TRUSTED® COMPUTING GROUP

SPECIFICATION

SMBIOS-based Component Class Registry

Version 1
Revision 00
October 29, 2020

Contact: admin@trustedcomputinggroup.org

PUBLIC REVIEW

Work in Progress

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

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.

Contact the Trusted Computing Group at 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
1.00/1.00	October 29, 2020	• Initial Release of Version 1.00.

CONTENTS

1 SCOPE

This specification describes an option for the translation of component identifiers within SMBIOS into a Platform Certificate that complies with the specification TCG Platform Certificate Profile version 1.1 or later [1]. This specification is a registry that specifies the values required in a Platform Certificate which claims usage of this registry.

1.1 Purpose

This specification defines the mapping of data from SMBIOS for a finite set of components, specifically how the data is encoded within the Platform Certificate. This mapping enables scalable component identification via SMBIOS structures and verification using the Platform Certificate content.

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.

https://trustedcomputinggroup.org/resource/tcg-platform-certificate-profile/

[2] System Management BIOS (SMBIOS) Reference Specification Version 3.4.0 Document Identifier: DSP0134.

https://www.dmtf.org/standards/smbios

2 Definitions

2.1.1 Platform Certificate `componentIdentifier` field

Start of informative comment

The **`componentIdentifier`** field is defined in [1].

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

This registry documents translation of SMBIOS data and encapsulation of that data within the **`componentIdentifier`** field.

Each component listed in the Platform Certificate can use its own registry; selection of the registry is done on a per-component basis by the Platform Certificate issuer.

End of informative comment

2.1.1.1 The componentClassRegistry field

The **`OBJECT IDENTIFIER`** that SHALL be used in the **`componentClassRegistry`** field to identify this registry is **`tcg-registry-componentClass-dmtf`**.

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

Start of informative comment

This registry does not present guidance on the inclusion of any optional field within the **`componentIdentifier`**.

If a Platform Certificate issuer wants to encode SMBIOS information differently from the recommendation of this specification, they should use the TCG Component Class Registry, identified by the OID **`tcg-registry-componentClass-tcg`**.

The **`tcg-registry-componentClass-dmtf`** and **`tcg-registry-componentClass-tcg`** are documented in [1].

End of informative comment

2.1.2 Notation

Start of informative comment

SMBIOS may contain information that is relevant to a **`componentIdentifier`**. This section defines notation used later in this document to read data from SMBIOS, and to format that data into a **`componentIdentifier`**.

SMBIOS data and methods to access SMBIOS data are defined in [2]. The notation used in this document is not defined in [2].

Section "A.3 Example uses of Notation on Sample SMBIOS Data" provides examples for using the notation defined in this section.

This document uses the "0x" prefix for hexadecimal notation. Any number without the prefix should be interpreted as decimal.

Refer to "Typographical conventions" in [2] for the hexadecimal form of SMBIOS values. For convenience, the reader is reminded that the SMBIOS specification notation for hexadecimal format uses the suffix 'h'.

End of informative comment

2.1.2.1 Literal Value

A cell in Table 1 of section 3 “Translation of SMBIOS information into a componentIdentifier” that contains only a decimal or hexadecimal number, with no other notation, means the literal value MUST be placed into the corresponding field of the **`componentIdentifier`**.

2.1.2.2 “N/A”

Start of informative comment

A cell in Table 1 that contains the string N/A means SMBIOS does not contain data relevant to that field. This document does not provide any recommendation for the corresponding field of the **`componentIdentifier`**.

End of informative comment

2.1.2.3 Value @ <OFFSET> [| <OFFSET> …]

A cell in Table 1 where the column is defined to use ASN.1 Encoding of type UTF8String and contains the notation Value @ <OFFSET> means the hexadecimal representation of the Value found at byte <OFFSET> within the SMBIOS table SHALL be encoded as a UTF8String. That UTF8String MUST be placed into the corresponding field of the **`componentIdentifier`**, if that field is present.

