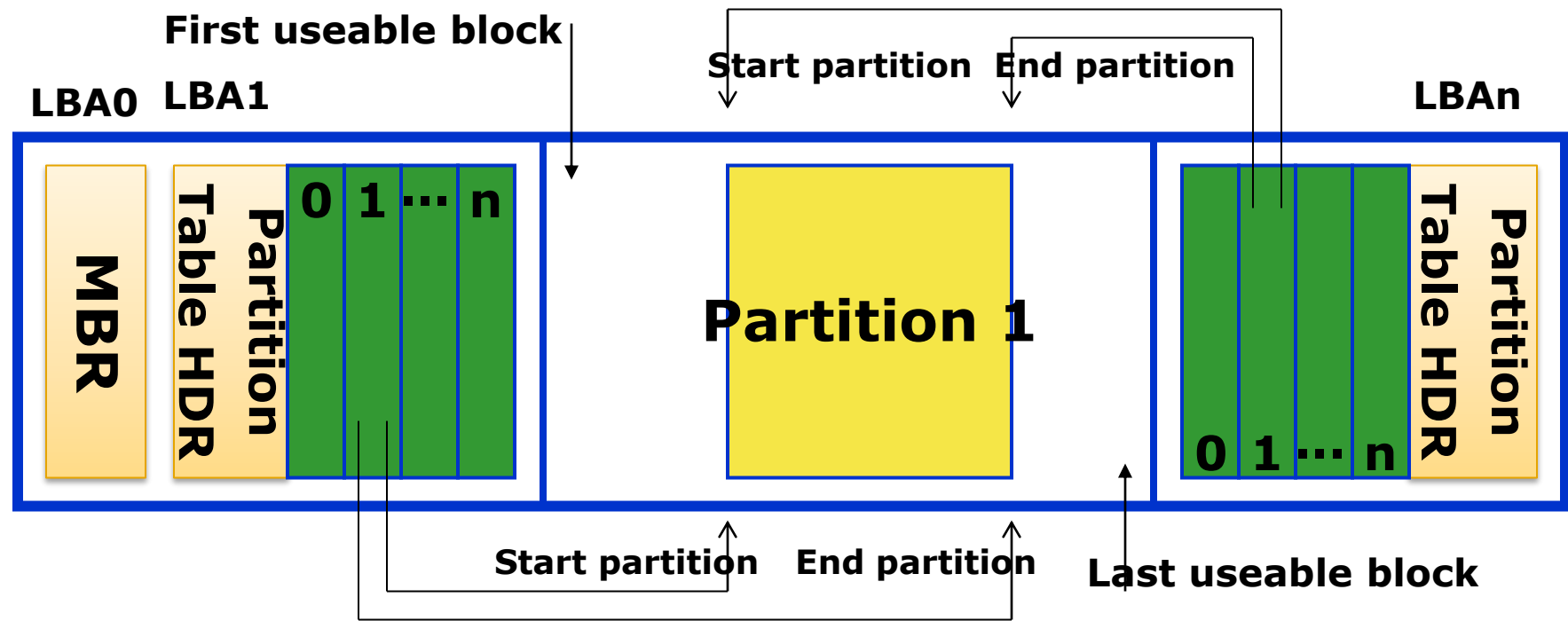
Boot Support – Device Types

- Hard disk
- Raid
- Fiber channel
- Removable media
 - CD-ROM, DVD-ROM
 - El Torito 1.0 “No emulation”
 - Floppy, USB Storage, etc.
- Network
 - PXE BIOS support specification (Wire for Management)
 - iSCSI
- Future media via extensibility methods

Full Device Support

**Expanded Capabilities
Versus Legacy BIOS**

GPT - New Partition Structure



Primary Partition Table

Backup Partition Table

>2.2 Tb support , Unique GUID signature support in partion table HDR
See Section 5 of the UEFI 2.3.1 Spec.

GPT Advantages over MBR Partition Table

- 64-bit LBA
 - No more 2.2TB limit
 - Up to 9.8 zettabytes
- Improved partitioning
 - Supports unlimited number of partitions
 - Uses a primary and backup table for redundancy
 - Defines a GUID for identifying each partition
 - Uses GUID & attributes to define partition type
- Uses version number and size fields for future expansion.
- Uses CRC32 fields for improved data integrity
- Each partition contains a 36 Unicode character human readable name.
- No MBR problems
 - No “magic code” must execute as part of booting

UEFI Specification - Key Concepts

- Objects - manage system state, including I/O devices, memory, and events
- UEFI System Table - data structure with data information tables to interface with the systems
- Handle Database and Protocols - callable interfaces that are registered
- UEFI Images - the executable content format
- Events - the software can be signaled in response to some other activity
- Device Paths - a data structure that describes the hardware location of an entity

GUID

- Globally Unique Identifier
 - 128-bit quantity defined by Wired for Management (WfM) 2.0 specification **
- Used to identify protocols
 - 1:1 with interfaces
- Regulate extension mechanism
 - Documented in the spec
 - Added through drivers

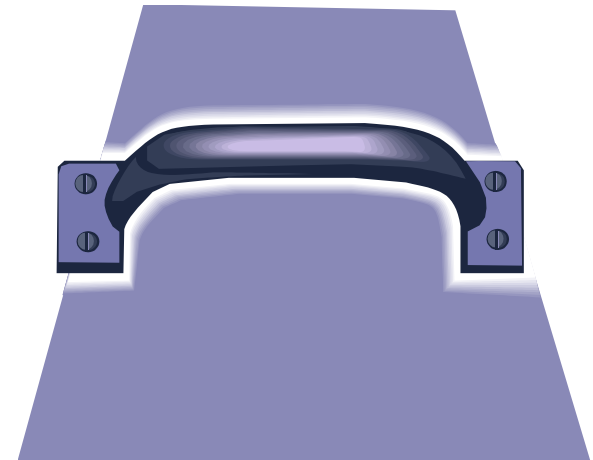


Safe co-existence of 3rd party extensions

** <http://www.intel.com/design/archives/wfm/index.htm>

Handles

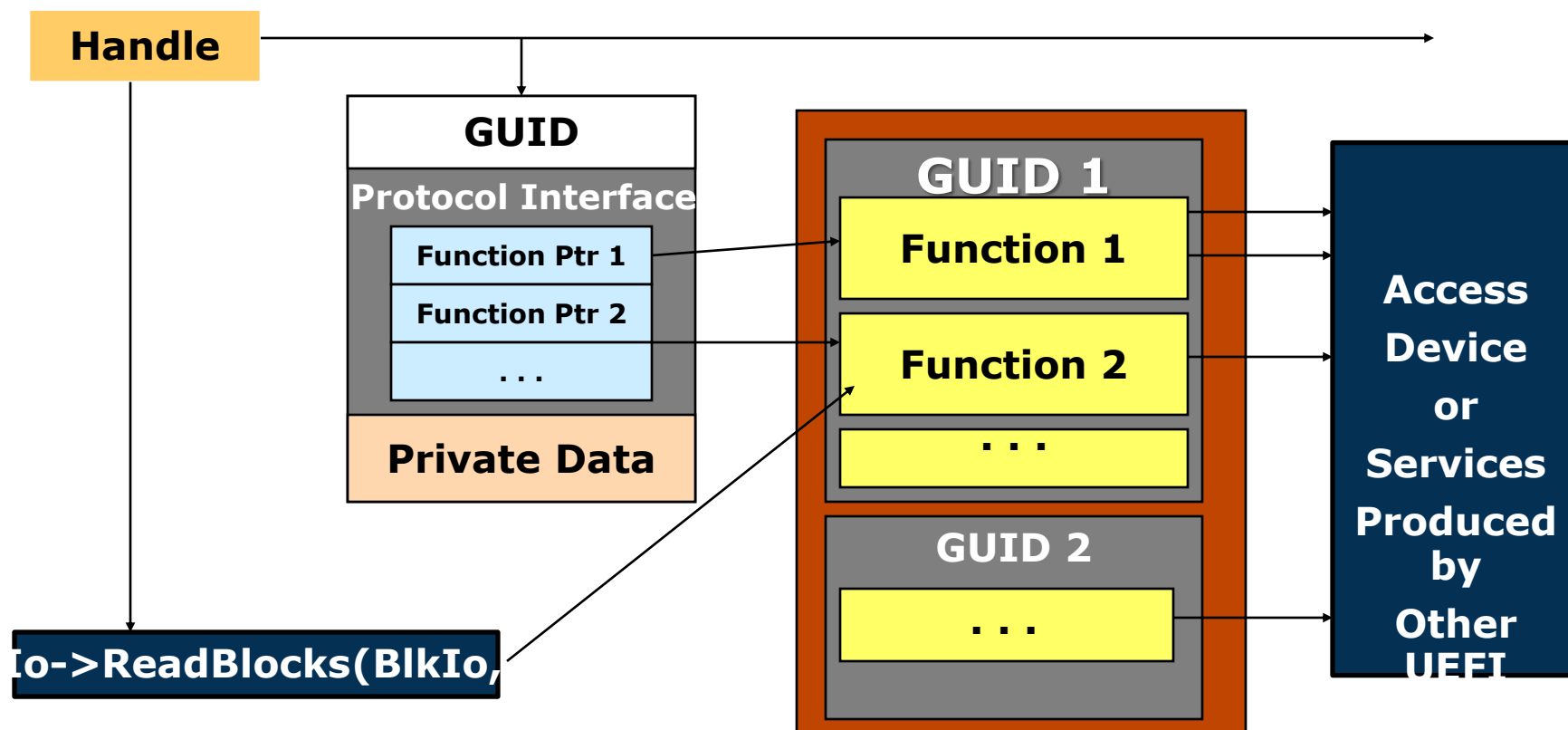
- All protocols have an associated handle
- Every device and executable image in UEFI has a handle protocol in the handle database
- Every boot device must have a device path protocol to describe it



Protocols (API)

▪ GUID, Interface Structure, Services

- DEVICE_PATH, DEVICE_IO, BLOCK_IO, DISK_IO, FILE_SYSTEM, SIMPLE_INPUT, SIMPLE-TEXT_OUTPUT, SERIAL_IO, PXE_BC, SIMPLE_NETWORK, LOAD_FILE, UNICODE_COLLATION





Device Path Protocol

- A data structure description of where a device is in the platform
- All boot devices, logical devices and images must be described by a UEFI device path
- The UEFI Specification defines six types of device paths

Six Types of Device Path Types

- Hardware – where is the device in the system
- ACPI – UID/HID of device in AML
- Messaging – Classifies device as LAN, Fiber Channel, ATAPI, SCSI, USB, ...
- Media – i.e. Hard Drive, Floppy or CD-ROM
- BIOS Boot Specification – used to point to boot legacy operating systems
- End of hardware – marks end of device path

Device Path Examples

`Acpi (PNP0A03, 01) / Pci (1F|1) / Ata (Primary, Master) / HD (Part3, Sig001100112)`

`Acpi (PNP0A03, 1) / Pci (1E/0) / Pci (0|0) / Mac (0002 B3647D69)`

`Acpi (PNP0A03, 0) / Pci (1F|0) / Acpi (PNP0501, 0) / UART (115200 81)`

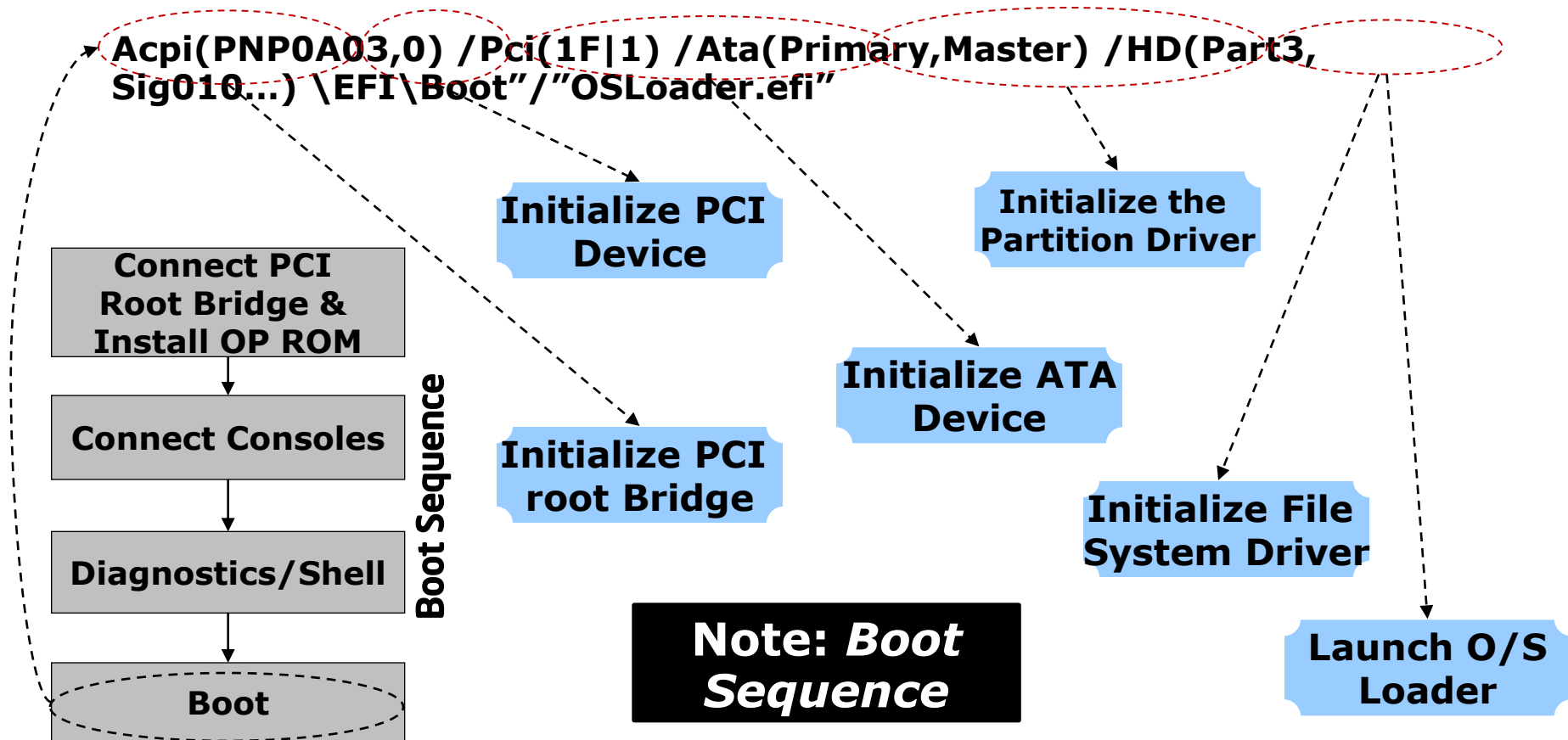
See § 9 UEFI 2.X Spec.

