

# **Storage Component Class Registry**

---

Version 1.0  
Revision 22  
January 26, 2024

Contact: [admin@trustedcomputinggroup.org](mailto:admin@trustedcomputinggroup.org)

PUBLISHED

## DISCLAIMERS, NOTICES, AND LICENSE TERMS

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification and to the implementation of this specification, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this specification or any information herein.

This document is copyrighted by Trusted Computing Group (TCG), and no license, express or implied, is granted herein other than as follows: You may not copy or reproduce the document or distribute it to others without written permission from TCG, except that you may freely do so for the purposes of (a) examining or implementing TCG specifications or (b) developing, testing, or promoting information technology standards and best practices, so long as you distribute the document with these disclaimers, notices, and license terms.

Contact the Trusted Computing Group at [www.trustedcomputinggroup.org](http://www.trustedcomputinggroup.org) for information on specification licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

## CHANGE HISTORY

REVISION	DATE	DESCRIPTION
Version 1.0, Revision 16	June 27, 2022	Release of Version 1.0's draft for public review.
Version 1.0, Revision 21	September 20, 2023	Release of Version 1.0's second draft for public review.
Version 1.0, Revision 22	January 26, 2024	Publication 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
Alex Tzonkov	Advanced Micro Devices, Inc.
Gongyuan Zhuang	Advanced Micro Devices, Inc.
Jen ye	Advanced Micro Devices, Inc.
Robert Strong	Advanced Micro Devices, Inc.
Adonay Behre	AMI
Frederick Otumfuor	AMI
Stuart Yoder	ARM Ltd.
Monty Wiseman (IWG co-chair)	Beyond Identity
Bill Sulzen	Cisco Systems
Doug Ambrisko	Cisco Systems
Jeff Hewett	Cisco Systems
Scott Phuong	Cisco Systems
Tom Brostrom	Cyber Pack Ventures
Amy Nelson	Dell, Inc.
Jason Young	Dell, Inc.
Nick Grobelny	Dell, Inc.
Henk Birkholz	Fraunhofer Institute for Secure Information Technology (SIT)
Brandon Weeks	Google Inc.
Jason Young	Google Inc.
Jiankun Lu	Google Inc.
Theo Koulouris	Hewlett Packard Enterprise
Tom Laffey (IWG co-chair)	Hewlett Packard Enterprise
Adrian Shaw	HP Inc.
Joshua Schiffman	HP Inc.
Ken Goldman	IBM
Andreas Fuchs	Infineon Technologies
Ga-Wai Chin	Infineon Technologies
Joerg Borchert	Infineon Technologies
Ahmad Khalifeh	Intel Corporation
Arkadiusz Berent	Intel Corporation

Name	Affiliation
Avani Dave	Intel Corporation
Eduardo Cabre	Intel Corporation
Ned Smith	Intel Corporation
Thomas Bowen	Intel Corporation
Carolin Baumgartner	Invited expert
David Challener	Invited expert
David Safford	Invited expert
Graeme Proudler	Invited expert
Scott Kelly	Invited expert
Guy Fedorkow	Juniper Networks, Inc.
William Bellingrath	Juniper Networks, Inc.
Bill Keown	Lenovo (United States) INC
Masoud Manoo	Lenovo (United States) INC
Brad Litterell	Microsoft
Erich McMillan	Microsoft
Ronald Aigner	Microsoft
Thomas Hardjono	MIT Connection Science
Toru Tomita	NEC Corporation
Dan Morav	Nuvoton Technology Corporation
Joe Pennisi	NVIDIA Corporation
Ludovic Jacquin (co-editor)	NVIDIA Corporation
Steven Bellock	NVIDIA Corporation
Dick Wilkins	Phoenix Technologies Ltd.
Fernando Tavares	Positivo Tecnologia S.A.
Eric Hibbard	Samsung Semiconductor Inc.
Jim Hatfield	Seagate Technology
Santosh Kumar	Solidigm
Fabien Arrivé	STMicroelectronics
Joe Pilozzi	STMicroelectronics
Eoin Carroll	TOYOTA Motor Corporation
Andrew Medak (co-editor)	United States Government
Jerry Vacek	United States Government
Jessica Fitzgerald-McKay	United States Government
Lawrence Reinert	United States Government

