TRUSTED® COMPUTING GROUP

SPECIFICATION

PCIe-based Component Class Registry

Version 1.0
Revision 18
October 27, 2021

Contact: admin@trustedcomputinggroup.org

PUBLISHED

DISCLAIMERS, NOTICES, AND LICENSE TERMS

CHANGE HISTORY

REVISION	DATE	DESCRIPTION
Version 1.0, Revision 18	October 27, 2021	• Initial Release of Version 1.0.

Acknowledgements

The TCG wishes to thank those members and others who contributed to this specification. This document builds on considerable work done in the various working groups in the TCG.

Special thanks to the members of the IWG group and others contributing to this document:

Name	**Affiliation**
Kathir Nadarajah	Advanced Micro Devices, Inc.
Monty Wiseman (IWG co-chair)	Beyond Identity
Amy Nelson	Dell, Inc.
Jason Young	Dell, Inc.
Andreas Fuchs	Fraunhofer Institute for Secure Information Technology (SIT)
Ludovic Jacquin[1] (co-editor)	Hewlett Packard Enterprise
Tom Laffey (IWG co-chair)	Hewlett Packard Enterprise
Ken Goldman	IBM
Arkadiusz Berent	Intel Corporation
Eduardo Cabre	Intel Corporation
Ned Smith	Intel Corporation
David Safford	Invited expert
William Bellingrath	Juniper Networks, Inc.
Masoud Manoo	Lenovo (United States) INC
Ronald Aigner	Microsoft
Joe Pennisi	NVIDIA Corporation
Dick Wilkins	Phoenix Technologies Ltd.
Jim Hatfield	Seagate Technology
Andrew Medak (co-editor)	United States Government
Lawrence Reinert	United States Government

[1] Results incorporated in this standard received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883335.

CONTENTS

1 SCOPE

This specification describes an option for the translation of component identifiers within PCI and PCIe capabilities into a Platform Certificate that complies with the specification TCG Platform Certificate Profile version 1.2 or later [1]. This specification is a registry that specifies the values required in a Platform Certificate that claims usage of this registry.

1.1 Purpose

This specification defines the mapping of data from PCI and PCIe capabilities, specifically how the data is encoded within the Platform Certificate. This mapping enables scalable component identification via PCI and PCIe capabilities and verification using the Platform Certificate.

1.2 Key Words

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this document’s normative statements are to be interpreted as described in RFC-2119, Key words for use in RFCs to Indicate Requirement Levels.

1.3 Statement Type

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statements.

EXAMPLE: Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the TCG specification the user must read the specification. (This use of MUST does not require any action).

End of informative comment

1.4 References

[1] TCG Platform Certificate Profile Version 1.1 Revision 19 or later. Infrastructure Working Group, TCG. https://trustedcomputinggroup.org/resource/tcg-platform-certificate-profile/

[2] TCG PC Client Specific Platform Firmware Profile Specification. PC Client Working Group, TCG. https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/

[3] PCI Express Base Specification Revision 5.0, Version 1.0. PCI-SIG. https://members.pcisig.com/wg/PCI-SIG/document/13005

[4] PCI Express Base Specification Revision 4.0, Version 1.0. PCI-SIG. https://members.pcisig.com/wg/PCI-SIG/document/10912

[5] PCI Express Base Specification Revision 3.0. PCI-SIG. https://members.pcisig.com/wg/PCI-SIG/document/download/8265

[6] Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID), IEEE. https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/tutorials/eui.pdf

[7] PCI Code and ID Assignment Specification, Revision 1.9 or later. PCI-SIG. https://pcisig.com/pci-code-and-id-assignment-specification-agreement

2 Definitions

2.1.1 Platform Certificate `componentIdentifier` field

Start of informative comment

A Platform Certificate's **`componentIdentifier`** field is defined in [1].

Each **`componentIdentifier`** in a Platform Certificate represents an individual component in the platform. The **`componentIdentifier`** field is encoded as an ASN.1 SEQUENCE. Multiple fields inside the sequence record data regarding the individual component.

This specification documents the translation of PCIe components' data and encapsulation of that data within the **`componentIdentifier`** field.

There are other registries than that specified in this specification. This specification does not specify that a Platform Certificate issuer has to use the registry specified in this specification. The specific registry used to translate and encapsulate a specific PCIe component's information is chosen on a component-by-component basis by the Platform Certificate issuer.

End of informative comment

2.1.1.1 The `componentClassRegistry` field

Start of informative comment

This specification does not specify the inclusion of any optional field within the **`componentIdentifier`**.

If a Platform Certificate issuer wants to encode PCIe information differently from the recommendation of this specification, they should use the TCG Component Class Registry, identified by the OID **`tcg-registry-componentClass-tcg`**, which is specified in [1].