¹ ACPI Name space - contain HID, CID, and UID fields that match the HID, CID, and UID values that are present in the platform's ACPI tables

²Truncated to fit on slide, GUIDs are 128 bits

Why UEFI Device Path?

- The UEFI Device Path describes a boot target
 - Binary description of the physical location of a specific target



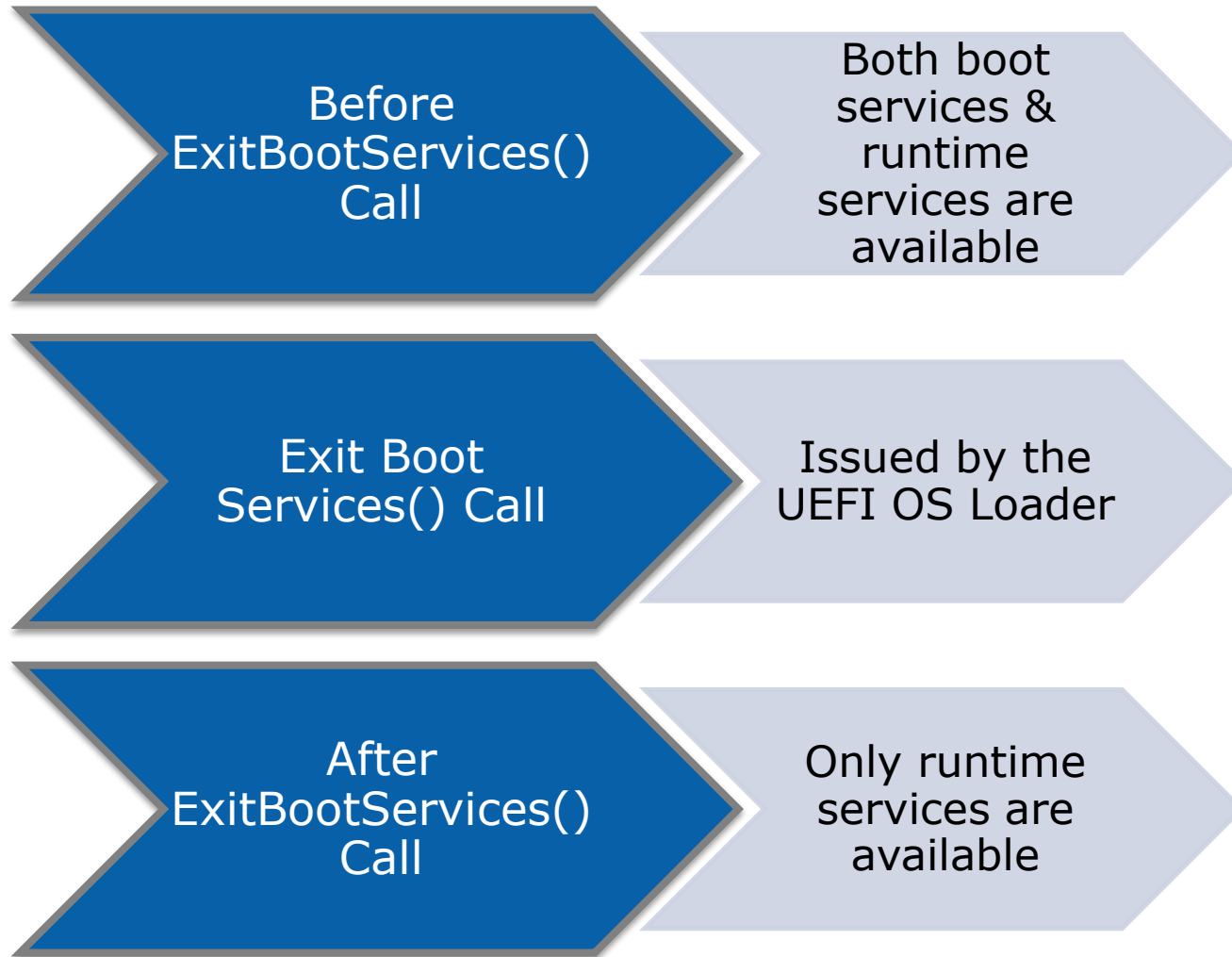
Boot Services

- *Full set of firmware services for pre-boot*
- Events and notifications
 - Polled devices, no interrupts
- Watchdog timer
 - Elegant recovery
- Memory allocation
- Handle location – for finding protocols
- Image loading
 - For drivers, applications & the OS loader

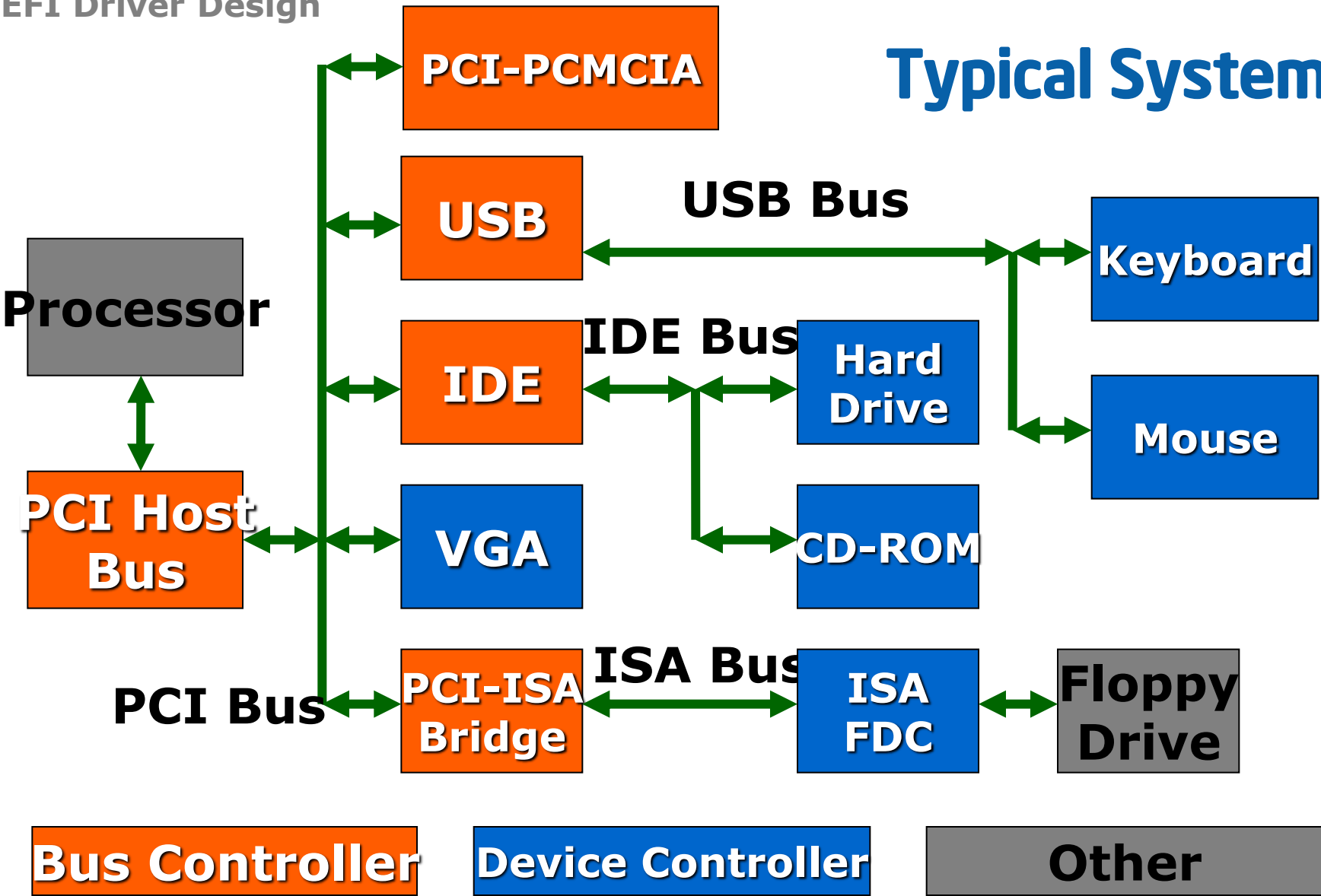
Runtime Services

- *Minimal set of services for the UEFI Aware OS*
- *Available in boot services and at OS runtime*
- Timer, Wakeup Alarm
 - Allows system to wake up or power on at a set time
- Variables
 - Boot manager handshake
- System reset

ExitBootServices()



