UEFI
Conformance Profiles

Allowing “Reduced Model”
Implementations

Authors:

Richard Wilkins, Ph.D.
Principal Technology Liaison
Phoenix Technologies, Ltd.
Dick_Wilkins@phoenix.com

Samer El-Haj-Mahmoud
Senior Principal Architect
Arm Limited
Samer.El-Haj-Mahmoud@arm.com

July 2022

Understanding UEFI "Conformance Profiles" and Their Uses

This document explains a new "Conformance Profiles" capability, provided by the UEFI Specification, that will allow the creation subsets of UEFI required interfaces, along with specifics of how to communicate descriptions of those subsets to loaded software, in a standard way.

Introduction

When the Extensible Firmware Interface (EFI) was originally developed, it targeted traditional computing platforms (servers, desktops, and laptops) using general purpose operating systems. It provided a standard, and easily extensible, interface between platform firmware and systems software. This standard provided for common interactions and interoperability between various platforms, their firmware implementations, and the various available systems software implementations.

In 2005, the Unified EFI Forum, a nonprofit industry standards body, was formed to maintain the Unified Extensible Firmware Interface (UEFI) Specification.

The interoperability provided by the UEFI standard has encouraged its use in an ever-growing series of platform and software types. Some industry groups have required that standard UEFI interfaces be provided for a platform to be certified by their organization. These market-driven requirements have resulted in the UEFI standard interfaces being implemented on many types of systems, well beyond the originally targeted computing platforms.

As this migration to new platform types has been happening, developers of firmware for platform types not originally contemplated by the developers of EFI, have pointed out that some UEFI standard required interfaces may not be needed by some platform types. Implementing these unused interfaces and their supporting code is an unnecessary burden to smaller and constrained devices such as those for the IoT, embedded, automotive, and other markets.

The new capability described here, to provide "Conformance Profiles", will allow subsets of the UEFI required interfaces to be created and the specifics of how a subset's description is communicated to loaded software in a standard way.

What is normally required?

While the UEFI Specification is well over 2400 pages in length, with hundreds of interfaces defined, only a small subset are **"required"** to be implemented. These required elements are documented in Chapter 2 of the specification. But even that subset may be an unnecessary burden to some constrained devices.

If a firmware implementation lacks some required elements, then it is not fully compliant with the UEFI Specification and software may not have access to all expected facilities. On the other hand, it may be desirable for a constrained platform to provide only those elements that will actually be used by their target software and remove those that are not going to be needed.

UEFI Forum contributing members have agreed that implementing requirements subsets for specific platforms is desirable. They also saw the need to communicate the details of the subset being provided

by a platform to loaded software so that software is aware of the limitations to the interfaces provided by that platform. Software could then decide if this was a viable environment for it to run.

The EFI_CONFORMANCE_PROFILE_TABLE

Section 4.6 of the latest UEFI Specification will define a new conformance profile UEFI configuration table. This table will list identifiers (GUIDs) of the specific profiles of required elements supported by a platform firmware implementation.

The definition of these "profiles" of required elements are defined and agreed to by the platform firmware developers and the software intended to run on that platform. They will typically **NOT** be recorded in the UEFI Specification. Instead, they may be defined in other industry standard specifications, or platform-specific specifications.

Software running on a platform can check for the presence of the EFI_CONFORMANCE_PROFILE_TABLE and assume:

1. If the table is not present, the firmware complies with the UEFI Specification version identified in the EFI_SYSTEM_TABLE. This is needed for backward compatibility. If the table is not present, it is assumed that the firmware implementation was done prior to the table's definition being included in the specification, and the firmware is fully compliant with the UEFI "required elements".
2. If the table is present and contains the EFI_CONFORMANCE_PROFILES_UEFI_SPEC_GUID defined below, 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.

The following GUID is defined in the UEFI Specification and indicates full conformance to the UEFI specification required elements.

```
#define EFI_CONFORMANCE_PROFILES_UEFI_SPEC_GUID \
{ 0x523c91af, 0xa195, 0x4382, \
{ 0x81, 0x8d, 0x29, 0x5f, 0xe4, 0x00, 0x64, 0x65}}
```

After the definition of the EFI_CONFORMANCE_PROFILE_TABLE becomes part of the published UEFI Specification, it is recommended that this table be included in all future UEFI based firmware implementations with the correct GUID(s) included in that table.

If the table is present and defines a GUID for a subset of required elements, the loaded operating system or software can assume that predefined subset is present without the need of detailed probing of the implementation to confirm the presence of any particular interface.

