**# Title:**

Introduce UEFI Conformance Profiles

**# Status:**

Draft

**# Document:**

UEFI Specification 2.9 (Future Errata)

**# License:**

SPDX-License-Identifier: CC-BY-4.0

**# Submitter:**

* Samer El-Haj-Mahmoud, ARM
* TianoCore Community (<https://www.tianocore.org>)

**# Summary of the change**

This proposal updates the UEFI specification to allow publishing UEFI Conformance profiles, allowing implementations to advertise to the OS if they follow a reduced-model of UEFI implementation, which conforms to an external specification defining a subset of the UEFI specification requirements.

**# Benefits of the change**

This proposal updates the UEFI specification to introduce the concept of “Conformance Profiles”. One motivation for such minimized firmware/OS interface is to allow various firmware architecture, such as U-Boot, linuxboot, to implement the UEFI ABI, starting with a well-defined subset, that is compatible with some OSes targeting those firmware projects.

**# Impact of the change**

* **Platform FW:**
  + Generally: no impact to existing mainstream UEFI FW implementations (such as those based on TianoCore / EDK2 and derived commercial FW)
    - Such implementations can optionally implement the new UEFI configuration table in this proposal if desired
  + FW implementations with deviations from UEFI spec requirements (as defined by other external specifications) may choose to publish their profiles to indicate so.
    - An example is the EFI implementation in UBoot (https://github.com/u-boot/u-boot), which may want to advertise the “EBBR v 2.0 Profile”
    - Another example is LinuxBoot project (<https://www.linuxboot.org/>), which may choose to define its own deviations from the UEFI spec requirements, and publish a conformance profile.
* **Operating Systems:**
  + OSes may need to be updated to look at known profiles to get insight on the platform FW capabilities without deep probing of interfaces. For instance, Linux may choose to do this to identify UBoot based UEFI FW that is EBBR compliant. This is optional, and may be left until use-cases come up.
* **Compliance tests**
  + Test suites like FWTS and SCT should be updated to understand the profiles concept, and use in judging the FW compliance to those specifications, as needed

**# Detailed description of the change [normative updates]**

* Links to other sections highlighted
* Insertions highlighted
* Removals in ~~red~~

**2.6 Requirements**

This document is an architectural specification. As such, care has been taken to specify architecture in ways that allow maximum flexibility in implementation. However, there are certain requirements on which elements of this specification must be implemented to ensure that operating system loaders and other code designed to run with UEFI boot services can rely upon a consistent environment.

For the purposes of describing these requirements, the specification is broken up into required and optional elements. In general, an optional element is completely defined in the section that matches the element name. For required elements however, the definition may in a few cases not be entirely self contained in the section that is named for the particular element. In implementing required elements, care should be taken to cover all the semantics defined in this specification that relate to the particular element.

**2.6.1 Required Elements**

Table 2-11 lists the required elements. Any system that is designed to conform to this specification *must* provide a complete implementation of all these elements. This means that all the required service functions and protocols must be present and the implementation must deliver the full semantics defined in the specification for all combinations of calls and parameters. Implementers of applications, drivers or operating system loaders that are designed to run on a broad range of systems conforming to the UEFI specification may assume that all such systems implement all the required elements.

A system vendor may choose not to implement all the required elements, for example on specialized system configurations that do not support all the services and functionality implied by the required elements. However, since most applications, drivers and operating system loaders are written assuming all the required elements are present on a system that implements the UEFI specification; any such code is likely to require explicit customization to run on a less than complete implementation of the required elements in this specification. On such systems, the implementation may choose to advertise the profile which it conforms to using EFI\_CONFORMANCE\_PROFILES\_TABLE (see Section 4.6).

…

**2.6.2 Platform-Specific Elements**

There are a number of elements that can be added or removed depending on the specific features that a platform requires. Platform firmware developers are required to implement UEFI elements based upon the features included. The following is a list of potential platform features and the elements that are required for each feature type:

…

4.6 EFI Configuration Table & Properties Table

…

**EFI\_MEMORY\_ATTRIBUTES\_TABLE**

**…**

**EFI\_CONFORMANCE\_PROFILE\_TABLE**

**Summary**

This table allows the platform to advertise its UEFI specification conformance in the form of pre-defined profiles. Each profile is identified by a GUID, with known profiles listed in the Description section below.

The absence of this table shall indicate that the platform implementation is conformant with the UEFI specification requirements, as defined in section 2.6. This is equivalent to publishing this configuration table with the EFI\_CONFORMANCE\_PROFILES\_UEFI\_SPEC\_GUID conformance profile.

**Prototype**

#define EFI\_CONFORMANCE\_PROFILES\_TABLE\_GUID \

{ 0x36122546, 0xf7e7, 0x4c8f, \

{ 0xbd, 0x9b, 0xeb, 0x85, 0x25, 0xb5, 0x0c, 0x0b }}

typedef struct {

UINT16 Version;

UINT16 NumberOfProfiles;

//EFI\_GUID ConformanceProfiles [];

} EFI\_CONFORMANCE\_PROFILES\_TABLE;

*Version* Version of the table, must be 0x1

#define EFI\_CONFORMANCE\_PROFILES\_TABLE\_VERSION 0x1

*NumberOfProfiles* The number of conformance profiles GUIDs present in *ConformanceProfiles*.

*ConformanceProfiles* An array of conformance profile GUIDs that are supported by this system.

The address reported in the EFI configuration table entry of this type will be referenced as physical and will not be fixed up when transition from preboot to runtime phase.

**Description**

The following list shows the GUIDs of known conformance profiles. This list is not exhaustive and does not show GUIDs for all possible profiles. Additional profiles can be defined and published in other specifications.

#define EFI\_CONFORMANCE\_PROFILES\_UEFI\_SPEC\_GUID \

{ 0x523c91af, 0xa195, 0x4382, \

{ 0x81, 0x8d, 0x29, 0x5f, 0xe4, 0x00, 0x64, 0x65 }}

Conformance profile defined by this specification, as defined in section 2.6.

#define EFI\_CONFORMANCE\_PROFILES\_EBBR\_2\_0\_GUID \

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

Conformance profile defined by the EBBR 2.0.x specification. For more information, see “Links to UEFI-Related Documents” (<http://uefi.org/uefi>) under the heading “EBBR Specification”

**# Additional Instructions**

Add link to <http://uefi.org/uefi>

|  |  |
| --- | --- |
| EBBR Specificaiton | <https://github.com/arm-software/ebbr> |