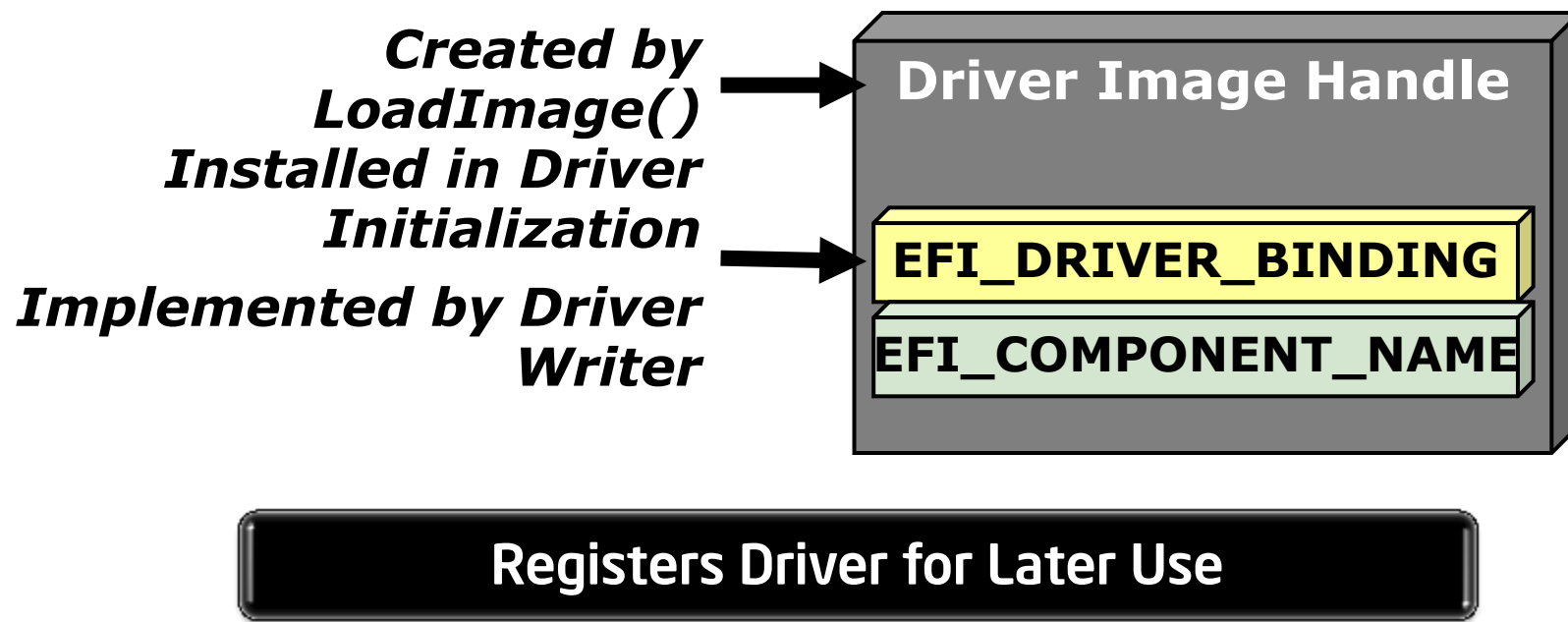
Typical System



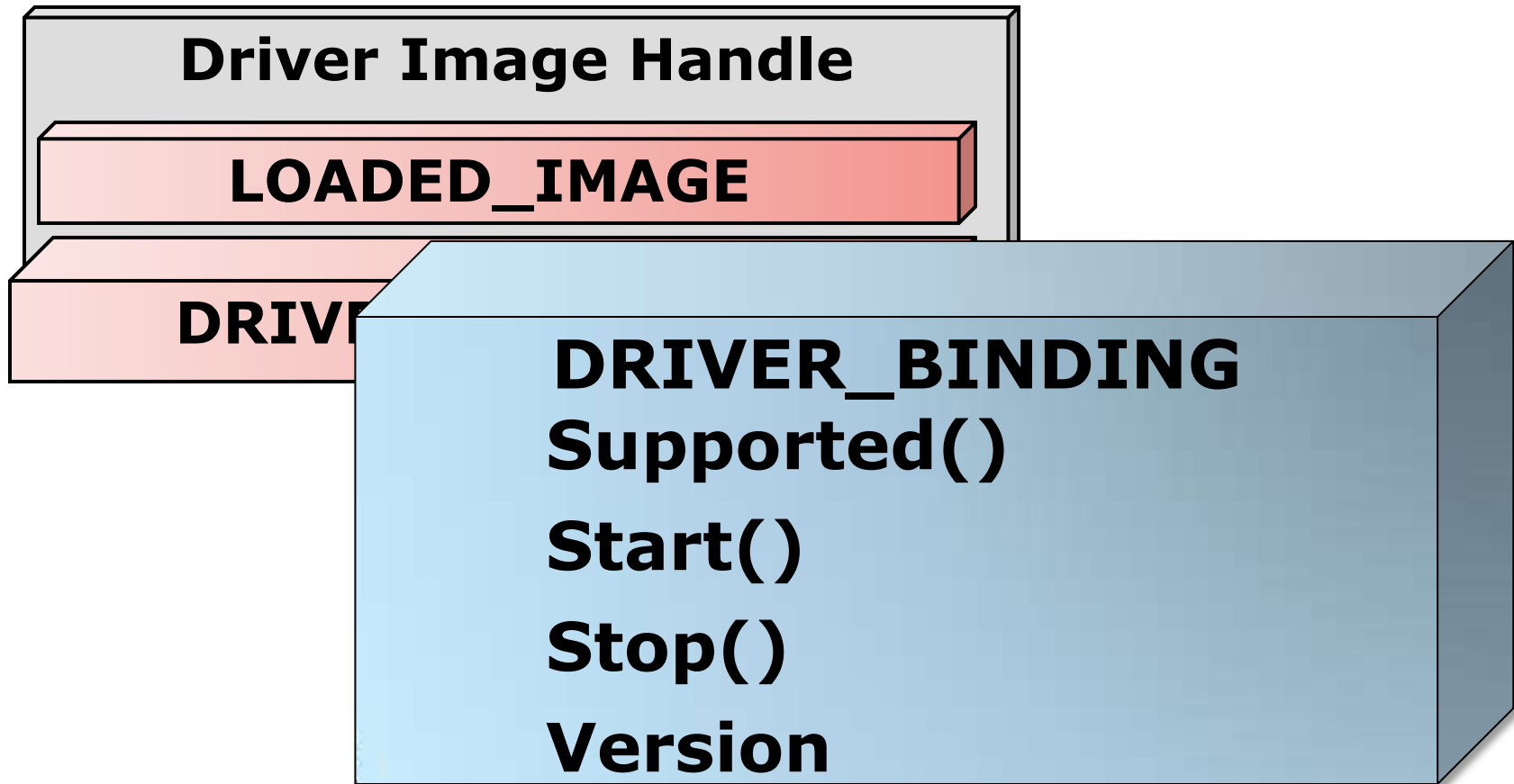
See § 2.5 UEFI 2.X Spec.

Driver Initialization

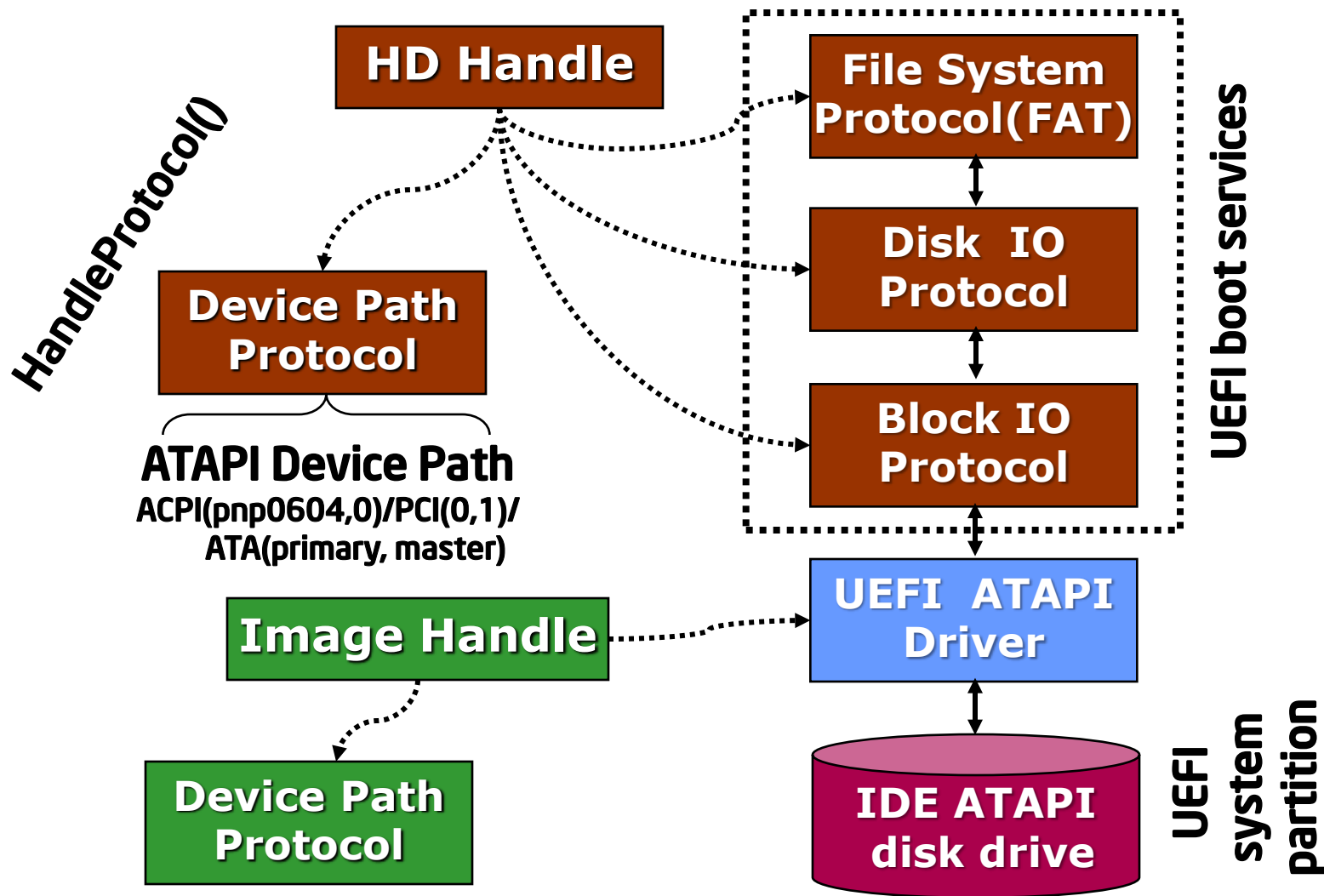
- UEFI Driver Handoff State
- Not Allowed to Touch Hardware Resources
- Installs Driver Binding on Driver Image Handle



Driver Binding Protocol



Example: UEFI ATAPI Driver Stack



UEFI 2.1 Features

- Added protocols
 - HII (several protocols)
 - Absolute pointer protocol
- New member functions or equivalent
 - Driver Supported Version (for option roms)
 - Extended Simple Text In (more function keys supported)
 - Authenticated Variables
 - Extended SCSI Pass through
 - Signal on configuration change
 - EHCI exclusive ownership
 - Firmware storage device path
 - Hot key registration support
 - Run-time services with interrupts enabled
- Clean-up: e.g. error returns, * vs ** in declarations in several protocols introduced as a result of implementation

Required	Optional
----------	----------

UEFI 2.2 Significant Features

- Networking – IPv6
 - IPv6 stack corresponding to existing IPv4 stack
 - Replacement for PXE protocols which are IPv6 compliant and large network friendly
 - Now being worked through IETF
 - Support for more LAN protocols: EAP and VLAN
- Security – Driver signing
 - Added optional ability to create firmware / OS trust relationships
 - Via key exchange
 - More signature combinations
 - Good / Bad list support
 - Platform owner control of denial response
 - Pre-Boot Authentication (PBA) Framework
 - Passwords, Smart cards, Fingerprint sensors, etc.

Required	Optional
----------	----------

UEFI 2.2 Other Features

- HII
 - Additional operators for mapping to other standards
 - Page by page security control
 - Animation updates
- EFI_ATA_PASS_THRU Protocol
 - Gives direct access to ATA devices
- UEFI Driver Health
 - Allow for a driver to fix/re-configure (e.g. rebuild RAID set)
- ABI Updates/Clarifications
 - Floating Point/MMX/XMM
 - 16-Byte stack alignment
- EFI_LOAD_FILE2 Protocol
 - Loads non-boot-option EXEs (PCI option ROMs & apps)
 - Modifies LoadImage() behavior
- EFI_LOADED_IMAGE Protocol
 - Associates entire device path with EXE image
- Bug fixes in spec for rest of document

Required	Optional
----------	----------

UEFI 2.3 Features

- Two possible views
 - Special release for ARM binding
 - Fairly quick release for items including ARM binding
- Also includes
 - Boot Services protocol for firmware update
 - Mainly for Option ROMs
 - Bug fixes
- Other items on deck
 - Ubiquitous Firmware Update

Required	Optional
----------	----------

UEFI Forum Updates



- **UEFI Specification**
 - Version 2.3.1, Errata A published on Sept. 7, 2011
 - Clarifications from version 2.3.1
 - Additional ECRs are work in progress
- **UEFI Self Compliance Tests (SCT)**
 - Published a UEFI Winter 2012 Plugfest Release in Feb, 2012
 - Version 2.3.1 compliance test preview
 - Investigating coverage for 2.3.1 Errata A
- **Be Ready for Windows* 8**
 - UEFI 2.3.1 support
 - UEFI drivers and applications
 - Secure boot (sign the executables)
 - Seamless boot, hybrid boot, fast boot
 - IPv6 and IPv4 network stack
 - UEFI Spring 2012 Plugfest in Taipei (May 8-10), Redmond (July 16-20th)
- **PI Specification**
 - Version 1.2 Errata C published in October 2011

2012 marks the ubiquitous adoption of UEFI on PCs

Intel® UDK2010 SR1 (UEFI 2.3.1)

- User Identity (UID) Support (UEFI 2.3.1 a)
- Secure Storage Protocol
 - Enable **Opal/eDrive** SATA devices using the EFI_STORAGE_SECURITY_COMMAND_PROTOCOL, ATA-8 Trusted Send/Receive and IEEE1667 Silo (UEFI 2.3.1 a)
- Networking Improvements
 - Errata related to Netboot6-DUID
 - Provide more DHCP4 & DHCP6 API support
 - iSCSI (ip6) open source implementation for IPv6
- TCG Physical Presence (PP). Based on the Physical Presence Interface Specification Version 1.20, Revision 1.0.
- Support ATA Asynchronous Block Io (UEFI 2.3.1 a)
- USB 3.0 Controller Support (XHCI)
- Update Internal Forms Representation (IFR) implementation to match UEFI 2.3.1 Specification
- Fast boot support (asynchronous blockIO2)

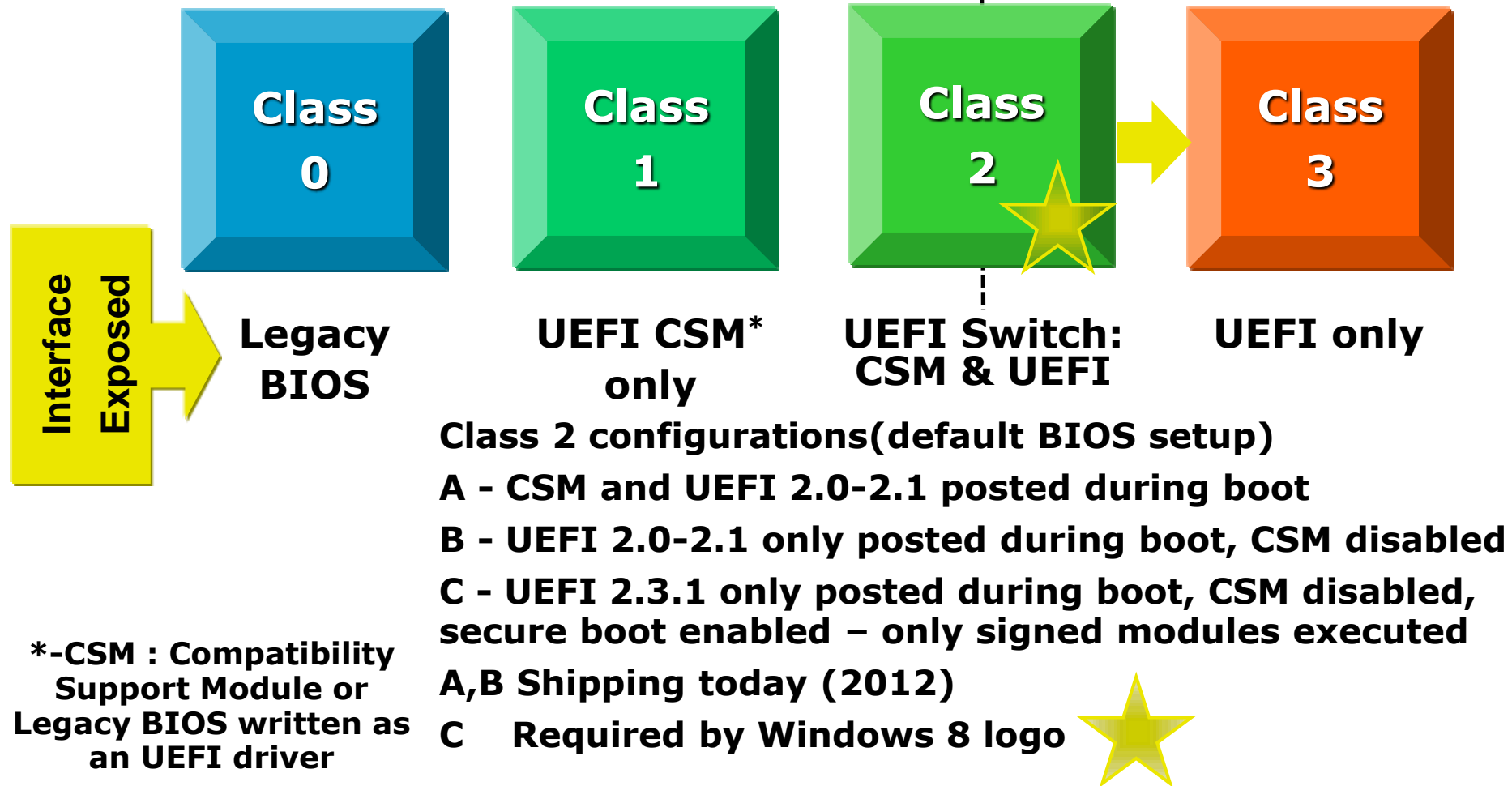


www.intel.com/UDK

Intel® UEFI Development Kit 2010 (Intel® UDK2010)