End of informative comment

The **`OBJECT IDENTIFIER tcg-registry-componentClass-pcie`** SHALL be used in the **`componentClassRegistry`** field to identify the registry defined by this specification.

If the registry defined by this specification is identified via the **`componentClassRegistry`** field of a **`componentIdentifier`**, the encoding defined in this specification SHALL be used for the specified **`componentIdentifier`**'s fields.

2.1.1.1.1 ASN.1 definition of `tcg-registry-componentClass-pcie`

Start of informative comment

`tcg-registry-componentClass-pcie` is a TCG-defined OID that is part of the OID namespace inherited from TCPA specifications and [1]. For the convenience of the Reader, the OIDs used by this specification are:

```
-- TCG specific OIDs

tcg OBJECT IDENTIFIER ::= {

   joint-iso-itu-t(2) international-organizations(23) tcg(133) }

tcg-registry OBJECT IDENTIFIER ::= {tcg 18}


-- TCG Registry OIDs

tcg-registry-componentClass OBJECT IDENTIFIER ::= {tcg-registry 3}
```

End of informative comment

```
tcg-registry-componentClass-pcie OBJECT IDENTIFIER ::= {
                                       tcg-registry-componentClass 4 }
```

2.1.2 Notation

Start of informative comment

PCIe capabilities and registers may contain information that is relevant to a `componentIdentifier`. This section defines notation used later in this specification to read data from PCIe capabilities and registers, and to format that data into a `componentIdentifier`.

This specification uses the “0x” prefix for hexadecimal notation. Any number without the prefix should be interpreted as decimal.

Refer to [6] for endianness of PCIe Configuration Space values. The Reader is reminded that PCIe Configuration Space values assume a little-endian ordering convention, unless explicitly specified otherwise. When the registry defined by this specification is used, PCIe register values are encoded using their big-endian hexadecimal representation.

Users of this specification, particularly implementers of a verifier, need to be aware of the different endianness used in TCG specifications related to PCIe components. For example, the TCG PC Client Specific Platform Firmware Profile Specification [2] does not specify endianness for PCIe device context in measurement, allowing little-endian to be used.

When a value or field does not exist or is not returned, an empty UTF8 string is used with the delimiters.

End of informative comment

The UTF8 encoding characters of a value MUST be big-endian hexadecimal representation and MUST use uppercase letters.

2.1.2.1 Concatenation

Start of informative comment

This specification uses different PCIe registers that can be included in the same UTF8String; the vertical pipe character (“|”) is used to denote concatenation of two strings.

End of informative comment

The notation “|” means that the UTF8String value MUST concatenate the strings on either side of the “|” character.

2.1.2.2 Literal String

Start of informative comment

This specification uses colons (“:”) to delimit different PCIe registers that are included in the same UTF8String.

End of informative comment

When a cell from column content in Table 1 contains a string inside quotation marks, that string MUST be used.

2.1.2.3 Val(*register*)

Start of informative comment

When encoding a register’s value in a UTF8String, the big-endian hexadecimal representation does not include any prefix or suffix. For example, the “0x” prefix is not included in the UTF8String encoding. The UTF8String uses uppercase letters.

End of informative comment

When a cell from the Content column in Table 1 contains the notation Val(*register*), the UTF8 characters encoding the value of *register* MUST be used.

2.1.2.4 VPD_str(*keyword*)

Start of informative comment

When encoding a Vital Product Data ASCII string, a NULL character is not added at the end of the UTF8 encoding.

End of informative comment

When a cell from column content in Table 1 contains the notation VPD_str(*keyword*), the UTF8 characters encoding the ASCII characters of the Vital Product Data's (VPD) data field for *keyword* MUST be used when the VPD capability exists and *keyword* is present.

2.1.2.5 EUI_64(*register*)

Start of informative comment

This specification uses the IEEE 64-bit Extended Unique Identifier (EUI-64) defined in [6]. A EUI-64 is encoded in a UTF8String using the base-16 form defined in [6]: the octet array 0xACDE48234567019F is represented by the string "ACDE48234567019F".

End of informative comment

When a cell from the Content column in Table 1 contains the notation EUI_64(*register*), the UTF8 characters encoding the base-16 form of the value of *register* MUST be used.

3 Translation of PCIe information into a `componentIdentifier` field

Start of informative comment

PCIe does not contain data that is relevant to the **`componentManufacturerId`**, **`fieldReplaceable`**, **`componentAddresses`**, **`componentPlatformCert`**, **`componentPlatformCertUri`**, and **`status`** fields of **`componentIdentifier`**. This specification does not provide any recommendation for the content of those fields.