<b>Name</b>	<b>Affiliation</b>
Zachary Blum	United States Government

## CONTENTS

DISCLAIMERS, NOTICES, AND LICENSE TERMS .....	2
CHANGE HISTORY .....	3
1 SCOPE .....	9
1.1 Purpose .....	9
1.2 Key Words .....	9
1.3 Statement Type .....	9
2 Definitions .....	10
2.1 Platform Certificate componentIdentifier field.....	10
2.1.1 The componentClassRegistry field .....	10
2.1.2 ASN.1 definition of tcg-registry-componentClass-storage .....	10
2.2 Notation .....	11
2.2.1 Concatenation .....	11
2.2.2 Literal Value .....	11
2.2.3 Literal String .....	12
2.2.4 ATA_FormFactor() .....	12
2.2.5 ATA_AOI() .....	12
2.2.6 ATA_UNIQUEID() .....	12
2.2.7 ATA_String( <i>field</i> ).....	12
2.2.8 SPC_INQUIRY_Class() .....	12
2.2.9 SPC_INQUIRY_String( <i>field</i> ) .....	13
2.2.10 SPC_VPD_SN_String( <i>field</i> ) .....	13
2.2.11 SPC_VPD_DI_UNIQUEID_String().....	13
2.2.12 PCIe_CC() .....	14
2.2.13 NVMe_Val(Capability or Feature) .....	14
2.2.14 NVMe_String(Capability or Feature) .....	15
2.2.15 NVMe_OUI(Capability or Feature) .....	15
3 Translation of ATA information into a componentIdentifier field .....	16
4 Translation of SCSI information into a componentIdentifier field .....	18
5 Translation of NVMe information into a componentIdentifier field.....	20
6 References.....	23
Appendix A: Examples .....	24
A.1 Sample PlatformConfiguration .....	24
A.2 Sample ATA Log data .....	25
A.2.1 ATA Component Sample 1 .....	25

A.3 Sample SCSI data.....	26
A.3.1  SCSI Component Sample 1 .....	26
A.3.2  SCSI Component Sample 2 .....	26
A.4 Sample NVMe Identify Controller data .....	27
A.4.1  NVMe Component Sample 1 .....	27

# 1 SCOPE

This specification is a registry that maps component identifiers within ATA [3], SCSI [4] and NVM Express (NVMe®) [5] capabilities into a Platform Certificate that complies with the specification TCG Platform Certificate Profile version 1.1 or later [1] and 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 ATA, SCSI and NVMe capabilities, specifically how the data is encoded within the Platform Certificate. This mapping in this specification enables scalable component identification via ATA, SCSI and NVMe 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 statement.

### 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

## 2 Definitions

### 2.1 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. The fields inside the sequence record data about the individual component.

This specification documents the mapping of ATA [3], SCSI [4] and NVMe [5] components' data and encapsulation of that mapped data within the `componentIdentifier` field.

There are other registries than that specified in this specification, such as the PCIe-based Component Class Registry [10]. This specification does not specify that a Platform Certificate issuer is required to use the registry specified in this specification. The specific registry used to translate and encapsulate a specific ATA, SCSI or NVMe component's information is chosen on a component-by-component basis by the Platform Certificate issuer.

#### End of informative comment

#### 2.1.1 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 storage 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-storage` 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.