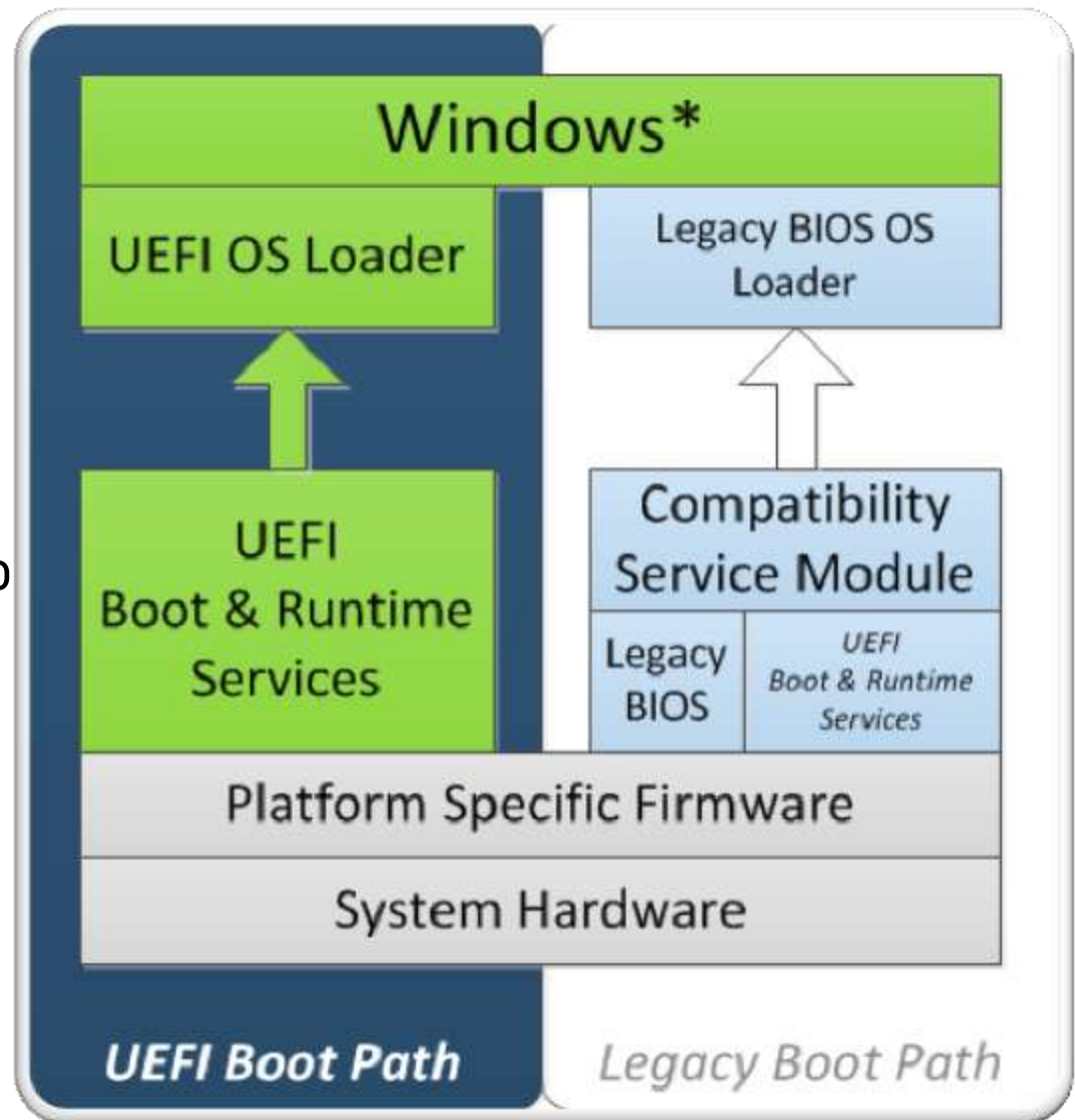
UEFI System Classes

Based on Firmware I/F
Today 2013



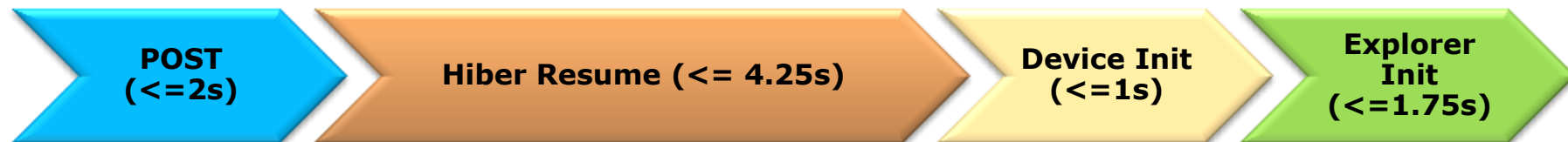
Windows 8 Boot Flow

- Windows 8 installs UEFI OS Loader if UEFI is detected
- Most PCs today boot through CSM path
- For compatibility the CSM boot path available
- Windows 8 logo requirement to boot UEFI only (cannot run csm in client builds)
- Client must boot with UEFI secure boot enabled
- For server if implemented



Windows 8 Certification Requirements - UEFI Boot

Boot Performance Requirements



- Windows 8 aims to support <10s boot, on SSD systems
 - POST: <2s (without TPM; SSD)
 - Resume: <4s (without CSM)
 - Device Init: <2s (varies by quality of driver)
- New WHQL Requirements for hardware design
 - TPM: <300ms init
 - Total time 2.3 seconds max for boot with SSD and UEFI secure boot

What is Security from BIOS Perspective

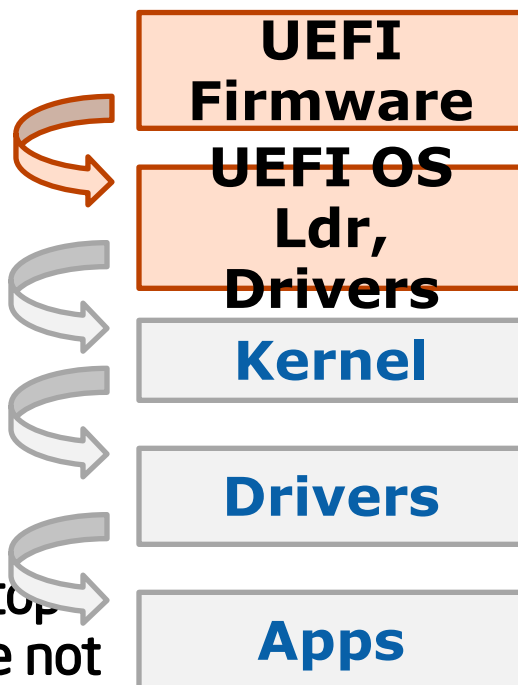
- **Secure Boot - UEFI**
 - Defined a policy for Image loading
 - Cryptographically signed
 - Private key at signing server
 - Public key in platform
- **Measured Boot -Trusted Computing Group (TCG)**
 - Trusted Platform Module (TPM)
 - Isolated storage and execution for Logging changes, attestation
- **NIST 800-147 -Security Guidelines for System BIOS Implementations**

UEFI Secure Boot VS TCG Trusted Boot

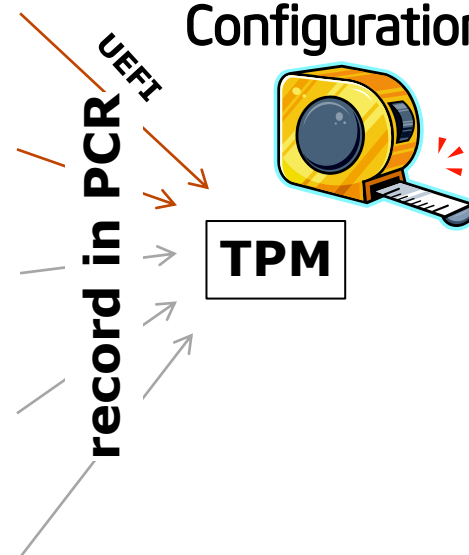
UEFI authenticate OS loader
(pub key and policy)

Check signature of
before loading

- UEFI Secure boot will stop platform boot if signature not valid (OEM to provide remediation capability)
- UEFI will require remediation mechanisms if boot fails

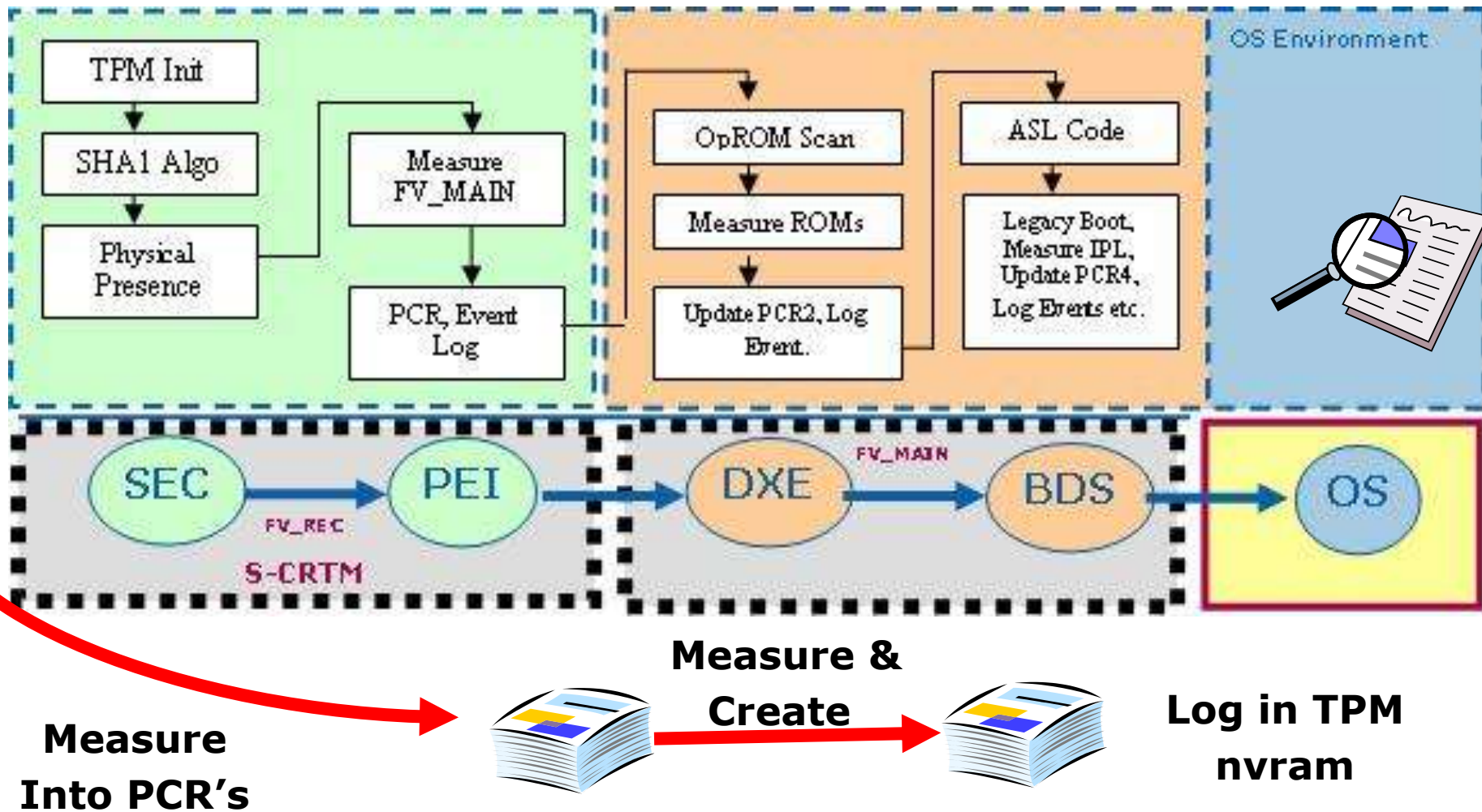


UEFI PI will measure OS loader & UEFI drivers into iTPM PCR (Platform Configuration Register)