Start of informative comment

In the case of a UTF8String encoding, the hexadecimal representation does not include any prefix or suffix. For example, “0x” is not included in the UTF8String encoding.

End of informative comment

A cell in Table 1 where the column is defined to use ASN.1 Encoding of type OCTET STRING and contains the notation Value @ <OFFSET> means the bytes that make up the Value found at byte <OFFSET> within the SMBIOS table SHALL be encoded as an OCTET STRING. That OCTET STRING MUST be placed into the corresponding field of the **`componentIdentifier`**, if that field is present.

The notation “|” means to concatenate multiple “Value @ <OFFSET>” within the same encoding.

Start of informative comment

The Length of the Value @ <OFFSET> is defined for each field within each SMBIOS structure. It could be, but is not limited to: BYTE (1); WORD (2); DWORD (4); QWORD (8). Refer to [2] to determine the Length. The bytes to be encoded for Value @ <OFFSET> are at <OFFSET> to <OFFSET>+Length-1 of the SMBIOS structure.

Notice that the Length of the Value @ <OFFSET> for a value of type STRING is NOT the Length of the STRING as the Value is a positional reference to a string at the end of the structure.

Refer to “Section 5.1 General” in [2] for endianness of SMBIOS values. For convenience, the reader is reminded that the SMBIOS structures assume a little-endian ordering convention, unless explicitly specified otherwise, i.e., multi-byte numbers (WORD, DWORD, etc.) are stored with the low-order byte at the lowest address and the high-order byte at the highest address.

The bytes to be encoded for Value @ <OFFSET1> | <OFFSET2> | <OFFSET3> are at <OFFSET1> to <OFFSET1>+Length1-1 concatenated with <OFFSET2> to <OFFSET2>+Length2-1 concatenated with <OFFSET3> to <OFFSET3>+Length3-1.

End of informative comment

2.1.2.4 Strref @ <OFFSET>

A cell in Table 1 that contains the notation Strref @ <OFFSET> means the string referenced by the Value found at byte <OFFSET> within the SMBIOS table MUST be placed into the corresponding field of the **`componentIdentifier`**, if that field is present.

In the event that no string is referenced by Strref @ <OFFSET>, the UTF8String encoding of a string of length 0 MUST be placed into the corresponding field of the **`componentIdentifier`**, if that field is present.

Start of informative comment

The Strref @ <OFFSET> notation will only ever be used on fields defined to use ASN.1 Encoding of type UTF8String.

When using the notation Strref @ <OFFSET>, the Value found at byte <OFFSET> could be 0x00. In this case, there is no string referenced.

End of informative comment

2.1.2.5 Bit Field @ <OFFSET> [& <MASK>] != <TEST_VALUE>

A cell in Table 1 that contains the notation Bit Field @ <OFFSET> != <TEST_VALUE> means the corresponding BOOLEAN field of the **`componentIdentifier`** MUST be encoded TRUE if the Value found at byte <OFFSET> within the SMBIOS table is not equal to <TEST_VALUE>. Otherwise, the corresponding BOOLEAN field of the **`componentIdentifier`** MUST be encoded FALSE.

A cell in Table 1 that contains the notation Bit Field @ <OFFSET> & <MASK> != <TEST_VALUE> means the corresponding BOOLEAN field of the **`componentIdentifier`** MUST be encoded TRUE if the result of performing bit-wise AND on <MASK> and the Value found at byte <OFFSET> within the SMBIOS table is not equal to <TEST_VALUE>. Otherwise, the corresponding BOOLEAN field of the **`componentIdentifier`** MUST be encoded FALSE.

Start of informative comment

Example: If the Value of the Bit Field at Offset <BYTE> is 0x3E and the Mask is 0x13, the result is 0x12.

Example: If the Value of the Bit Field at Offset <BYTE> is 0x1996 and the Mask is 0x0003, the result is 0x0002.

