presented by

UEFI Conformance Profiles: Allowing "Reduced Model" Implementations

UEFI 2022 Virtual Plugfest

November 17, 2022

Dick Wilkins (Phoenix Technologies, Inc.) and Samer El-Haj-Mahmoud (Arm Limited)

Meet the Presenters

Dick Wilkins Principal Technology Liaison, Phoenix

Samer El-Haj-Mahmoud Senior Principal Architect, Arm

More Questions?

Following today's webinar, join the live, interactive WebEx Q&A for the opportunity to chat with the presenters

- **Visit this link to attend:** https://bit.ly/3esVWIR
  Meeting number: 2553 732 0183
  Password: uefiforum (83343678 from phones and video systems)

Agenda

- Introduction & Motivation
- What is normally required
- The “Conformance Table”
- An example
- Why it is important
- Next steps
- Questions?

Motivation

- EFI was developed for traditional computing platforms
- The market has demanded that UEFI be used on other types of systems
- Many of these new targets are “constrained” and cannot, or do not choose to, support all required aspects of UEFI

What Is Normally Required

Table 2-12 - UEFI Specification 2.10 - Required Elements

Element	Description
EFI_SYSTEM_TABLE	Provides access to UEFI Boot Services, UEFI Runtime Services, consoles, firmware vendor information, and the system configuration tables.
EFI_BOOT_SERVICES	All functions defined as boot services.
EFI_RUNTIME_SERVICES	All functions defined as runtime services.
EFI_LOADED_IMAGE_PROTOCOL	Provides information on the image.
EFI_LOADED_IMAGE_DEVICE_PATH_PROTOCOL	Specifies the device path that was used when a PE/COFF image was loaded through the EFI Boot Service LoadImage().
EFI_DEVICE_PATH_PROTOCOL	Provides the location of the device.
EFI_DECOMPRESS_PROTOCOL	Protocol interfaces to decompress an image that was compressed using the EFI Compression Algorithm.
EFI_DEVICE_PATH_UTILITIES_PROTOCOL	Protocol interfaces to create and manipulate UEFI device paths and UEFI device path nodes.

Additional Required Elements

- Platform Specific Elements

  Depending on specific hardware features, more elements may be required

- Driver Specific Elements

  If specific drivers are included in platform firmware, features may be required in case they are needed by those drivers

To Be UEFI Specification Conformant…

- Firmware MUST provide all required elements
- But constrained platforms may not actually need all the required elements
- What do we do?

Conformance Profiles

- A new UEFI configuration table
  `EFI_CONFORMANCE_PROFILE_TABLE`
- The table contains one or more GUID(s) identifying a “profile” of the required elements supported by this platform
- Use of `EFI_CONFORMANCE_PROFILES_UEFI_SPEC_GUID` in the table indicates full conformance to the UEFI spec required elements

More on the Table

1. If the table is not present, then it can be assumed that firmware complies with the UEFI Specification version identified in the `EFI_SYSTEM_TABLE`.
2. If the table is present and contains the `EFI_CONFORMANCE_PROFILES_UEFI_SPEC_GUID`, the firmware complies with the UEFI Specification version identified in the `EFI_SYSTEM_TABLE`.
3. If the table exists and contains one or more other GUIDs, it is based on the UEFI version in the `EFI_SYSTEM_TABLE` but supports only those elements agreed to by the specification that assigned the GUID.

Who Creates Non-Conformant GUIDs

- Anyone can create a GUID and corresponding subset of required elements
- They can be publicly documented or held private
- They need not be declared in the UEFI specification
- If a creator chooses to have a GUID included in the UEFI spec, they should point to a public location where it is documented

How Is the Table Used?