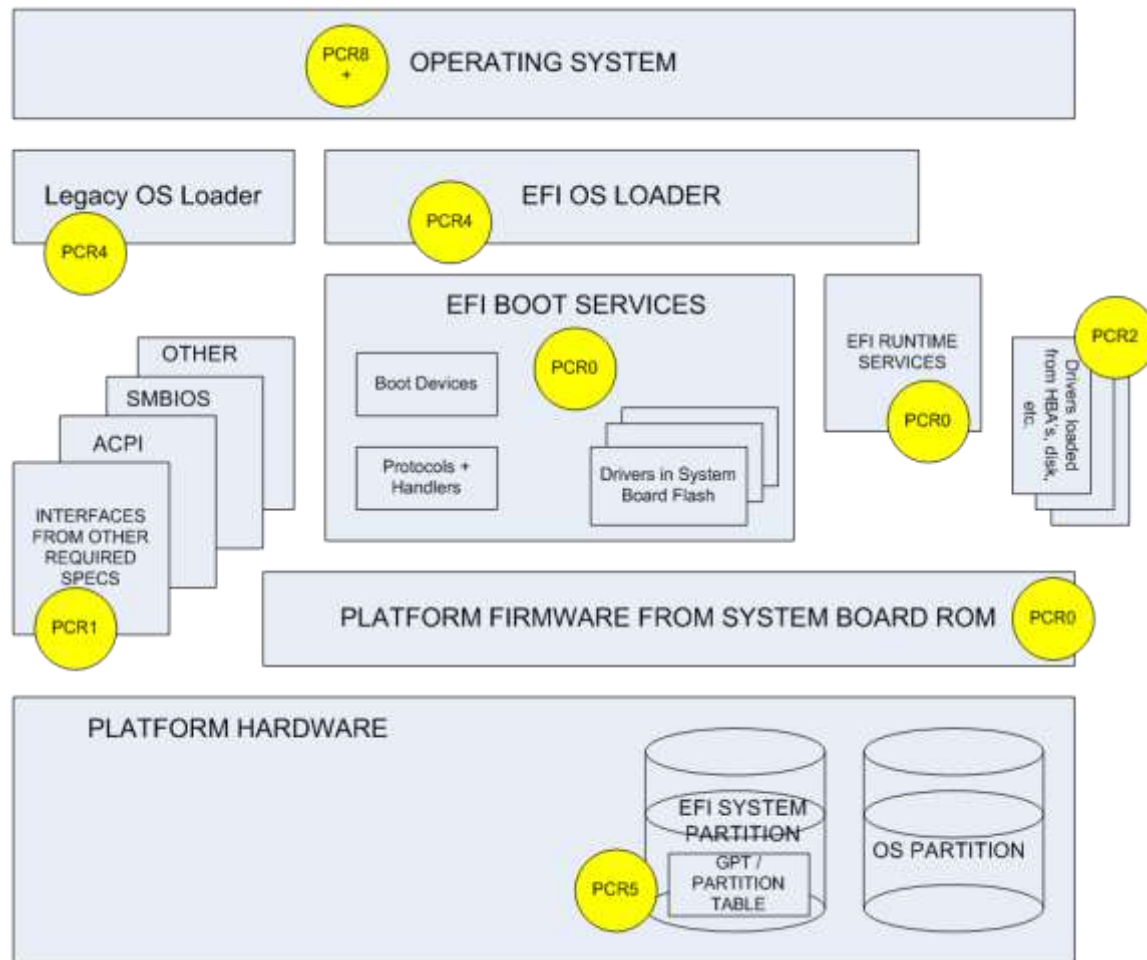


- TCG Trusted boot will never fail
- Incumbent upon other SW to make security decision using attestation

UEFI/PI Architecture Boot Flow – Create/Evaluate Integrity List



Measured items in UEFI in Trusted Boot

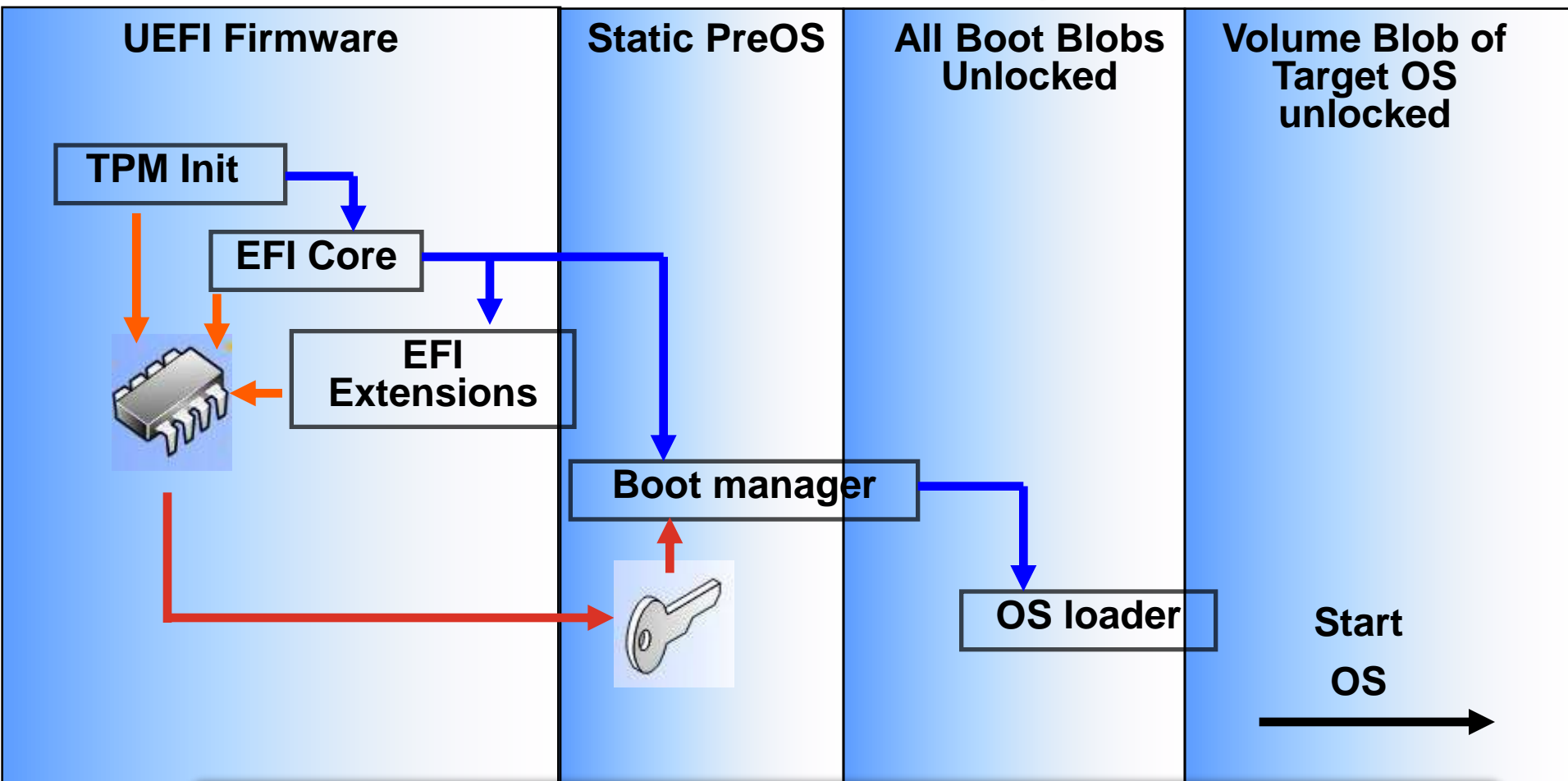


PCR_x = Register in TPM

Standardized way to measure and report

BitLocker™ Drive Encryption

Static Root of Trust Measurement of early boot components

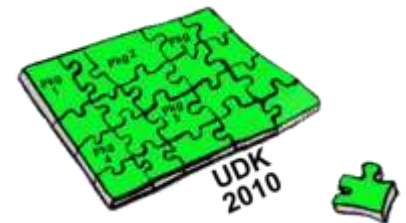


UEFI Windows* is using SRTM

UEFI 2.3.1 Secure Boot

Intel® UDK2010 SR1 Security Features

- UEFI Secure Boot
 - UEFI variable support for UEFI Secure Boot as defined by UEFI 2.3.1a (EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS attribute with EFI_VARIABLE_AUTHENTICATION_2 and EFI_VARIABLE_AUTHENTICATION support)
 - DXE Image Verification library to support UEFI Secure Boot (UEFI 2.3.1a)
 - PK x509 Certificate Support
 - Support EFI_VARIABLE_AUTHENTICATION_2 for PK variable format (UEFI 2.3.1a)
 - Add enable/disable mechanism for UEFI Secure Boot
- TCG Trusted Boot
 - TCG EFI Platform Specification

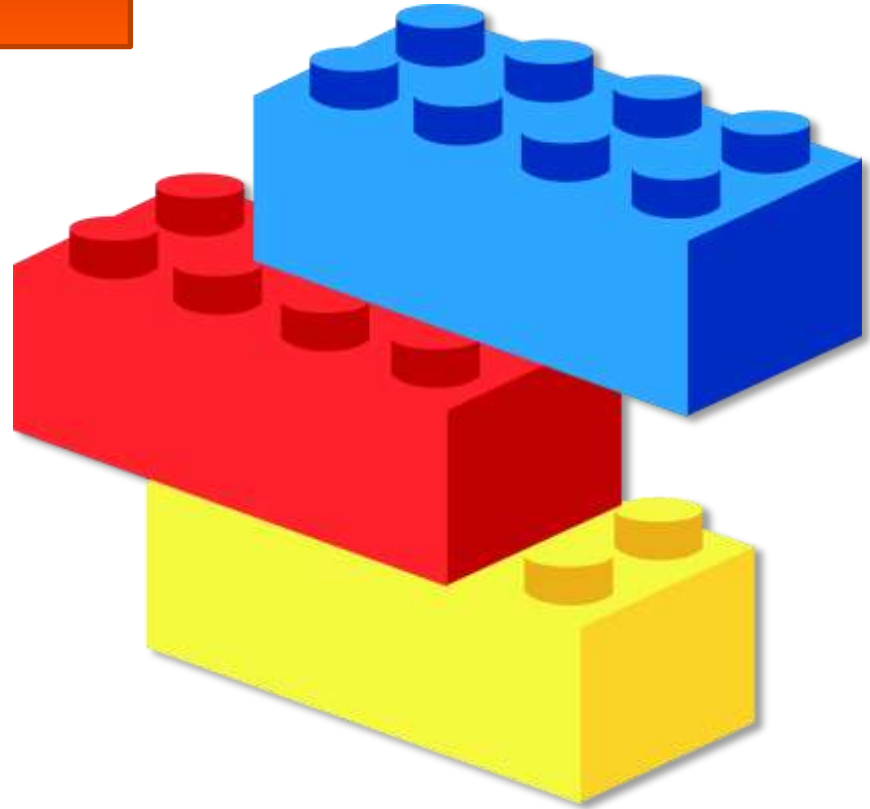


Secure Boot - Three Components

1. Authenticated Variables

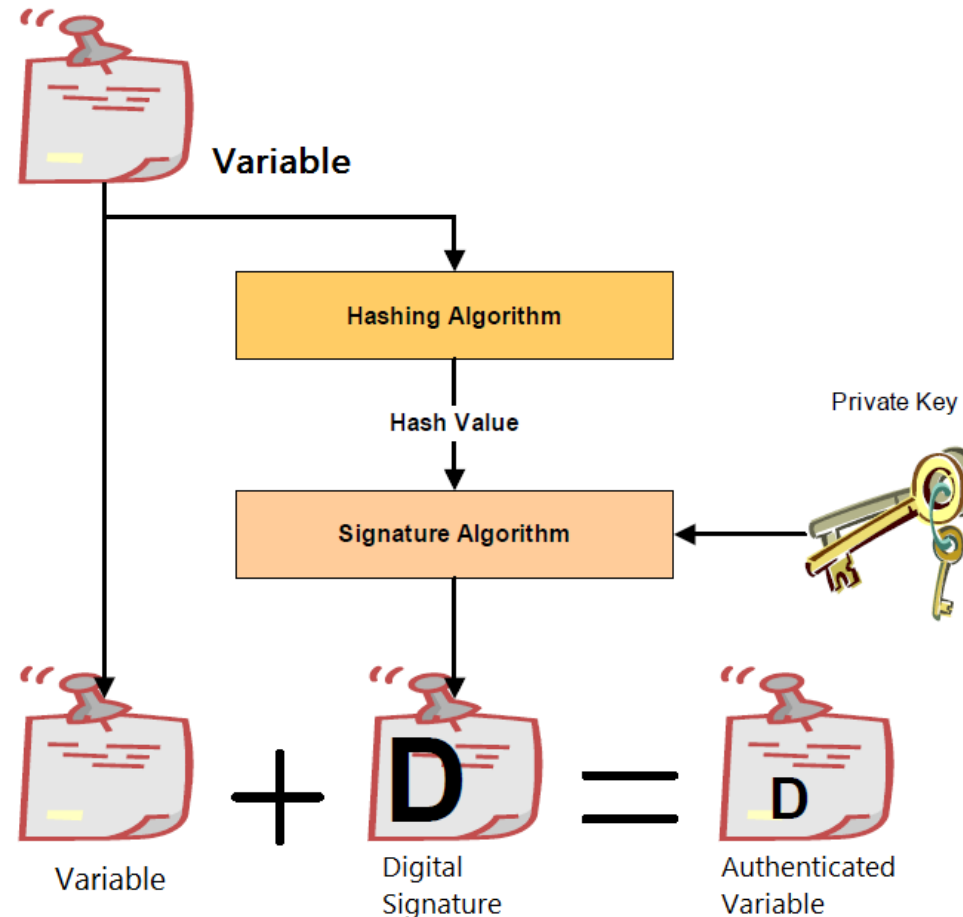
2. Driver Signing

3. System-Defined
Variables



UEFI Authenticated Variables

- Uses standard UEFI Variable Functions
- Available Pre-boot and also Runtime
- Typically stored in Flash
- Variable Creator signs Variable Hash with Private Key (PKCS-7 Format)
- Signature & Variable Passed Together for Create, Replace, Extend, or Delete
- Several System-defined variables for Secure Boot



Extensible Integrity Architecture

Updating Authenticated Variable



**New in
UEFI 2.3.1**

- Support for Append added (UEFI 2.3.1)
- Counter-based authenticated variable (UEFI 2.3)
 - Uses monotonic count to against suspicious replay attack
 - Hashing algorithm - SHA256
 - Signature algorithm - RSA-2048
- Time-based authenticated variable (UEFI 2.3.1) *
 - Uses timestamp as rollback protection mechanism
 - Hashing algorithm - SHA256
 - Signature algorithm - X.509 certificate chains
 - Complete X.509 certificate chain
 - Intermediate certificate support (non-root certificate as trusted certificate.