End of informative comment

3 Translation of SMBIOS information into a componentIdentifier

Start of informative comment

Table 1 defines notations as defined in section 2.1.2 Notation.

- The Field, ASN.1 Encoding, and Optional rows contain relevant field properties from [1].
- The **`componentManufacturerId`**, **`componentAddresses`**, **`componentPlatformCert`**, **`componentPlatformCertUri`**, and **`status`** fields of the **`componentIdentifier`** are N/A for all component types in Table 1. See 2.1.2.2 “N/A.

End of informative comment

Table 1: Translation Table (normative)

Field	componentClassValue				componentManufacturer	componentModel	componentSerial	componentRevision	fieldReplaceable
ASN.1 Encoding	OCTET STRING SIZE(4)				UTF8String	UTF8String	UTF8String	UTF8String	BOOLEAN
Optional	Mandatory				Mandatory	Mandatory	Optional	Optional	Optional
	MSB	2nd	3rd	LSB					
	SMBIOS Type								
Baseboard	0x00	0x02	0x00	Value @ 0x0D	Strref @ 0x04	Strref @ 0x05	Strref @ 0x07	Strref @ 0x06	Bit Field @ 0x09 & 0x1C != 0
BIOS	0x00	0x00	Value @ 0x12	Value @ 0x13	Strref @ 0x04	Strref @ 0x05	N/A	Value @ 0x14 | 0x15	N/A
Chassis	0x00	0x03	0x00	Value @ 0x05	Strref @ 0x04	Value @ 0x05	Strref @ 0x07	Strref @ 0x06	N/A
Processor	0x00	0x04	0x00	Value @ 0x05	Strref @ 0x07	Value @ 0x06	Strref @ 0x20	Strref @ 0x10	Bit Field @ 0x19 != 0x06
RAM	0x00	0x11	0x00	Value @ 0x12	Strref @ 0x17	Strref @ 0x1A	Strref @ 0x18	Strref @ 0x2B	N/A
System	0x00	0x01	0x00	0x00	Strref @ 0x04	Strref @ 0x05	Strref @ 0x07	Strref @ 0x06	N/A
Power Supply	0x00	0x27	0x00	0x00	Strref @ 0x07	Strref @ 0x0A	Strref @ 0x08	Strref @ 0x0B	Bit Field @ 0x0E & 0x0100 != 0
TPM	0x00	0x2B	0x00	0x00	Value @ 0x04 | 0x05 | 0x06 | 0x07	Value @ 0x08 | 0x09	N/A	Value @ 0x0A | 0x0E	N/A