#### 2.1.2 ASN.1 definition of `tcg-registry-componentClass-storage`

#### Start of informative comment

`tcg-registry-componentClass-storage` 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-storage OBJECT IDENTIFIER ::= {
    tcg-registry-componentClass 5 }
```

## 2.2 Notation

### Start of informative comment

ATA, SCSI and NVMe capabilities and fields may contain information that is relevant to a **componentIdentifier**. This section defines notation used later in this specification to read data from ATA, SCSI and NVMe capabilities and fields, 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 [3] for the alignment of ATA strings. The Reader is reminded that ATA swaps each pair of bytes in a string.

Refer to [4] for the alignment of SCSI strings. The Reader is reminded that SCSI uses both left-aligned and right-aligned ASCII strings.

Refer to [5] for endianness of NVMe values. The Reader is reminded that NVMe values assume a little-endian ordering convention, unless explicitly specified otherwise. When the registry defined by this specification is used, NVMe values are encoded using their big-endian hexadecimal representation.

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, which include NVMe devices. 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 SHALL be the big-endian hexadecimal representation and SHALL use uppercase letters.

The UTF8 empty string SHALL be a string of zero length, which is a byte array composed of only the NUL character.

### 2.2.1 Concatenation

#### Start of informative comment

This specification uses different parts 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 value on either side of the vertical pipe character SHALL be concatenated.

### 2.2.2 Literal Value

#### Start of informative comment

This specification uses big-endian values that are encoded in byte arrays. Each literal value needs to be encoded on a single byte.

#### End of informative comment

When a cell from column Content in Table 1, Table 2 or Table 3 contains the hexadecimal representation of a value, that value SHALL be used, encoded in a single byte.

## 2.2.3 Literal String

### Start of informative comment

This specification uses colons (":") to delimit different fields that are included in the same UTF8String.

### End of informative comment

When a cell from column Content in Table 1, Table 2 or Table 3 contains a string inside quotation marks, that string SHALL be used.

## 2.2.4 ATA\_FormFactor()

### Start of informative comment

The ATA IDENTIFY DEVICE data log contains the *NOMINAL FORM FACTOR* field, which indicates the dimensional or connector form factor of the device. The *NOMINAL FORM FACTOR* field's value is encoded in one byte in a Platform Certificate using this specification.

### End of informative comment

The notation ATA\_FormFactor() in Table 1 SHALL be substituted with the value of NOMINAL FORM FACTOR field of the IDENTITY DEVICE data log.

## 2.2.5 ATA\_AOI()

### Start of informative comment

The ATA IDENTIFY DEVICE data log contains the *IEEE AOI* field inside the *WORLD WIDE NAME* field.

This specification uses the IEEE 24-bit Administered Organizational Identifier (AOI) defined in [9]. An AOI is encoded in a UTF8String using the base-16 form defined in [3]: the octet array 0xACDE48234567019F is represented by the string "ACDE48234567019F".

### End of informative comment

The notation ATA\_AOI() in Table 1 SHALL be substituted with the UTF8 characters encoding the AOI base-16 form of the *IEEE AOI* field of the IDENTITY DEVICE data log.

## 2.2.6 ATA\_UNIQUEID()

### Start of informative comment

The ATA IDENTIFY DEVICE data log contains the *UNIQUE ID* field inside the *WORLD WIDE NAME* field.

### End of informative comment

The notation ATA\_UNIQUEID() in Table 1 SHALL be substituted with the UTF8 characters encoding the *UNIQUE ID* field of the IDENTITY DEVICE data log.

## 2.2.7 ATA\_String(*field*)

### Start of informative comment

Some *fields* of the ATA IDENTIFY DEVICE data log are already encoded in ASCII printable characters (code values between 0x20 and 0x7E). When encoding such a *field*, all ASCII padding space characters (0x20) are omitted.

### End of informative comment

The notation ATA\_String(*field*) in Table 1 SHALL be substituted with the IDENTITY DEVICE data log's *field* ASCII characters, excluding the padding ASCII space characters.

## 2.2.8 SPC\_INQUIRY\_Class()

### Start of informative comment

The SCSI Primary Commands (SPC) standard INQUIRY data contains the *PERIPHERAL QUALIFIER* and *PERIPHERAL DEVICE TYPE* fields, which identify the logical unit. The *PERIPHERAL QUALIFIER* and *PERIPHERAL DEVICE TYPE* fields are encoded together in a Platform Certificate using this specification.

#### **End of informative comment**

The notation SPC\_INQUIRY\_Class() in Table 2 SHALL be substituted with the value of Byte 0 of the standard INQUIRY data.