**New in
UEFI 2.3.1**

* only Time-based authenticated variables implemented in Tianocore.org UDK2010 SR1

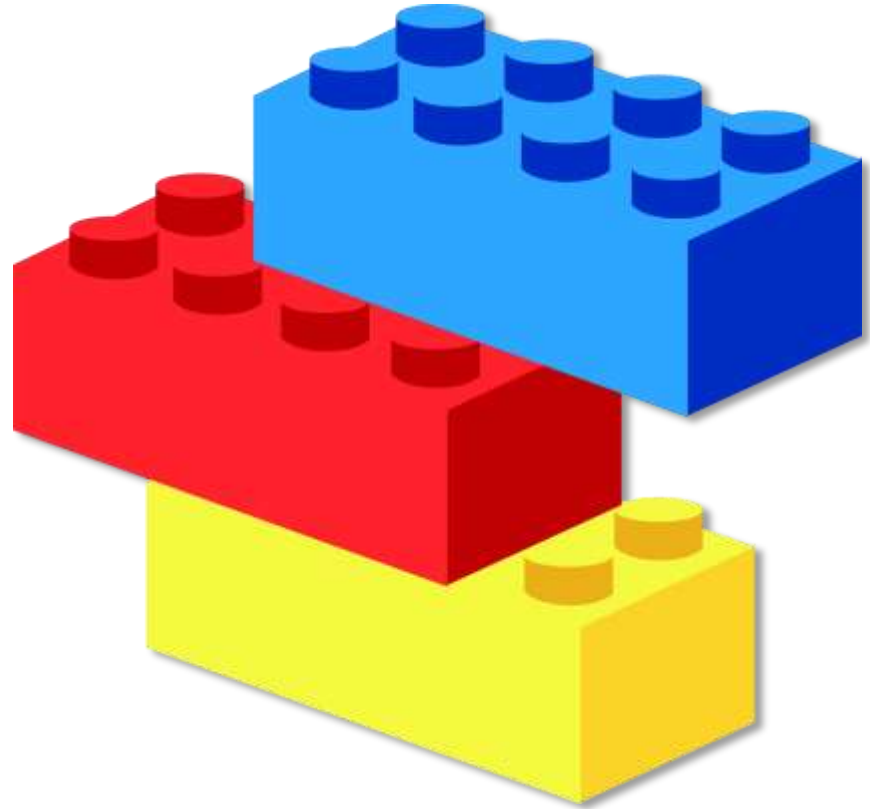
Protected Variables that can be Securely Updated

Secure Boot - Three Components

1. Authenticated Variables

2. Driver Signing

3. System-Defined
Variables



UEFI Driver Signing

- UEFI Driver Signing Utilizes Microsoft* Authenticode* Technology to sign UEFI executables
- In Secure Boot, signatures should be checked:
 1. UEFI Drivers loaded from PCI-Express Cards
 2. Drivers loaded from mass storage
 3. Pre-boot EFI Shell Applications, f/w updaters
 4. OS UEFI Boot-loaders
- UEFI Signing is not applied to
 1. Drivers in the Factory BIOS
 2. Legacy BIOS components (also known as CSM)
 3. CSM must be disabled in boot for system to be secure (UEFI boot)
 - CSM can be enabled in setup for non-UEFI boot options
 - Shell is not considered a secure boot option



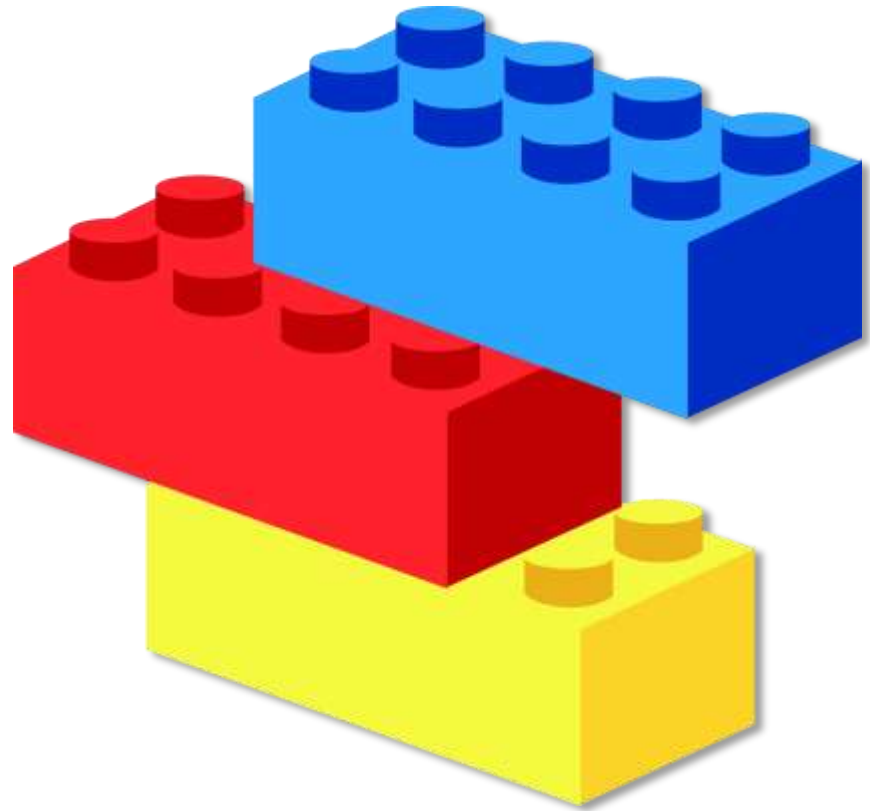
**Enhanced by
UEFI 2.3.1**

Secure Boot - Three Components

1. Authenticated Variables

2. Driver Signing

3. System Defined
Variables



Secure Boot Authenticated Variables

PK	Platform Key – Root key set to enable Secure Boot
KEK	Key Exchange Key List of Cert. Owners with db, dbx update privilege
db	List of Allowed Driver or App. Signers (or hashes)
dbx	List of Revoked Signers (or hashes)
SetupMode	1 = in Setup Mode, 0 = PK is Set (User Mode)
SecureBoot	1 = Secure Boot in force

Notes:

- **Owner of cert. in KEK can update db, dbx**
- **Owner of cert. in PK can update KEK**

UEFI Defines System Databases for Secure Boot

Secure Boot - Three Components

1. Authenticated Variables



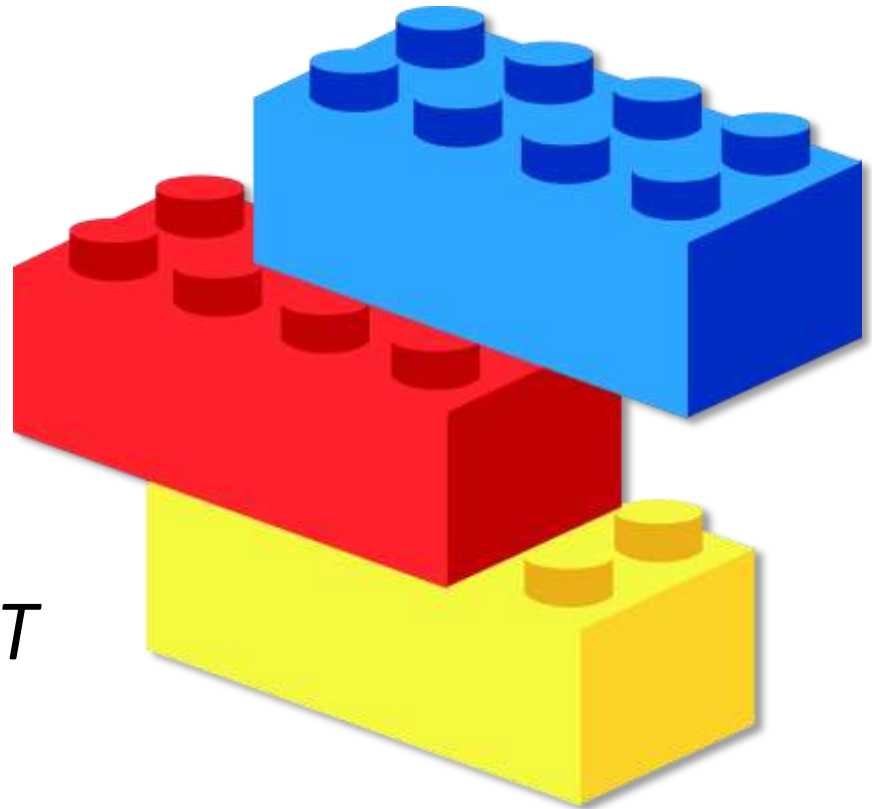
2. Driver Signing



3. System Defined Variables



UEFI 2.3.1 SECURE BOOT



Secure Boot Begins @ the Factory

Pre-production — Production — User

Certificate Generating
Station @ OEM

1



OEM collects certificates
provided by OSVs,
Partners, and OEM's own
keys.

"DB Generator" creates the
Initial Security Load for
new computers.

2

Initial
Security
Load

Initial Security Load is
installed onto each
computer at the factory,
enabling Secure Boot.

- 1) Initial db and dbx
- 2) KEK with allowed
updaters
- 3) Platform Key (PK)

3



After delivery, the
OEM or OSV can
update with new
certificates or
revoked certificates (dbx)

***OEM Responsible for Initializing Secure Boot
and can allow user to disable Secure Boot or
add KEK, PK, DB in setup***

Secure Boot Protects the User

User attempts to boot a compromised system



OS Boot-loader image checked against pre-loaded database



Root-kit fails checks, user protected by Secure Boot



Secure Boot Tests Signatures to Reject Potential Threats

For more information - UEFI Secure Boot

Intel Technology Journal, Volume 15, Issue 1, 2011, UEFI Today: Bootstrapping the Continuum, UEFI Networking and Pre-OS Security, page 80 at

<http://www.intel.com/technology/itj/2011/v15i1/pdfs/Intel-Technology-Journal-Volume-15-Issue-1-2011.pdf>

Rosenbaum, Zimmer, "A Tour Beyond BIOS into UEFI Secure Boot," Intel Corporation, July 2012

[http://sourceforge.net/projects/edk2/files/General%20Documentation/A To ur Beyond BIOS into UEFI Secure Boot White Paper.pdf/download](http://sourceforge.net/projects/edk2/files/General%20Documentation/A%20Tour%20Beyond%20BIOS%20into%20UEFI%20Secure%20Boot%20White%20Paper.pdf/download)

UEFI 2.3.1 specification: Sections 7.2 (Variable Services) and Sections 27.2 through 27.8 (Secure Boot) of the at www.uefi.org

Beyond BIOS: Developing with the Unified Extensible Firmware Interface, 2nd Edition, Zimmer, et al, ISBN 13 978-1-934053-29-4, Chapter 10 – Platform Security and Trust,

<http://www.intel.com/intelpress>

"Hardening the Attack Surfaces," MSFT 2012 UEFI Plugfest

[http://www.uefi.org/learning_center/UEFI Plugfest 2012Q1 Microsoft AttackSurface.pdf](http://www.uefi.org/learning_center/UEFI_Plugfest_2012Q1_Microsoft_AttackSurface.pdf)

"Building hardware-based security with a TPM" MSFT BUILD <http://channel9.msdn.com/Events/BUILD/BUILD2011/HW-462T>

Tunnel Mountain Intel DQ57TM UEFI 2.3.1 platform

Intel® UDK 2010 SR1 UP1 Compatible, supports UEFI 2.3.1 (updates match tianocore.org open source)

Pre-assembled systems available at HDNW, visit

<http://www.Tunnelmountain.net> based on DQ57TM

tomk@hdnw.com, (425) 943-5515 ext 42234. Use product name "Tunnel Mountain" when ordering

Comes with class 2 CSM and UEFI enabled firmware

Download site has UEFI class2 firmware(csm not on by default)

Comes with serial port for debug

Can be ordered with optional ITP connector and socketed SPI flash

***Romley Server UEFI 2.3.1 to be available Sep '12**

***Maho Bay DQ77MK Q4 '12**

*** - not available yet in validation test**



Visit <http://www.UEFIDK.com> for the latest collateral on UEFI systems

For UEFI developers

Getting your own UEFI 2.3.1 systems

Intel Production motherboards

UEFI 2.3.1 enabled Windows 8 client desktop motherboards

Sept - Oct 2012 rollout for 7 series Ivybridge systems (DH77, DQ77 etc.)