- Any code that intends to use EFI interfaces should scan for the new table
- If the table is not present or contains the `EFI_CONFORMANCE_PROFILES_UEFI_SPEC_GUID`, software can assume a fully conformant system
- If not, standard software can assume the needed elements may not be present, and can respond accordingly

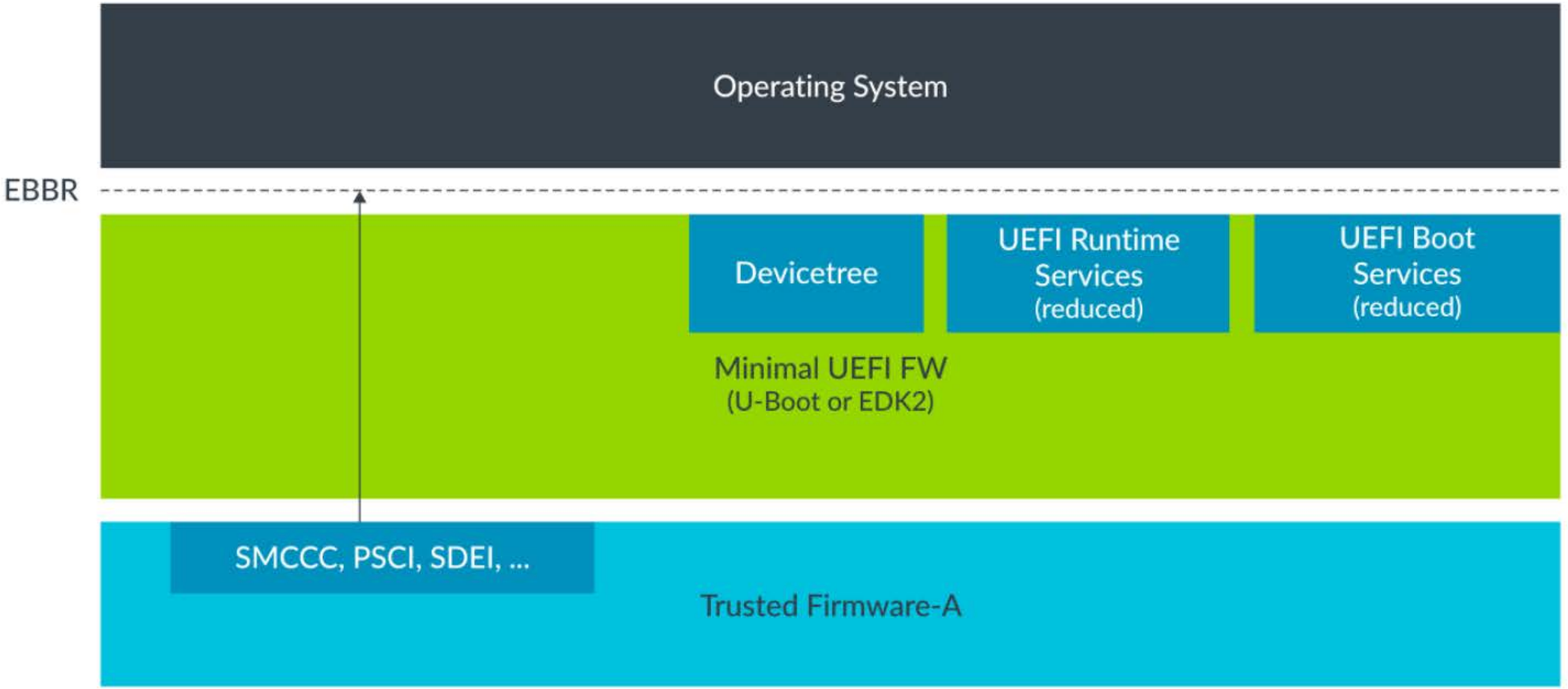
Other GUIDs?

- If another GUID is present, software can assume that a subset of required elements is supported
- Software can check for a GUID indicating support for their specific needs

Example – EBBR

- The Embedded Base Boot Requirements (EBBR) specification (https://github.com/arm-software/ebbr) defines requirements of embedded systems to enable interoperability between SoCs, hardware platforms, firmware implementations, and operating systems.
- Focuses on the UEFI interfaces necessary to boot OSes
  - EBBR does not require a specific firmware implementation.
  - Both U-Boot (https://github.com/u-boot/u-boot) and EDK2 (https://www.tianocore.org/) implement EBBR.