### **2.2.9 SPC\_INQUIRY\_String(*field*)**

#### **Start of informative comment**

Some SPC standard INQUIRY data's *fields* are already encoded in ASCII printable characters (code values between 0x20 and 0x7E) with potentially one or more ASCII null character (0x00). Those *fields* are left-aligned as specified in [4]. When encoding such a *field*, all ASCII characters are included in the Platform Certificate.

#### **End of informative comment**

The notation SPC\_INQUIRY\_String(*field*) in Table 2 SHALL be substituted with the standard INQUIRY data *field*'s ASCII characters.

### **2.2.10 SPC\_VPD\_SN\_String(*field*)**

#### **Start of informative comment**

Some SCSI Vital Product Data (VPD) tables' *fields* are already encoded in ASCII printable characters (code values between 0x20 and 0x7E). Those *fields* are right-aligned as specified in [4]. When encoding such a *field*, all ASCII characters are included in the Platform Certificate.

#### **End of informative comment**

The notation SPC\_VPD\_SN\_String(*field*) in Table 2 SHALL be substituted with the Unit Serial Number VPD page *field*'s ASCII characters.

### **2.2.11 SPC\_VPD\_DI\_UNIQUEID\_String()**

#### **Start of informative comment**

The SCSI Device Identification VPD page, defined in Section 7.7.6 Device Identification VPD page of [4], can contain unique identifiers in some of its *DESIGNATOR* fields. The *DESIGNATOR* field to use, if there is a choice of fields, is determined based on the algorithm detailed below. A Platform Certificate issuer or Verifier that follows this specification needs to find the appropriate designation descriptor in the list contained in the SCSI Device Identification VPD page.

The T10 vendor ID based *designator descriptor*, whose *DESIGNATOR TYPE* is 1, is defined in Section 7.7.6.4 T10 vendor ID based designator format of [4]. The *VENDOR SPECIFIC IDENTIFIER* field is a variable length.

The EUI-64 based *designator descriptor*, whose *DESIGNATOR TYPE* is 2, is defined in Section 7.7.6.5 EUI-64 based designator format of [4]. Regardless of the EUI-64 length and format, the *VENDOR SPECIFIC EXTENSION IDENTIFIER* field is a 40-bit numeric value.

The Network Address Authority (NAA) based *designator descriptor*, whose *DESIGNATOR TYPE* is 3, is defined in Section 7.7.6.6 NAA designator format of [4]. For an IEEE Registered Extended designator (whose NAA field is 6), the *VENDOR SPECIFIC IDENTIFIER* field concatenated with the *VENDOR SPECIFIC EXTENSION IDENTIFIER* field is a 100-bit numeric value. For an IEEE Registered designator (whose NAA field is 5), the *VENDOR SPECIFIC IDENTIFIER* field is a 36-bit numeric value. For an IEEE Extended designator (whose NAA field is 3), the *VENDOR SPECIFIC IDENTIFIER A* field concatenated with the *VENDOR SPECIFIC IDENTIFIER B* field is a 36-bit numeric value. For a Locally Assigned designator (whose NAA field is 2), the *LOCALLY ADMINISTERED VALUE* field is a 60-bit numeric value.

#### **End of informative comment**

The notation SPC\_VPD\_DI\_UNIQUEID\_String() in Table 2 SHALL be substituted with the UTF8 characters specified by the method below.

Only the SCSI device's *Designator descriptors* with an ASSOCIATION field equal to 0 SHALL be used.

The *Designator descriptors* SHALL be used in the order defined below:

1. If the *Designator descriptor* has a DESIGNATOR TYPE field equal to 3, the *Designator descriptor* SHALL be used, in the order defined below:
  - a. If a *Designator descriptor* has a DESIGNATOR TYPE field equal to 3 and a NAA field equal to 6, the UTF8 characters encoding the value of the VENDOR SPECIFIC IDENTIFIER field concatenated with the VENDOR SPECIFIC EXTENSION IDENTIFIER field SHALL be used, else
  - b. If a *Designator descriptor* has a DESIGNATOR TYPE field equal to 3 and a NAA field equal to 5, the UTF8 characters encoding the value of the VENDOR SPECIFIC IDENTIFIER field SHALL be used, else
  - c. If a *Designator descriptor* has a DESIGNATOR TYPE field equal to 3 and a NAA field equal to 2, the UTF8 characters encoding the value of the VENDOR SPECIFIC IDENTIFIER A field concatenated with the VENDOR SPECIFIC IDENTIFIER B field SHALL be used, else
  - d. If a *Designator descriptor* has a DESIGNATOR TYPE field equal to 3 and a NAA field equal to 3, the UTF8 characters encoding the value of the LOCALLY ADMINISTERED VALUE field SHALL be used, else
  - e. If a *Designator descriptor* has a DESIGNATOR TYPE field equal to 3 and a NAA field equal to any other value, an empty UTF8 string SHALL be used.
2. If the *Designator descriptor* has a DESIGNATOR TYPE field equal to 2, the UTF8 characters encoding the value of the VENDOR SPECIFIC EXTENSION IDENTIFIER field SHALL be used.
3. If the *Designator descriptor* has a DESIGNATOR TYPE field equal to 1, the VENDOR SPECIFIC IDENTIFIER field's ASCII characters SHALL be used.