Appendix A: Examples

A.1 Sample `PlatformConfiguration`

Start of informative comment

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

```
SEQUENCE (1 elem)
  [0] (12 elem)
    SEQUENCE (4 elem)
      SEQUENCE (2 elem)
        OBJECT IDENTIFIER 2.23.13318.3.3
        OCTET STRING (4 byte) 00003307
      UTF8String Sample BIOS Vendor
      UTF8String Q1.09.xyz
      [1] (4 byte) 080F
    SEQUENCE (5 elem)
      SEQUENCE (2 elem)
        OBJECT IDENTIFIER 2.23.13318.3.3
        OCTET STRING (4 byte) 00010000
      UTF8String Sample System Manufacturer
      UTF8String Sample System Model
      [0] (27 byte) Sample System Serial Number
      [1] (22 byte) Sample System Revision
    SEQUENCE (6 elem)
      SEQUENCE (2 elem)
        OBJECT IDENTIFIER 2.23.13318.3.3
        OCTET STRING (4 byte) 0002000A
      UTF8String Sample Baseboard Manufacturer
      UTF8String Sample Baseboard Model
      [0] (30 byte) Sample Baseboard Serial Number
      [1] (25 byte) Sample Baseboard Revision
      [3] (1 byte) FF
    SEQUENCE (5 elem)
      SEQUENCE (2 elem)
        OBJECT IDENTIFIER 2.23.13318.3.3
        OCTET STRING (4 byte) 00030003
      UTF8String Sample Chassis Manufacturer
      UTF8String 03
      [0] (28 byte) Sample Chassis Serial Number
      [1] (23 byte) Sample Chassis Revision
    SEQUENCE (6 elem)
      SEQUENCE (2 elem)
        OBJECT IDENTIFIER 2.23.13318.3.3
        OCTET STRING (4 byte) 00040003
      UTF8String Sample Processor Manufacturer
      UTF8String C6
      [0] (30 byte) Sample Processor Serial Number
      [1] (25 byte) Sample Processor Revision
      [3] (1 byte) FF
    SEQUENCE (6 elem)
      SEQUENCE (2 elem)
        OBJECT IDENTIFIER 2.23.13318.3.3
        OCTET STRING (4 byte) 00040003
      UTF8String Sample CPU Manufacturer
```



```
        UTF8String EF
        [0] (24 byte) Sample CPU Serial Number
        [1] (19 byte) Sample CPU Revision
        [3] (1 byte) 00
      SEQUENCE (5 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.23.13318.3.3
          OCTET STRING (4 byte) 00110018
        UTF8String Sample RAM Manufacturer
        UTF8String Sample RAM Model
        [0] (26 byte) Sample RAM Serial Number 1
        [1] (19 byte) Sample RAM Revision
      SEQUENCE (5 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.23.13318.3.3
          OCTET STRING (4 byte) 00110018
        UTF8String Sample RAM Manufacturer
        UTF8String
        [0] (26 byte) Sample RAM Serial Number 2
        [1] (19 byte) Sample RAM Revision
      SEQUENCE (5 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.23.13318.3.3
          OCTET STRING (4 byte) 00110018
        UTF8String Sample RAM Manufacturer
        UTF8String Sample RAM Model
        [0] (26 byte) Sample RAM Serial Number 3
        [1] (19 byte) Sample RAM Revision
      SEQUENCE (5 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.23.13318.3.3
          OCTET STRING (4 byte) 00110018
        UTF8String Sample RAM Manufacturer
        UTF8String Sample RAM Model
        [0] (26 byte) Sample RAM Serial Number 4
        [1] (19 byte) Sample RAM Revision
      SEQUENCE (6 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.23.13318.3.3
          OCTET STRING (4 byte) 00270000
        UTF8String Sample Power Supply Manufacturer
        UTF8String Sample Power Supply Model Part Number
        [0] (33 byte) Sample Power Supply Serial Number
        [1] (34 byte) Sample Power Supply Revision Level
        [3] (1 byte) 00
      SEQUENCE (4 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.23.13318.3.3
          OCTET STRING (4 byte) 002B0000
        UTF8String 41424300
        UTF8String 0200
        [1] (16 byte) 0102030405060708
```