EBBR Firmware Interfaces



EBBR UEFI Requirements

- The EBBR specification defines a reduced set of UEFI requirements that deviate from the UEFI spec requirements
  - UEFI Required Elements (UEFI 2.10, section 2.6.1)
  - Required Platform Specific Elements (UEFI 2.10, section 2.6.2)
  - UEFI Global variables (UEFI 2.10, section 3.3)
  - Etc.
- The reduced set of requirements fit the EBBR targeted market and operating systems

EBBR UEFI Requirements

EBBR deviations from the Required UEFI Elements in UEFI 2.10 specification, section 2.6.1

Element	Description
EFI_BOOT_SERVICES	Methods for unsupported or unimplemented behavior must return an appropriate error code.
EFI_RUNTIME_SERVICES	Methods for unsupported or unimplemented behavior must return an appropriate error code. If any runtime service is unimplemented, it must be indicated via the EFI_RT_PROPERTIES_TABLE.
EFI_DECOMPRESS_PROTOCOL	Built-in EFI decompression is rarely used and therefore not required.

EBBR UEFI Requirements

EBBR deviations from the Platform Required Elements in UEFI 2.10 specification, section 2.6.2

.	Description
LoadImage()	Not required to install an EFI_HII_PACKAGE_LIST_PROTOCOL for an image containing a custom PE/COFF resource with the type 'HII'.
ConnectController()	Not required to support the EFI_PLATFORM_DRIVER_OVERRIDE_PROTOCOL, EFI_DRIVER_FAMILY_OVERRIDE_PROTOCOL, and EFI_BUS_SPECIFIC_DRIVER_OVERRIDE_PROTOCOL.
Option ROM support	EBBR use cases do not have requirements to generically support any PCIe add-in cards at the firmware level. When PCIe devices are used, drivers for the device are often built into the firmware itself rather than loaded as option ROMs.
Some HII protocols, Disk IO, PXE boot, UEFI Network stack, PCI bus support, USB bus support, NVMe pass through, Serial IO, etc.	Not required for most EBBR use cases

EBBR UEFI Requirements

EBBR deviations from the UEFI Runtime Services in UEFI 2.10 specification, section 8

EFI_RUNTIME_SERVICES function	Before ExitBootServices()	After ExitBootServices()
GetTime	Required if RTC present	Optional
SetTime	Required if RTC present	Optional
GetWakeupTime	Required if wakeup supported	Optional
SetWakeupTime	Required if wakeup supported	Optional
SetVirtualAddressMap	N/A	Required
ConvertPointer	N/A	Required
GetVariable	Required	Optional
GetNextVeriableName	Required	Optional
SetVariable	Required	Optional
GetNextHighMonotonicCount	N/A	Optional
ResetSystem	Required	Optional
UpdateCapsule	Required for in-band update	Optional
QueryCapsuleCapabilities	Optional	Optional
QueryVariableInfo	Optional	Optional

EBBR and the Conformance Profile

- The EBBR specification defines a UEFI Conformance Profile GUID for EBBR 2.0:

EFI Conformance Profile Table

The following GUID in the EFI Conformance Profile Table is used to indicate compliance to version 2.0 of the EBBR specification:

```
#define EFI_CONFORMANCE_PROFILE_EBBR_2_0_GUID
{ 0xcce33c35, 0x74ac, 0x4087, \
{ 0xbc, 0xe7, 0x8b, 0x29, 0xb0, 0x2e, 0xeb, 0x27 }}
```

- Support is merged in U-Boot: https://github.com/u-boot/u-boot/blob/master/lib/efi_loader/efi_conformance.c

Next Steps

- Support `EFI_CONFORMANCE_PROFILE_TABLE` in TianoCore/EDK2
  - Implement the `EFI_CONFORMANCE_PROFILES_UEFI_SPEC_GUID` for fully compliant systems
  - Implement the `EFI_CONFORMANCE_PROFILES_EBBR_2_0_GUID` for EBBR compliant systems
- Add support for conformance profiles in compliance testing suites
  - UEFI Self Certification Test (SCT)
  - Firmware Test Suite (FWTS)
- Support for checking for the conformance profiles in Operating Systems, if needed

Summary

- There is now a way to use the UEFI standard with just the features you need
- Coordinate between the firmware and system being booted
- Create a GUID to indicate your specific recipe
- Arm EBBR is an example

Questions?

More Questions?

Following today's webinar, join the live, interactive WebEx Q&A for the opportunity to chat with the presenters

- **Visit this link to attend:** https://bit.ly/3esVWIR
  Meeting number: 2553 732 0183
  Password: uefiforum (83343678 from phones and video systems)

Thanks for attending the UEFI 2022 Virtual Plugfest

For more information on the Unified EFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by