4. Else, an empty UTF8 string SHALL be used.

## 2.2.12 PCIe\_CC()

### Start of informative comment

When encoding the PCIe® Class Code Register's value in an OCTET STRING, the 24 bits big-endian representation is used.

### End of informative comment

The notation PCIe\_CC() in Table 3 SHALL be substituted with the big-endian representation of the PCIe Class Code register value. If the PCIe Class Code register does not exist for the component, the value 0x000000 SHALL be used.

## 2.2.13 NVMe\_Val(Capability or Feature)

### Start of informative comment

When encoding an NVMe Controller or Namespace Structure (CNS) Capability or Feature'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

The notation NVMe\_Val(Capability or Feature) in Table 3 SHALL be substituted with the UTF8 characters encoding the value of register.

## 2.2.14 NVMe\_String(*Capability or Feature*)

### Start of informative comment

Some NVMe CNS *Capabilities or Features* are already encoded in ASCII printable characters (code values between 0x20 and 0x7E). Those ASCII strings are left justified as specified in [5]. When encoding such ASCII string, the trailing ASCII space characters (0x20) are omitted.

### End of informative comment

The notation NVMe\_String(*Capability or Feature*) in Table 3 SHALL be substituted with the CNS *Capability or Feature*'s ASCII printable characters, excluding the trailing ASCII space characters.

## 2.2.15 NVMe\_OUI(*Capability or Feature*)

### Start of informative comment

This specification uses the IEEE 24-bit OUI defined in [9]. An OUI is encoded in a UTF8String using the base-16 form defined in [9]: the octet array 0xACDE48234567019F is represented by the string “ACDE48234567019F”.

The Reader is reminded that the OUI is in little endian format in the Identify Controller Data Structure of NVMe.

### End of informative comment

The notation NVMe\_OUI(*Capability or Feature*) in Table 3 SHALL be substituted with the UTF8 characters encoding the OUI base-16 form of *Capability or Feature*.

### 3 Translation of ATA information into a `componentIdentifier` field

#### Start of informative comment

ATA 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 `componentClassValue`, `componentManufacturer`, `componentModel`, `componentSerial` and `componentRevision` fields of `componentIdentifier`. The `componentClassValue` field is encoded as 4 octets, while the remaining fields are encoded as UTF8String in a TCG Platform Certificate and their content is derived from different ATA fields of the IDENTIFY DEVICE data log:

- The *NOMINAL FORM FACTOR* field (page 0x03, offset bytes 32 to 39) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.5.5 Form Factor (NOMINAL FORM FACTOR field) of [3]. It is a 4-bit value that is gathered by `ATA_FormFactor()`, which is the fourth octet of the `componentClassValue`.
- The *IEEE AOI* field (page 0x03, offset bits 516 to 539) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.5.8 World Wide Name of [3]. It is a 24-bit AOI value assigned by the IEEE Registration Authority, which composes the `componentManufacturer` field.
- The *MODEL NUMBER* field (page 0x05, offset bytes 48 to 87) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.7.4 MODEL NUMBER field of [3]. It is a 40-byte ASCII string that composes the `componentModel` field, since UTF8 is backward compatible with ASCII.
- The *SERIAL NUMBER* field (page 0x05, offset bytes 8 to 27) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.7.2 SERIAL NUMBER field of [3]. It is a 20-byte ASCII string that partially composes the `componentSerial` field, since UTF8 is backward compatible with ASCII.
- The *UNIQUE ID* field (page 0x03, offset bits 540 to 575) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.5.8 World Wide Name of [3]. It is a 36-bit value assigned by the device manufacturer that partially composes the `componentSerial` field.
- The *FIRMWARE REVISION* field (page 0x05, offset bytes 32 to 39) of the ATA IDENTIFY DEVICE data log (Log Address 0x30) is defined in Section 9.10.7.3 FIRMWARE REVISION field of [3]. It is an 8-byte ASCII string that composes the `componentRevision` field, since UTF8 is backward compatible with ASCII.

#### End of informative comment

The **componentIdentifier** structure SHALL be populated with values obtained from the methods specified in Table 1's Content column for an ATA-compliant component. The values SHALL be encoded as specified in Table 1.

**Table 1: Component Identity Information for ATA (normative)**

Field name	ASN.1 Encoding	Content
<b>componentClassValue</b>	OCTET STRING SIZE(4)	0x00   0x00   0x00   ATA_FormFactor()
<b>componentManufacturer</b>	UTF8String	ATA_AOI()
<b>componentModel</b>	UTF8String	ATA_String( <i>MODEL NUMBER</i> )
<b>componentSerial</b>	UTF8String	ATA_String( <i>SERIAL NUMBER</i> )   ":"   ATA_UNIQUEID()
<b>componentRevision</b>	UTF8String	ATA_String( <i>FIRMWARE REVISION</i> )