End of informative comment

A.2 Sample SMBIOS data

Start of informative comment

The following is sample SMBIOS data that was used to inform the encoding above. The data blob is base64 encoded.


```
X1NNM1/cGAMCAAEAVAkAABgAAAAAAAAAABgAAAECAPADD4CYi38BAAAAMwcID///U2FtcGxlIEJp
b3MgVmVuZG9yAFExLjA5Lnh5egAwOS8wMS8yMDIwAAABGwEAAQIDBAACAAMABAAFAAYABwAIAAkG
BQZTYW1wbGUgU3lzdGVtIE1hbnVmYWN0dXJlcgBTYW1wbGUgU3lzdGVtIE1vZGVsAFNhbXBsZSBT
eXN0ZW0gUmV2aXNpb24AU2FtcGxlIFN5c3RlbSBTZXJpYWwgTnVtYmVyAFRvIEJlIEZpbGxlZCBC
eSBPLkUuTS4AVG8gQmUgRmlsbGVkIEJ5IE8uRS5NLgAAAg8CAAECAwQFCQYDAAoAU2FtcGxlIEJh
c2Vib2FyZCBNYW51ZmFjdHVyZXIAU2FtcGxlIEJhc2Vib2FyZCBNb2RlbABTYW1wbGUgQmFzZWJv
YXJkIFJldmlzaW9uAFNhbXBsZSBCYXNlYm9hcmQgU2VyaWFsIE51bWJlcgBUbyBCZSBGaWxsZWQg
QnkgTy5FLk0uAFRvIEJlIEZpbGxlZCBCeSBPLkUuTS4AAAMVAwABAwIDBAMDAwMAAAAAAAEAAFNh
bXBsZSBDaGFzc2lzIE1hbnVmYWN0dXJlcgBTYW1wbGUgQ2hhc3NpcyBSZXZpc2lvbgBTYW1wbGUg
Q2hhc3NpcyBTZXJpYWwgTnVtYmVyAFRvIEJlIEZpbGxlZCBCeSBPLkUuTS4AAAQoBAABA8YCpQYB
AP/7678DAIUAagpABkEBBQAGAAcABAUGBAQIBABDUFVTb2NrZXQAU2FtcGxlIFByb2Nlc3NvciBN
YW51ZmFjdHVyZXIAU2FtcGxlIFByb2Nlc3NvciBSZXZpc2lvbgBTYW1wbGUgUHJvY2Vzc29yIFNl
cmlhbCBOdW1iZXIAVG8gQmUgRmlsbGVkIEJ5IE8uRS5NLgBUbyBCZSBGaWxsZWQgQnkgTy5FLk0u
AAAEKAQAAQPvAqUGAQD/++u/AwCFAGoKQAZBBgUABgAHAAQFBgQECAQAQ1BVU29ja2V0AFNhbXBs
ZSBDUFUgTWFudWZhY3R1cmVyAFNhbXBsZSBDUFUgUmV2aXNpb24AU2FtcGxlIENQVSBTZXJpYWwg
TnVtYmVyAFRvIEJlIEZpbGxlZCBCeSBPLkUuTS4AVG8gQmUgRmlsbGVkIEJ5IE8uRS5NLgAABxMF
AAGAAAABAAEBAAEAAAQDBUwxLUNhY2hlAAAHEwYAAYEAAAQABAEAAQAABQUHTDItQ2FjaGUAAAcT
BwABggEAIAAgAQABAAAFBQhMMy1DYWNoZQAACQ0IAAEGBQMDAQAMAVBDSTEAAAkNCQABBgUDAwIA
DAFQQ0kyAAAJDQoAAaUIBAMRAAwBUENJRTEAAAkNCwABpQ0DAxIADAFQQ0lFMgAACQ0MAAGlCAMD
EwAMAVBDSUUzAAAJDQ0AAaUNBAMUAAwBUENJRTQAAAkNDgABpQoDAxUADAFQQ0lFNQAAEA8PAAMD
BgAAAAz+/wYAAAATDxAAAAAAAP//vwAPAAQAABFUEQAPAP7/SABAAAAICQABAhgCADUFAwQFBgAA
AAAAAAAAAAAAAAABAAAHAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAERJ
TU0wAEJBTkswAFNhbXBsZSBSQU0gTWFudWZhY3R1cmVyAFNhbXBsZSBSQU0gU2VyaWFsIE51bWJl
ciAxAEFzc2V0VGFnTnVtMABTYW1wbGUgUkFNIE1vZGVsAFNhbXBsZSBSQU0gUmV2aXNpb24AABQT
EgAAAAAA//8fABEAEAABAAEAABFUEwAPAP7/SABAAAAICQABAhgCADUFAwQFAAAAAAAAAAAAAAAA
AAABAAAHAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAERJTU0xAEJBTksx
AFNhbXBsZSBSQU0gTWFudWZhY3R1cmVyAFNhbXBsZSBSQU0gU2VyaWFsIE51bWJlciAyAEFzc2V0
VGFnTnVtMQBTYW1wbGUgUkFNIE1vZGVsAFNhbXBsZSBSQU0gUmV2aXNpb24AABQTFAAAACAA//8/
ABMAEAABAAEAABFUFQAPAP7/SABAAAAICQABAhgCADUFAwQFBgAAAAAAAAAAAAAAAAABAAAHAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAERJTU0yAEJBTksyAFNhbXBsZSBS
QU0gTWFudWZhY3R1cmVyAFNhbXBsZSBSQU0gU2VyaWFsIE51bWJlciAzAEFzc2V0VGFnTnVtMgBT
YW1wbGUgUkFNIE1vZGVsAFNhbXBsZSBSQU0gUmV2aXNpb24AABQTFgAAAEAA//9fABUAEAABAAEA
ABFUFwAPAP7/SABAAAAICQABAhgCADUFAwQFBgAAAAAAAAAAAAAAAAAABAAAHAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAERJTU0zAEJBTkszAFNhbXBsZSBSQU0gTWFudWZh
Y3R1cmVyAFNhbXBsZSBSQU0gU2VyaWFsIE51bWJlciA0AEFzc2V0VGFnTnVtMwBTYW1wbGUgUkFN
IE1vZGVsAFNhbXBsZSBSQU0gUmV2aXNpb24AABQTGAAAAGAA//9/ABcAEAABAAEAACAUHQAAAAAA
AAAAAAAAAAAAAAAAAnFiEAAAECAwQFBgeAABIJ////////U2FtcGxlIFBvd2VyIFN1cHBseSBM
b2NhdGlvbgBTYW1wbGUgUG93ZXIgU3VwcGx5IERldmljZSBOYW1lAFNhbXBsZSBQb3dlciBTdXBw
bHkgTWFudWZhY3R1cmVyAFNhbXBsZSBQb3dlciBTdXBwbHkgU2VyaWFsIE51bWJlcgBTYW1wbGUg
UG93ZXIgU3VwcGx5IEFzc2V0IFRhZyBOdW1iZXIAU2FtcGxlIFBvd2VyIFN1cHBseSBNb2RlbCBQ
YXJ0IE51bWJlcgBTYW1wbGUgUG93ZXIgU3VwcGx5IFJldmlzaW9uIExldmVsAAArHyMAQUJDAAIA
AQIDBAUGBwgBBAAAAAAAAAAAAAAAU2FtcGxlIFRQTSBEZXNjcmlwdGlvbgAAfwQlAAAA
```