Goto Intel.com under support to update BIOS on motherboards

Ivybridge Ultrabooks certified for Windows 8

Contact your favorite BIOS vendor and ask for a UEFI 2.3.1 enabled board

What Vintage of UEFI is your system?

Get into BIOS setup

IS there a switch for CSM or legacy bios (to turn it off or force UEFI only)?

Goto UEFI shell type "ver" -> UEFI system spec revision (2.3.1?)

Does the system contain drivers for boot/console devices in system?

UEFI shell -> drivers command

Get SCT's from UEFI.org and run them (Do they pass or which tests fail?)

is it one the OS needs?

Goto Ubuntu's Linux Firmware Test suite – Run the UEFI suite to see if it passes. Contribute effort to test UEFI!

[Git://kernel.ubuntu.com/hwe/fwts](https://kernel.ubuntu.com/hwe/fwts)

UEFI Industry Resources

UEFI Forum



www.uefi.org

UEFI Open Source



www.tianocore.sourceforge.net

Intel UEFI Resources



www.intel.com/UDK

Intel EBC Compiler



<http://software.intel.com/en-us/articles/intel-c-compiler-for-efi-byte-code-purchase/>

UEFI Books/ Collateral



www.intel.com/intelpres

www.intel.com/technology/itj/2011/v15i1

Training/IHVs Contact

Laurie Jarlstrom

- Intel UEFI Training
- Laurie.Jarlstrom@intel.com

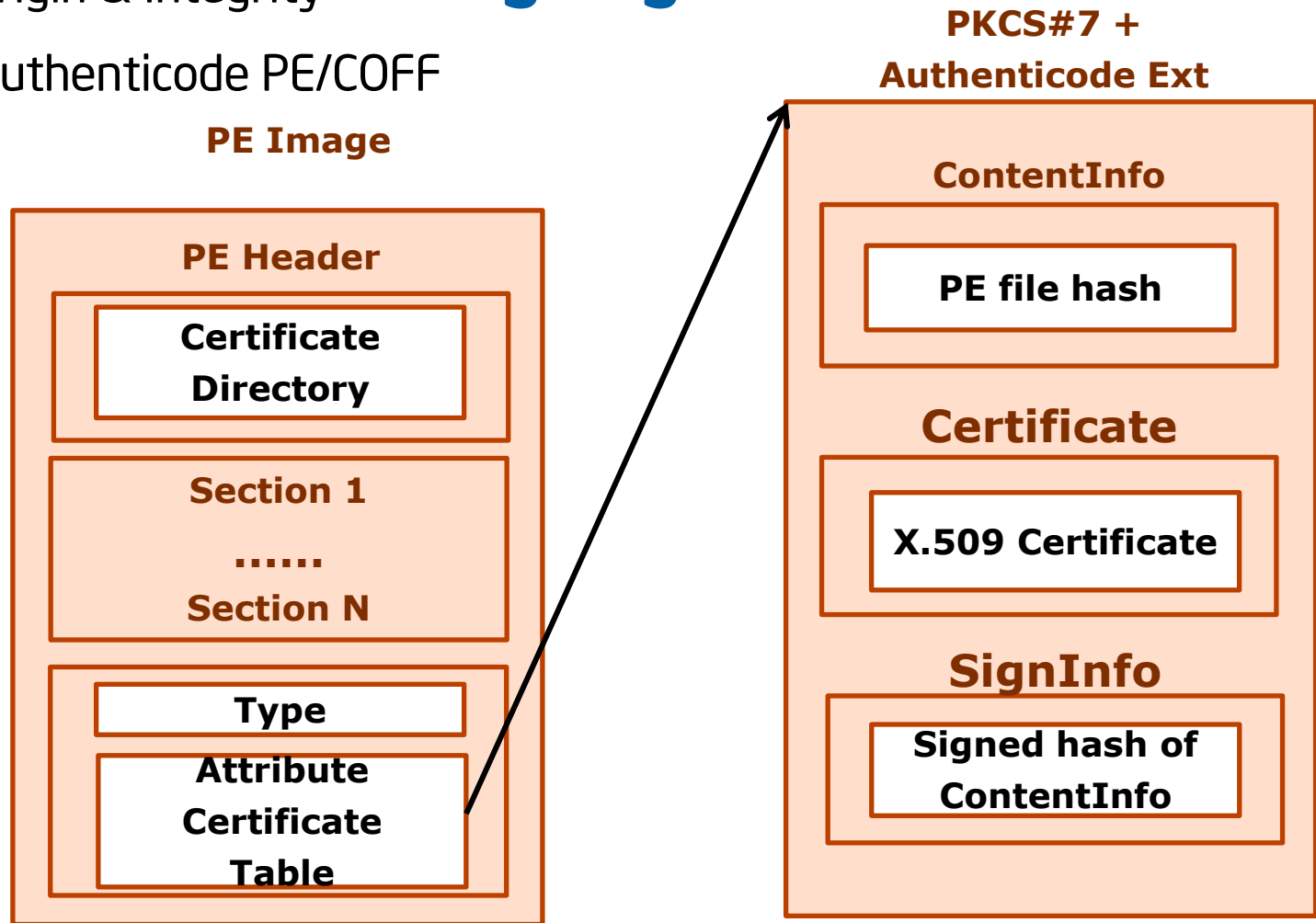
Brian Richardson

- Intel IHVs UEFI Support
- Brian.Richardson@intel.com

Backup

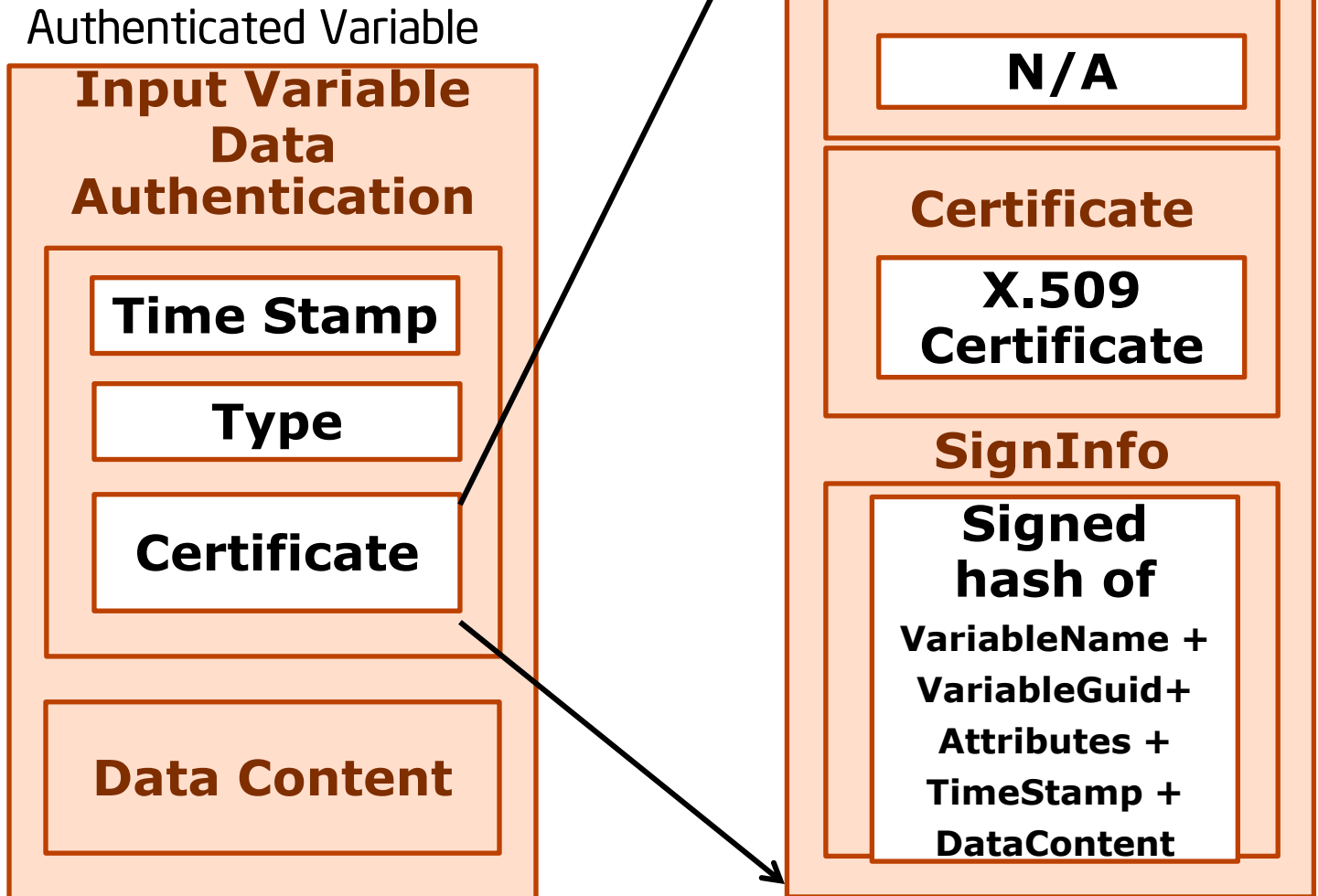
UEFI Image (driver & application/OS loader) Signing

- Why? – Origin & Integrity
- How? – Authenticode PE/COFF



UEFI Authenticated Variable

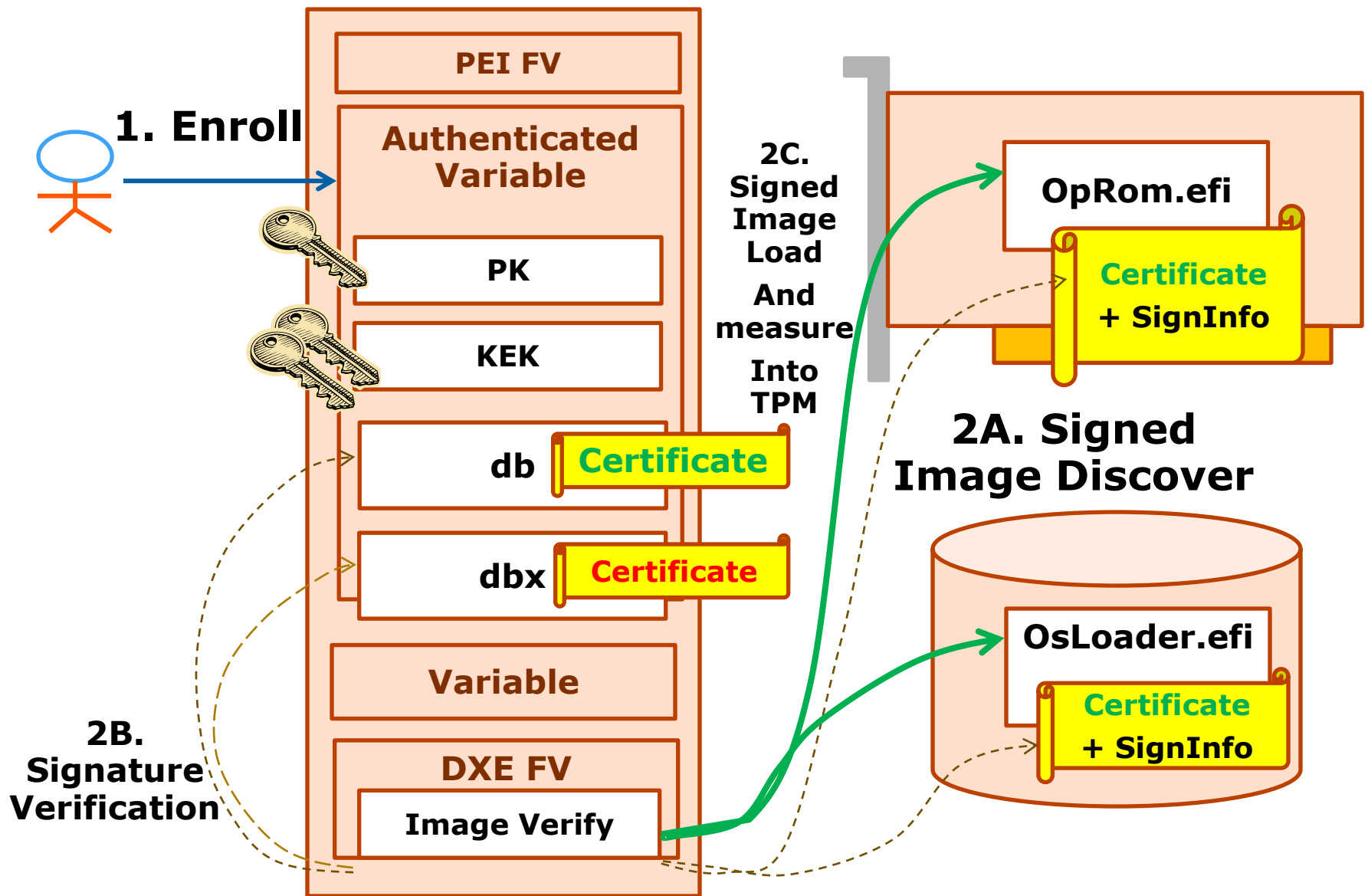
- **Why?** - Integrity (no confidentiality)
- **How?** - Time Based