## 4 Translation of SCSI information into a `componentIdentifier` field

### Start of informative comment

The SPC standard 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 2 specifies the content of the `componentClassValue`, `componentManufacturer`, `componentModel`, `componentSerial` and `componentRevision` fields of `componentIdentifier`. The `componentClassValue` field is encoded as 4 octets, while the remaining fields are encoded as UTF8String in a TCG Platform Certificate, and their content is derived from different SPC fields of the standard INQUIRY data, the Unit Serial Number VPD page, or the Device Identification VPD page:

- The *PERIPHERAL QUALIFIER* and *PERIPHERAL DEVICE TYPE* fields (byte 0) of the SPC standard INQUIRY data are defined in Section 6.7.2 Standard INQUIRY data of [4]. Together, they form an 8-bit value that is gathered by `SPC_INQUIRY_Class()`, which is the fourth octet of the `componentClassValue`.
- The *T10 VENDOR IDENTIFICATION* field (bytes 8 to 15) of the SPC standard INQUIRY data is defined in Section 6.7.2 Standard INQUIRY data of [4]. It is an 8-byte left-aligned ASCII string that composes the `componentManufacturer` field, since UTF8 is backward compatible with ASCII.
- The *PRODUCT IDENTIFICATION* field (bytes 16 to 31) of the SPC standard INQUIRY data is defined in Section 6.7.2 Standard INQUIRY data of [4]. It is a 16-byte left-aligned ASCII string that composes the `componentModel` field, since UTF8 is backward compatible with ASCII.
- The *DESIGNATOR* field (starting at byte 4) of the SPC Device Identification VPD page (page code 0x83) is defined in Section 7.7.6 Device Identification VPD page of [4]. Some data of the *DESIGNATOR* field can be used to partially compose the `componentSerial` field: this is detailed in Section 2.2.11.
- The *PRODUCT SERIAL NUMBER* field (starting at byte 4) of the SPC Unit Serial Number VPD page (page code 0x80) is defined in Section 7.7.19 Unit Serial Number VPD page of [4]. When present, it is a variable length right-aligned ASCII string that partially composes the `componentSerial` field, since UTF8 is backward compatible with ASCII.
- The *PRODUCT REVISION LEVEL* field (bytes 32 to 35) of the SPC standard INQUIRY data is defined in Section 6.7.2 Standard INQUIRY data of [4]. A firmware update of the SCSI device could change the value of this field. It is a 4-byte left-aligned ASCII string that composes the `componentRevision` field, since UTF8 is backward compatible with ASCII.

### End of informative comment

The **componentIdentifier** structure SHALL be populated with values obtained from the methods specified in Table 2's Content column for a SCSI-compliant component. The values SHALL be encoded as specified in Table 2.

**Table 2: Component Identity Information for SCSI (normative)**

Field name	ASN.1 Encoding	Content
<b>componentClassValue</b>	OCTET STRING SIZE(4)	0x01   0x00   0x00   SPC_INQUIRY_Class()
<b>componentManufacturer</b>	UTF8String	SPC_INQUIRY_String( <i>T10 VENDOR IDENTIFICATION</i> )
<b>componentModel</b>	UTF8String	SPC_INQUIRY_String( <i>PRODUCT IDENTIFICATION</i> )
<b>componentSerial</b>	UTF8String	SPC_VPD_DI_UNIQUEID_String()   ":"   SPC_VPD_SN_String( <i>PRODUCT SERIAL NUMBER</i> )
<b>componentRevision</b>	UTF8String	SPC_INQUIRY_String( <i>PRODUCT REVISION LEVEL</i> )

## 5 Translation of NVMe information into a `componentIdentifier` field

### Start of informative comment

The NVMe identity data structures (e.g., CNS) do 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 3 specifies the content of the `componentClassValue`, `componentManufacturer`, `componentModel`, `componentSerial` and `componentRevision` fields of `componentIdentifier`. The `componentClassValue` field is encoded as 4 octets, while the remaining fields are encoded as UTF8String in a TCG Platform Certificate and their content is derived from different Controller Capabilities or Features of the CNS:

- The *Class Code Register* is defined in sections 7.5.1.1 Type 0/1 Common Configuration Space, and 7.5.1.1.6 Class Code Register (Offset 09h) of the PCIe specification version 5.0 [6] or 4.0 [7]. For PCIe version 3.0 [8], *Class Code Register* is defined in sections 7.5.1 Type 0/1 Common Configuration Space, 7.5.2 Type 0 Configuration Space Header, and 7.5.3 Type 1 Configuration Space Header. The *Class Code Register* is a 24-bit value, which is converted into an OCTET STRING. For PCIe version 4.0 or 5.0: 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 *PCI Vendor ID (VID)* and *PCI Subsystem Vendor ID (SSVID)* are defined in Sections 4.3.1 PCI Vendor ID (VID) and PCI Subsystem Vendor ID (SSVID), and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. They are part of the Identify Controller Data Structure (CNS = 0x01), VID at bytes 0x00 to 0x01 and SSVID at bytes 0x02 to 0x03. Each is a 16-bit value, assigned by the PCI SIG, that partially forms the `componentManufacturer` field.