End of informative comment

A.2.1 Extracting the Sample SMBIOS data in Linux

Start of informative comment

Extracting the Sample SMBIOS data in Linux:

- Requires the base64 and dmidecode packages

- Assumes the SMBIOS example provided in A.2 Sample SMBIOS data is available in file `SMBIOS.base64`

```
base64 -d SMBIOS.base64 > SMBIOS.bin
dmidecode --from-dump SMBIOS.bin -u
```

End of informative comment

A.3 Example uses of Notation on Sample SMBIOS Data

Start of informative comment

The examples in this section show the translation between the Sample SMBIOS data in section A.2 and the Sample **`PlatformConfiguration`** in section A.1.

End of informative comment

A.3.1 Literal Value

Start of informative comment

Example of using a Literal Value, based on the TPM Device included in the A.2 Sample SMBIOS data. The TPM Device is also included in a **`componentIdentifier`** in the A**.1** Sample `PlatformConfiguration`. Consistent with Table 1, the 4 bytes of the **`componentClassValue`** OCTET STRING are set to the literal values { 0x00, 0x2B, 0x00, 0x00 }.

End of informative comment

A.3.2 Value @ <OFFSET> [| <OFFSET> …]

Start of informative comment

Example of encoding a UTF8String using the notation Value @ <OFFSET>: the encoding of the **`componentManufacturer`** field within the TPM **`componentIdentifier`** in the A**.1** Sample `PlatformConfiguration`. Table 1 specifies that the UTF8String be set to the Value @ 0x04 | 0x05 | 0x06 | 0x07. Those offsets within the TPM Device structure of the A.2 Sample SMBIOS data contain the values { 0x41, 0x42, 0x43, 0x20 }. The result is the string “41424320”.