Table 1 specifies the content of the **`componentManufacturer`**, **`componentModel`**, **`componentSerial`** and **`componentRevision`** fields of **`componentIdentifier`**. Those fields are encoded as UTF8String in a TCG Platform Certificate and their content is derived from different PCIe registers or extended capabilities:

- The *Class Code Register* is defined in sections 7.5.1.1 and 7.5.1.1.6 of the PCIe specification version 5.0 [2] or 4.0 [3]. For PCIe version 3.0 [4], *Class Code Register* is defined in sections 7.5.1, 7.5.2 and 7.5.3. The *Class Code Register* is a 24 bit value, which is converted into an OCTET STRING. For PCIe version 4.0 [3] or 5.0 [2]: the second octet of the **`componentClassValue`** contains the ***Base Class Code***, the third octet of the **`componentClassValue`** contains the ***Sub-Class Code*** and the fourth octet of the **`componentClassValue`** contains the ***Programming Interface*** of the *Class Code Register*.
- The *Device ID Register*, *Vendor ID Register* and *Revision ID Register* are defined in section 7.5.1.1 of the PCIe specification version 5.0 [2] or 4.0 [4]. For PCIe version 3.0 [5], they are defined in section 7.5.1. The *Device ID Register* and *Vendor ID Register* are 16 bit values, which are formatted as four UTF8 characters encoding the hexadecimal representation of their value. The *Revision ID Register* is an 8 bit value, which is formatted as two UTF8 characters encoding the hexadecimal representation of its value.
- The *Subsystem ID Register* and *Subsystem Vendor ID Register* are defined in section 7.5.1.2 of the PCIe specification version 5.0 [2] or 4.0 [4]. For PCIe version 3.0 [5], they are defined in section 7.5.2. The *Subsystem ID Register* and *Subsystem Vendor ID Register* are 16 bit values, which are formatted as four UTF8 characters encoding the hexadecimal representation of their value.
- The *MN* (Manufacturer), *PN* (Part Number) and *SN* (Serial Number) keywords of the VPD are defined in sections 6.28 and 7.9.19 of the PCIe specification version 5.0 [2] or 4.0 [4]. VPD does not exist in PCIe version 3.0 [5], thus no character is added to the UTF8String.
- The *Serial Number Register* is defined in section 7.9.3 of the PCIe specification version 5.0 [2] or 4.0 [4]. For PCIe version 3.0 [5], *Serial Number Register* is defined in section 7.12. For PCIe version 5.0 or 4.0, the ordering of octet in the Serial Number Register is different than the one presented in the IEEE guideline and used in this document.

End of informative comment

Table 1: Component Identity Information (normative)

`componentIdentifier`'s field	ASN.1 Encoding	Content
		Content and format that the field MUST use
`componentClassValue`	OCTET STRING SIZE(4)	0x00 | Val(*Class Code Register*)
`componentManufacturer`	UTF8String	Val(*Vendor ID Register*) | ":" | Val(*Subsystem Vendor ID Register*) | ":" | VPD_str(*MN*)
`componentModel`	UTF8String	Val(*Device ID Register*) | ":" | Val(*Subsystem ID Register*) | ":" | VPD_str(*PN*)
`componentSerial`	UTF8String	EUI_64(*Serial Number Register*) | ":" | VPD_str(*SN*)
`componentRevision`	UTF8String	Val(*Revision ID Register*)

If the *Class Code Register* value indicates a Network Controller as defined in [6], the **`componentAddresses`**, if included in the **`componentIdentifier`**, MUST be given a **`ComponentAddress`** entry that contains the Vendor Assigned MAC address from one of the Base Address Registers and a relevant **`AddressType`**. The **`addressValue`** SHALL use the uppercase hexadecimal representation of the MAC address, without delimiters.

Appendix A: Examples

A.1 Sample `PlatformConfiguration`

Start of informative comment

Sample encoded **`PlatformConfiguration`** sequence showing only **`componentIdentifiers`**:

```
SEQUENCE (1 elem)
  [0] (4 elem)
    SEQUENCE (5 elem)
      SEQUENCE (2 elem)
        OBJECT IDENTIFIER 2.23.133.18.3.4
        OCTET STRING (4 byte) 00010802
      UTF8String 0104:036F:Sample VPD Manufacture ID 1
      UTF8String 3467:24A7:Sample VPD Part Number 1
      [0] (43 byte) 9964DD12785FA237:Sample VPD Serial Number 1
      [1] (2 byte) 78
    SEQUENCE (5 elem)
      SEQUENCE (2 elem)
        OBJECT IDENTIFIER 2.23.133.18.3.4
        OCTET STRING (4 byte) 00060400
      UTF8String 0A79::
      UTF8String 3CEB::
      [0] (17 byte) 2FBDA6FFFF543107:
      [1] (2 byte) 00
    SEQUENCE (6 elem)
      SEQUENCE (2 elem)
        OBJECT IDENTIFIER 2.23.133.18.3.4
        OCTET STRING (4 byte) 00020000
      UTF8String 6D5F:4182:
      UTF8String 2B4E:8697:Sample VPD Part Number 2
      [0] (27 byte) :Sample VPD Serial Number 2
      [1] (2 byte) 22
      [4] (1 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.23.133.17.1
          UTF8String 112233445566
    SEQUENCE (4 elem)
      SEQUENCE (2 elem)
        OBJECT IDENTIFIER 2.23.133.18.3.4
        OCTET STRING (4 byte) 000C03FE
      UTF8String E528:069A:
      UTF8String C71D:43BF:
      [1] (2 byte) 00
```