- The *IEEE OUI Identifier (IEEE)* is defined in Sections 4.3.3 IEEE OUI Identifier (IEEE), and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x73 to 0x75. It is an OUI 24-bit value assigned by the IEEE Registration Authority that partially forms the `componentManufacturer` field.
- The *Model Number (MN)* is defined in Sections 4.3.2 Serial Number (SN) and Model Number (MN), and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x24 to 0x63. It is a 40-byte left justified (big endian) ASCII string assigned by the component's vendor that forms the `componentModel` field.
- The *Field Replaceable Unit (FRU) Globally Unique Identifier (FGUID)* is defined in Sections 4.3.4 IEEE Extended Unique Identifier (EUI64), and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x112 to 0x127. It is a 16-byte big endian value containing a EUI64 and an 8 bytes extension identifier assigned by the vendor that partially forms the `componentSerial` field. The OUI Identifier within the EUI64 is in big endian.
- The *Serial Number (SN)* is defined in Sections 4.3.2 Serial Number (SN) and Model Number (MN), and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x04 to 0x23. It is a 20-byte left justified (big endian) ASCII string assigned by the component's vendor that partially forms the `componentSerial` field.
- The *Version (VER)* is defined in Sections 3.1.3.2 Offset 8h: VS – Version, and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x80 to 0x83.

It is a 4-byte value containing the major, minor and tertiary version of the NVMe base specification that forms the **componentRevision** field.

- The *Firmware Revision (FR)* is defined in Sections 5.16.1.4 Firmware Slot Information (Log Identifier 03h), and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x64 to 0x71. It is an 8-byte ASCII string.
- The NVM Subsystem NVMe Qualified Name (SUBNQN) is defined in Sections 4.5 NVMe Qualified Names, and 5.17.2.1 Identify Controller Data Structure (CNS 01h) of [5]. It is part of the Identify Controller Data Structure (CNS = 0x01), at bytes 0x300 to 0x3FF. It is a UTF-8 string that partially composes the **componentSerial** field.

**End of informative comment**

The **componentIdentifier** structure SHALL be populated with values obtained from the methods specified in Table 3's Content column for an NVMe-compliant component. The values SHALL be encoded as specified in Table 3.

**Table 3: Component Identity Information for NVMe (normative)**

Field name	ASN.1 Encoding	Content
<b>componentClassValue</b>	OCTET STRING SIZE(4)	0x02   PCIe_CC()
<b>componentManufacturer</b>	UTF8String	NVMe_Val(VID)   ":"   NVMe_Val(SSVID)   ":"   NVMe_OUI(IEEE)
<b>componentModel</b>	UTF8String	NVMe_String(MN)
<b>componentSerial</b>	UTF8String	NVMe_Val(FGUID)   ":"   NVMe_String(SM)   ":"   NVMe_String(SUBNQN)
<b>componentRevision</b>	UTF8String	NVMe_Val(VER)   ":"   NVMe_String (FR)

## 6 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, version 1.05 or later. PC Client Working Group, TCG. <https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/>
- [3] Information Technology - ATA Command Set - 5 (ACS-5), INCITS 558-2021, ANSI.  
<https://webstore.ansi.org/Standards/INCITS/INCITS5582021>
- [4] Information Technology - SCSI Primary Commands - 5 (SPC-5), INCITS 502-2019, ANSI.  
<https://webstore.ansi.org/Standards/INCITS/INCITS5022019>
- [5] NVM Express® Base Specification Revision 2.0b, NVM Express, Inc.  
<https://nvmexpress.org/wp-content/uploads/NVM-Express-Base-Specification-2.0b-2021.12.18-Ratified.pdf>
- [6] PCI Express® Base Specification Revision 5.0, Version 1.0. PCI-SIG.  
<https://members.pcisig.com/wg/PCI-SIG/document/13005>
- [7] PCI Express® Base Specification Revision 4.0, Version 1.0. PCI-SIG.  
<https://members.pcisig.com/wg/PCI-SIG/document/10912>
- [8] PCI Express® Base Specification Revision 3.0. PCI-SIG.  
<https://members.pcisig.com/wg/PCI-SIG/document/download/8265>
- [9] 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>
- [10] PCIe-based Component Class Registry Version 1.0 Revision 18 or later. Infrastructure Working Group, TCG.  
<https://trustedcomputinggroup.org/resource/pcie-based-component-class-registry/>

## Appendix A: Examples

### A.1 Sample PlatformConfiguration

#### Start of informative comment

This sample **PlatformConfiguration** sequence only shows the **componentIdentifiers** sequence. The first component's information in the **componentIdentifiers** sequence is based on the ATA Component Sample 1 presented in section A.2.1. The second component's information is based on the SCSI Component Sample 1 presented in section A.3.1. The third component's information is based on the SCSI Component Sample 2 presented in section A.3.2. The fourth component's information is based on the NVMe Component Sample 1 presented in section A.4.1.