Example of encoding an OCTET STRING using the notation Value @ <OFFSET>: the encoding of the LSB of the **`componentClassValue`** field within the BIOS **`componentIdentifier`** in the A**.1** Sample `PlatformConfiguration`. Table 1 specifies it be set to the Value @ 0x13. That offset contains a value of Length BYTE, specifically { 0x07 }, within the A.2 Sample SMBIOS data. The result is the octet 0x07.

End of informative comment

A.3.3 Strref @ <OFFSET>

Start of informative comment

Example of encoding a UTF8String using the notation Strref @ <OFFSET>: the encoding of the **`componentManufacturer`** field within the Power Supply **`componentIdentifier`** in the A**.1** Sample `PlatformConfiguration`. Table 1 specifies the UTF8String be set to the Strref @ 0x07. That offset contains the value { 0x01 } within the A.2 Sample SMBIOS data. This means the result should be the first string at the end of the Power Supply structure. The result is the string “Sample Power Supply Manufacturer”.

Example of encoding a UTF8String using the notation Strref @ <OFFSET> when no string is referenced: the encodings of the **`componentModel`** fields within the RAM **`componentIdentifier`** in the A**.1** Sample `PlatformConfiguration`. One **`componentModel`** is encoded with a string of length 0 because the A.2 Sample SMBIOS data references no string at Strref @ 0x1A within that structure.

End of informative comment

A.3.4 Bit Field @ <OFFSET> [& <MASK>] != <TEST_VALUE>

Start of informative comment

Example 1 of encoding a BOOLEAN using the notation Bit Field @ <OFFSET> [& <MASK>] != <TEST_VALUE>: the encoding of the **`fieldReplaceable`** field within the Baseboard **`componentIdentifier`** in the **A.1** Sample `PlatformConfiguration`. Table 1 specifies the ASN1Boolean be set to the Bit Field @ 0x09 & 0x1C != 0. That offset contains the value of Length BYTE, specifically { 0x09 }, within the A.2 Sample SMBIOS data. 0x09 & 0x1C = 0x08. 0x08 is not equal to 0. The result is the ASN1Boolean being set to TRUE.

Example 2 of encoding a BOOLEAN using the notation Bit Field @ <OFFSET> [& <MASK>] != <TEST_VALUE>: the encoding of the **`fieldReplaceable`** field within the Power Supply **`componentIdentifier`** in the **A.1** Sample `PlatformConfiguration`. Table 1 specifies the ASN1Boolean be set to the Bit Field @ 0x0E & 0x0100 != 0. That offset contains a value of Length WORD, specifically { 0x12, 0x09 }, within the A.2 Sample SMBIOS data. 0x1209 & 0x0100 = 0. This fails the test. The result is the ASN1Boolean being set to FALSE.

End of informative comment

Appendix B: OS specific methods to access SMBIOS data

Start of informative comment

Any commands documented here are intended to provide assistance to anyone unfamiliar with accessing SMBIOS data. Raw SMBIOS data must be accessed for inclusion into the platform certificate. Tool-specific output may change the format of SMBIOS data and may not comply with this document.

End of informative comment

B.1 Linux

Start of informative comment

Accessing the SMBIOS raw data in Linux:

- Requires the dmidecode package

```
dmidecode --dump-bin FILE
```

End of informative comment

B.2 Windows PowerShell

Start of informative comment

Accessing the SMBIOS raw data in Windows:

- Requires Windows PowerShell

```
(Get-WmiObject -Class MSSMBios_RawSMBiosTables -Namespace root\wmi
 -ErrorAction SilentlyContinue).SMBiosData
```

End of informative comment