End of informative comment

A.2 Sample PCIe Configuration Space data

Start of informative comment

The following is sample data that was used to inform the encoding of section A.1. The data blobs are base64 encoded.

End of informative comment

A.2.1 Component Sample 1

A.2.1.1 Configuration Space

Start of informative comment

This sample represents a Type 0 Header with Vital Product Data presented in section A.2.1.2 and the Device Serial Number Capability.

```
BAFnNAcEEAB4AggBAAAAAAAAAN4MAADAAAAAAAwAANAAAAAAAeAAAAAAAABvA6ckAAAA32AAAAAA
AAAA/wEAAAMA4IB4eHh4AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwABADeiX3gS3WSZAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==
```

End of informative comment

A.2.1.2 Vital Product Data

Start of informative comment

```
ghQAU2FtcGxlIFBDSSBDb21wb25lbnSQjQBQThdTYW1wbGUgVlBEIFBhcnQgTnVtYmVyIDFNThlT
YW1wbGUgVlBEIE1hbnVmYWN0dXJlIElEIDFTThhTYW1wbGUgVlBEIFNlcmlhbCBOdW1iZXIgMVYw
KVNhbXBsZSBFeHRyYSBEYXRhIGluIFZQRCAgICAgICAgICAgICAgICAgVjIETi9BIFJWAS+RRwBW
MRhTYW1wbGUgUlcgRGF0YSBpbiBWUEQgICBZQQNOL0FZQhB4eHh4eHh4eHh4eHh4eHh4WUMQeHh4
eHh4eHh4eHh4eHh4eHg=
```

End of informative comment

A.2.2 Component Sample 2

Start of informative comment

This Configuration Space sample represents a Type 1 Header. This sample does not include subsystem information.

```
eQrrPAcEEAAAAAQGAACBAAAAAAAAAAAAAAAAAAAAANAAAAAAAAAAAEAAAABAAAAAAAAAAAEAAAAA
AAAA/wECAAAAAAAAAAAAAAAAAAAAAAABAAAAAQAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAREAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAOAAAAAwAf
AAAAAAAAAAAACQAUAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMAAQAHMVT//6a9LwAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==
```

End of informative comment

A.2.3 Component Sample 3

A.2.3.1 Configuration Space

Start of informative comment

This sample represents a Type 0 Header for a Network Controller, with Vital Product Data presented in section A.2.3.2. The MAC address is vendor-supplied and retrieved by firmware.

```
X21OKwcEEAAiAAACAAAAAAAAAN4MAADAAAAAAAwAANAAAAAAAeAAAAAAAACCQZeGAAAA32AAAAAA
AAAA/wEAAAAAAAAAAAAAAAAAAAAAAAABAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAFeIEABCDg/gAA
AAAnQAAAEAASAOGNLAEwKQAAAz1FAEAAAREAAAAAAAAAAAAAAAAAAAAAEwgEAAAEAAAOAAAAAwAf
AAAAAAAAAAAACQAUAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADAOCA
eHh4eAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==
```

End of informative comment

A.2.3.2 Vital Product Data

Start of informative comment

ghQAU2FtcGxlIFBDSSBDb21wb25lbnSQbwBQThdTYW1wbGUgVlBEIFBhcnQgTnVtYmVyIDJTThhT
YW1wbGUgVlBEIFNlcmlhbCBOdW1iZXIgMlYwKVNhbXBsZSBFeHRyYSBEYXRhIGluIFZQRCAgICAg
ICAgICAgICAgICAgVjIETi9BIFJWAQSRRwBWMRhTYW1wbGUgUlcgRGF0YSBpbiBWUEQgICBZQQNO
L0FZQhB4eHh4eHh4eHh4eHh4eHh4WUMQeHh4eHh4eHh4eHh4eHg=

End of informative comment

A.2.4 Component Sample 4

Start of informative comment

This Configuration Space sample represents a Type 0 Header without Vital Product Data.

KOUdxwcEEAAA/gMMAAAAAAAAAN4MAADAAAAAAAwAANAAAAAAAeAAAAAAAACaBr9DAAAAAAAAAAAA
AAAA/wEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==

End of informative comment