```

SEQUENCE (1 elem)
[0] (3 elem)
    SEQUENCE (5 elem)
        SEQUENCE (2 elem)
            OBJECT IDENTIFIER 2.23.133.18.3.5
            OCTET STRING (4 byte) 00000002
            UTF8String ABCDEF
            UTF8String Sample Model Number 1
            [0] (25 byte) Sample Serial 1:112233445
            [1] (8 byte) FW Ver 1
    SEQUENCE (5 elem)
        SEQUENCE (2 elem)
            OBJECT IDENTIFIER 2.23.133.18.3.5
            OCTET STRING (4 byte) 0100007F
            UTF8String EXAMPLE
            UTF8String SCSI Model 1
            [0] (28 byte) :Sample SCSI Serial Number 1
            [1] (4 byte) SV A
    SEQUENCE (5 elem)
        SEQUENCE (2 elem)
            OBJECT IDENTIFIER 2.23.133.18.3.5
            OCTET STRING (4 byte) 0100007F
            UTF8String EXAMPLE
            UTF8String SCSI Model 2
            [0] (43 byte) 123456789ABCDEF:Sample SCSI Serial Number 2
            [1] (4 byte) SV B
    SEQUENCE (5 elem)
        SEQUENCE (2 elem)
            OBJECT IDENTIFIER 2.23.133.18.3.5
            OCTET STRING (4 byte) 02010802
            UTF8String ABCD:1234:ABCDEF
            UTF8String M2
            [0] (89 byte) 1122334455667788ABCDEF99AABBCCDD:SN1:nqn.2014-
08.com.example:nvme:nvm-subsystem-sn-d78432
            [1] (3 byte) FW3

```

#### End of informative comment

## A.2 Sample ATA Log 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 ATA Component Sample 1

##### Supported Capabilities (page 0x03)

### Start of informative comment

```
gAAAAAAADAAGAAAAAAAAAAAAAAgAAAAAABAAAAAAgAAAAACAAAAAAA  
Fq83vESIzRFgAAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAA  
AAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAA  
AAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAA  
AAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAA  
AAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAA  
AAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAA  
AAA=
```

### End of informative comment

##### Strings (page 0x05)

### Start of informative comment

```
gAAAAAAFAFTYW1wbGUgU2VyaWFsIDEgICAgIAAAABGVyBWZXIgMQAAAAAAA  
WJ1ciAxICAgICAgICAgICAgICAgICAgIAAAABGVyBWZXIgMQAAAAAAA  
AAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAA  
AAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAA  
AAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAA  
AAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAA  
AAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAAgAAAAAABAAAAAA  
AAA=
```

### End of informative comment

## A.3 Sample SCSI 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.3.1 SCSI Component Sample 1

##### SCSI INQUIRY Data

###### Start of informative comment

```
fwAHAAAAAABFWFNUExFIFNDU0kgTW9kZWwgMSAgICBTViBBAAAAAAAAAAAAAAAAAABSGFiMXZ
wxAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA=
```

###### End of informative comment

##### SCSI Unit Serial Number VPD (page 0x80)

###### Start of informative comment

```
f4AAG1NhbXBsZSBTQ1NJIFNlcmlhbCB0dW1iZXIgMQ==
```

###### End of informative comment

#### A.3.2 SCSI Component Sample 2

##### SCSI INQUIRY Data

###### Start of informative comment

```
fwAHAAAAAABFWFNUExFIFNDU0kgTW9kZWwgMiAgICBTViBCAAAAAAAAAAAAAABSGFiMXZ
wxAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA=
```

###### End of informative comment

##### SCSI Unit Serial Number VPD (page 0x80)

###### Start of informative comment

```
f4AAG1NhbXBsZSBTQ1NJIFNlcmlhbCB0dW1iZXIgMg==
```

###### End of informative comment

##### SCSI Device Identification VPD (page 0x83)

###### Start of informative comment

```
f4MALgIBAB5FWEFNUExFIFNhbXBsZSBTZXJpYWwgTnVtYmVyIDMBAwAIMSNFZ4mrze8=
```

###### End of informative comment

#### A.4 Sample NVMe Identify Controller data

#### A.4.1 NVMe Component Sample 1

## **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.

This sample uses the PCIe Type 0 Header from Component Sample 1 in Section A.2 of [5].

#### **End of informative comment**

## Identify Controller Data Structure

## **Start of informative comment**

#### **End of informative comment**