Any firmware developer creating an implementation for a product that desires to use a subset of the UEFI required elements may define a GUID identifying their specific subset implementation. These GUIDs **need not be** recorded in the UEFI Specification and are only of interest to software confirming that the firmware they are communicating with meets their needs. Note: If a developer of a subset

profile would like to have their GUID documented in the UEFI Specification, they should have the GUID and a pointer to where it is publicly documented brought to the UEFI Specification working group for consideration of publication in a future UEFI Specification.

Software that requires a fully conformant UEFI firmware implementation should check for the presence of the conformance profiles table and if found, should check for GUID(s) not matching the EFI_CONFORMANCE_PROFILES_UEFI_SPEC_GUID. If they are found, then that software knows that it is running on a non-conformant firmware subset and will likely need to report an appropriate error and fail to run**.**

How "Non-Conformant" Implementations Use the Table

If a platform/firmware developer and a software provider choose a set of UEFI interfaces needed by their platform solution, they can match their software to a specific firmware solution using the EFI_CONFORMANCE_PROFILE_TABLE. Together, firmware and software developers agree to a subset of 'required elements', a set of 'optional interface protocols' and possibly one or more proprietary 'interface protocols' required for their product. They assign a GUID for that pattern of interface support and may publish it for others to use or may choose to keep it to themselves. The firmware publishes their GUID in the EFI_CONFORMANCE_PROFILE_TABLE and the software checks it at "boot" time. In this way, the software can confirm it is running on the correct platform early in its startup process.

A Usage Example

This is an example of one way in which this capability may be used.

At the time this document was created, Arm Holdings, Ltd., along with their developer community, have published a firmware specification for embedded devices called the Embedded Base Boot Requirements (EBBR) specification version 2.0. This specification describes an interface between platform firmware and operating systems appropriate for their target embedded market. This EBBR Specification describes a specific subset of the UEFI required elements.

Platforms that implement the EBBR Specification should implement the EFI_CONFORMANCE_PROFILE_TABLE, with a specific GUID to indicate EBBR 2.0 compliance and deviations from the UEFI Specification of required elements. This GUID would be defined and published in the EBBR Specification or other documents. This publication of a conformance profile GUID is not required by the UEFI Forum but is encouraged. It is **NOT** required that the GUID or any other details contained in the specification be added to the UEFI Specification unless the developer or group chooses.

Value of this feature

This new capability enables the usage of standard UEFI interfaces on a much wider range of platforms. While most existing implementations of the standard currently supply fully conformant firmware based on the Tianocore/EDK2 open-source or UEFI compliant proprietary codebases, this ability to define subset profiles allows new platform types an easier implementation of the interface standard on other codebases such as U-Boot, coreboot, and Linux Boot, for example.

This also opens the door to wider acceptance of the UEFI standard across the industry and allows it to help in the interoperability of devices from the largest computing servers to the smallest IoT and sensor devices. As the standard includes many security features, this broad acceptance enables more secure

implementations of all these devices, a real benefit to all users.

> ***Feature Status***
>
> *At the time of this writing, this feature has been approved by the UEFI Specification working group for inclusion in the next published version of the specification (expected to be 2.10). Developers may implement this feature at a time of their choosing.*
>
> *This feature was developed using the "Code First" process of the UEFI Forum. This process acknowledges the valuable insights and contributions of the open-source communities to the development of the UEFI Specifications. Developers who cannot be contributing members of the UEFI Forum can still contribute to the specs. Open-source venues (for example, the Tianocore open-source firmware project), that have compatible intellectual property rules, can provide a place to discuss, design, and demonstrate implementations of new spec features. When ready, a UEFI Contributor member may bring that completed work, that was created in public view, to a UEFI working group for acceptance or rejection as a package.*

About UEFI Forum

The UEFI Forum, a nonprofit industry standards body, champions firmware innovation through industry collaboration and the advocacy of a standardized interface that simplifies and secures platform initialization and firmware bootstrap operations. Both developed and supported by representatives from more than 350 industry-leading technology companies, UEFI Forum specifications promote business and technological efficiency, improve performance and security, facilitate interoperability between devices, platforms, and systems, and comply with next-generation technologies.

The Forum's spheres of input and influence are large: Membership represents major voices from all players in the industry—open source to proprietary technology, hardware to software, mobile to stationary devices. The Forum collaborates with other standards groups that are essential to computing.

For More Information

Please visit www.uefi.org for more information about UEFI, including current specifications and membership options.

# # #