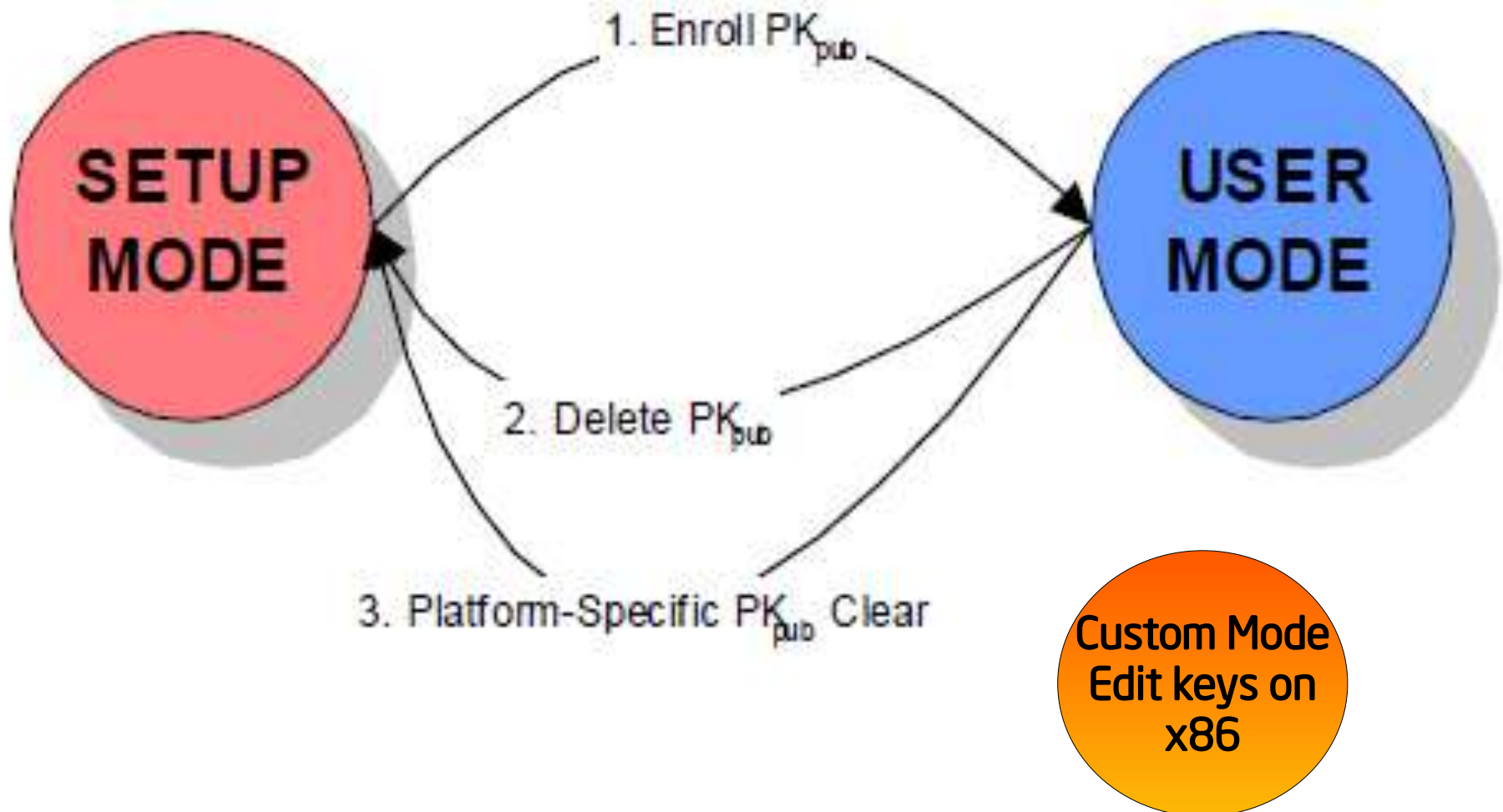
Secure Boot's Authenticated Variables

Key/ DB Name	Variable	Details
PkPub	PK	OEM and Platform FW- format is RSA-2048
Key Exchange Key	KEK	Platform FW and OS - format is RSA-2048
Authorized Signature DB	DB	Authorized Signing certificates - white list
Forbidden Signature DB	DBX	Unauthorized Signing certificates - Black list
Setup Mode	SetupMode	NULL - Secure Boot not supported 0 - PK is enrolled - in user mode User mode requires authentication 1 – Platform is in Setup mode – no PK enrolled
Secure Boot	SecureBoot	1-Platform in Secure boot mode

- Do NOT generate your own private keys
- Use FIPS certified key generators (hardware crypto)
- Protect your private keys (under physical protection)

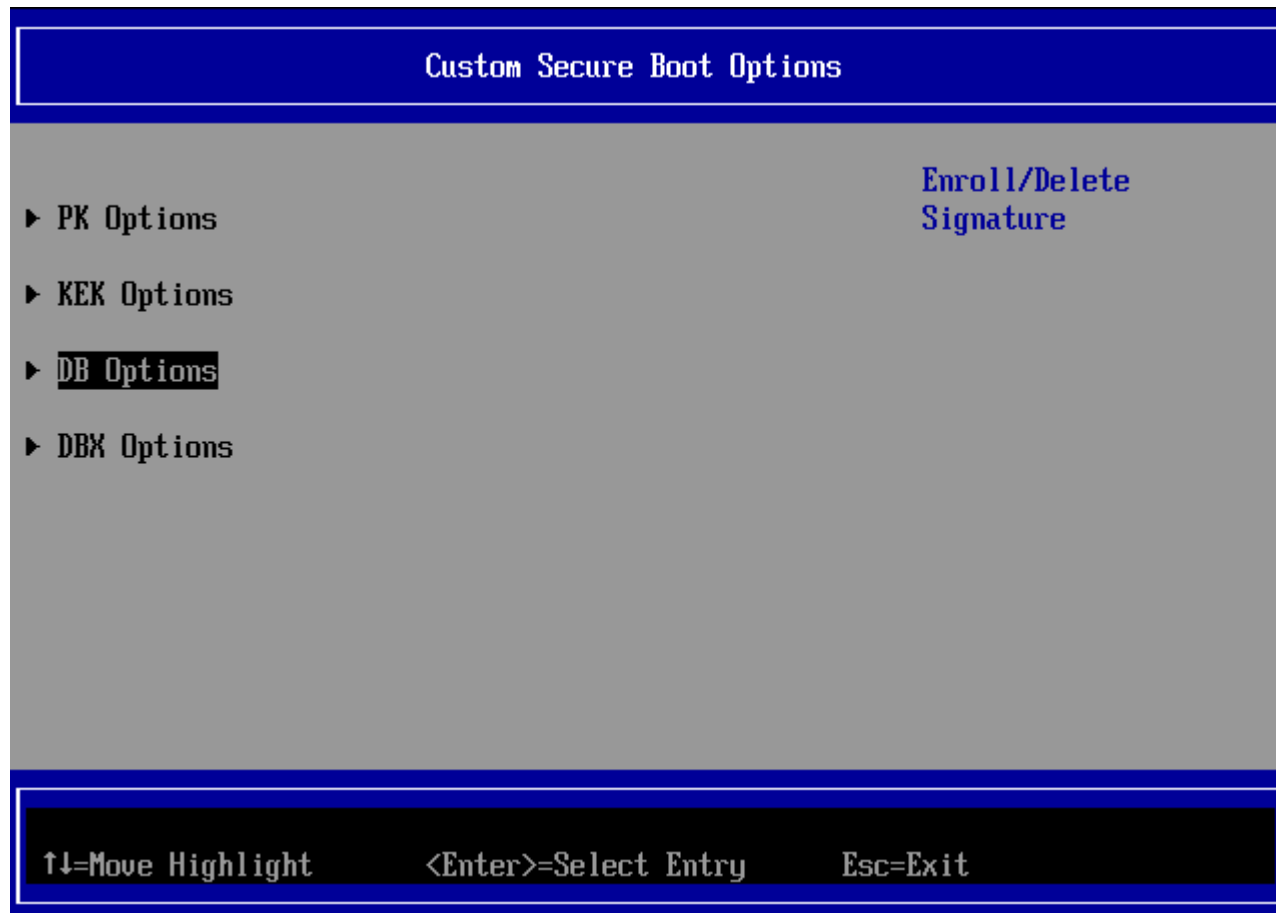


Put them altogether: UEFI Secure Boot







End user controls -Custom Secure Boot Options

- Enrolling DB and/or DBX for physically present user

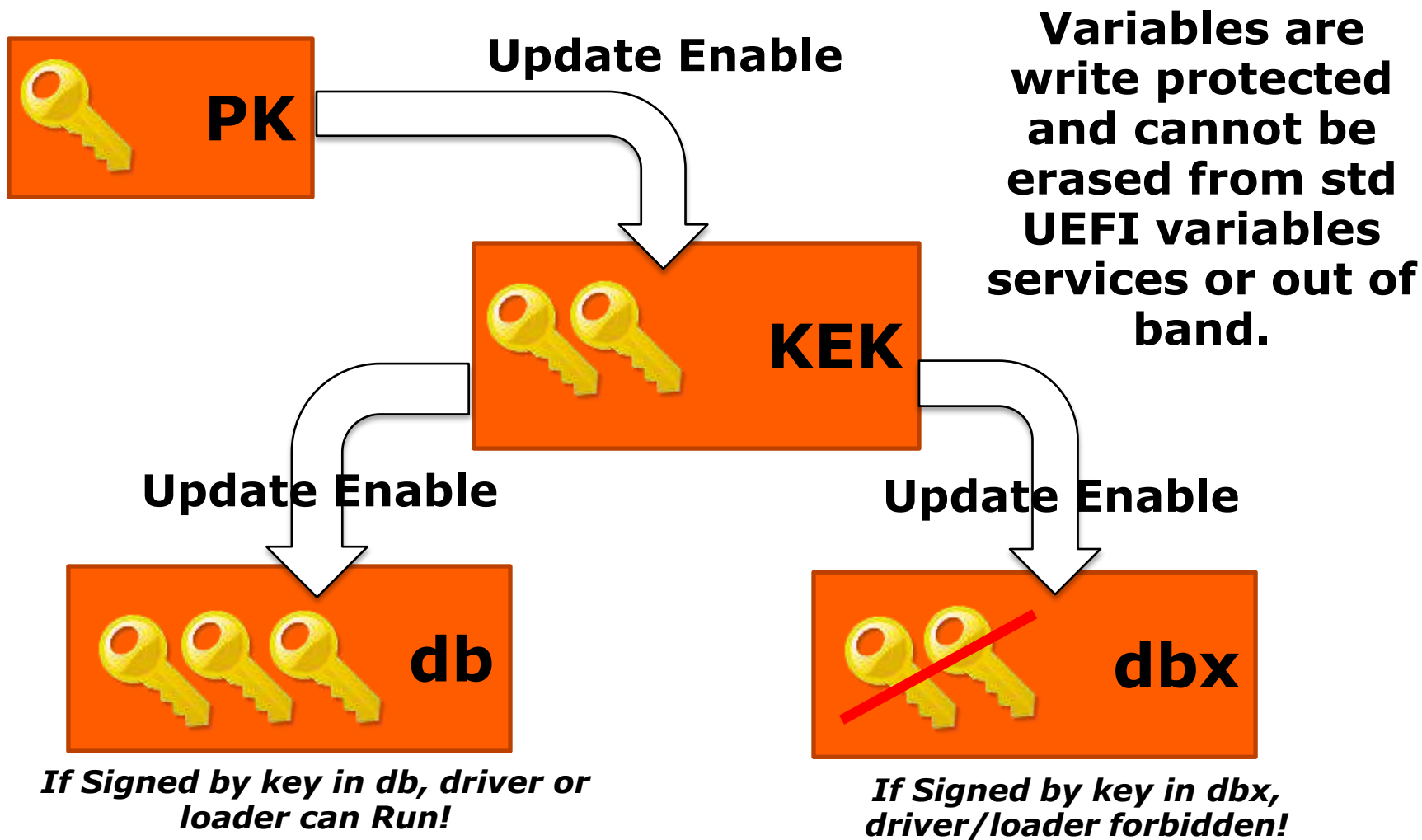


Disable Secure Boot

1. Select Custom Secure Boot Options
2. Select PK Options
3. Delete Pk (space bar)
4. Reset

Secure Boot		PK Options
 Attempt Secure Boot Secure Boot Mode ▶ Custom Secure Boot Options	 PK Options ▶ KEK Options ▶ DB Options ▶ DBX Options	<p>Choose to Delete PK, Otherwise keep the PK</p> <p>▶ Enroll PK Delete Pk </p> <p> Configuration changed. Reset to apply it Now. Press ENTER to reset</p>
↑↓=Move Highlight F9=Reset <Enter>	↑↓=Move Highlight	↑↓=Move Highlight F9=Reset to Defaults F10=Save <Spacebar>Toggle Checkbox Esc=Exit

UEFI Secure Boot Database Review



Public vs. Private Keys

- A pair of keys, one public, one private, are created
- Private keys stay secure at Partner or in the OEM's Security Office
- Private keys are used to 'sign' objects
- Only Public keys loaded into the Platform
- Public keys are used to check signatures



Who “Owns” The System Security Keys?

- PK – Key pair is created by Platform Manufacturer
Typically one PK pair used for a model or model Line
 - KEK – Key supplied by OS Partner, Msft
Optional: Include 2nd key created by OEM
 - db – OS Partner supplies Key, win8 msft
CA Partner supplies Key, UEFI CA hosted by msft
Optional: OEM App Signing Key
- Bios update key should not show up in DB , kek or PK

Signature Tests using db Keys Block Rogue S/W!

OEM Administration

- Keys are installed for testing with target OS
- Keys are installed in the factory before shipping

- Preparation Tasks

1. Gather public keys from partners
2. Generate PK for model
3. Make a package of initial key load
4. Occasional maintenance of forbidden list



- Repetitive Tasks

1. Factory will boot and install the initial key load

Careful Preparation Delivers Successful Launch