Document Identifier: DSP0270

Date: 2020-08-04

Version: 1.3.0

Redfish Host Interface Specification

Supersedes: 1.2.0

Document Class: Normative

Document Status: Published

Document Language: en-US

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party's reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit http://www.dmtf.org/about/policies/disclosures.php.

This document's normative language is English. Translation into other languages is permitted.

CONTENTS

1 Foreword

The Redfish Host Interface Specification was prepared by the Redfish Forum of the DMTF.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.

1.1 Acknowledgments

The DMTF acknowledges the following individuals for their contributions to this document:

- Jeff Autor - Hewlett Packard Enterprise
- Jeff Bobzin - Insyde Software Corp
- Patrick Caporale - Lenovo
- Phil Chidester - Dell Inc.
- Chris Davenport - Hewlett Packard Enterprise
- Samer El-Haj-Mahmoud - Lenovo
- Jeff Hilland - Hewlett Packard Enterprise
- John Leung - Intel Corporation
- Edward Newman - Hewlett Packard Enterprise
- Michael Raineri - Dell Inc.
- Hemal Shah - Broadcom Limited
- Paul Vancil - Dell Inc.

2 Abstract

This specification defines functional requirements for Redfish Host Interfaces. In the context of this document, the term "Host Interface" refers to interfaces that can be used by software running on a computer system to access the Redfish Service that is used to manage that computer system.

The target audience for this specification is system manufacturers that are providing Redfish Host Interfaces within computer systems, system and component manufactures that are providing devices or firmware that include or support Redfish Host Interfaces, and system firmware and software writers that are creating software or firmware that uses Redfish Host Interfaces.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For references without a date or version, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

- DMTF DSP0134 System Management BIOS Reference Specification (SMBIOS), https://www.dmtf.org/standards/smbios
- UEFI Unified Extensible Firmware Interface Specifications (UEFI), http://www.uefi.org/specifications
- DMTF DSP0239 Management Component Transport Protocol (MCTP) IDs and Codes, Version 1.4, http://www.dmtf.org/sites/default/files/standards/documents/DSP0239_1.4.pdf
- DMTF DSP0266 Redfish Scalable Platforms Management API Specification, http://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.0.pdf
- ISO/IEC Directives, Part 2 Rules for the structure and drafting of International Standards, http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype
- PCI Firmware Specification, PCI SIG http://pcisig.com/specifications/conventional/pci_firmware
- IETF RFC4122, A Universally Unique IDentifier (UUID) URN Namespace, The Internet Society, July 2005, http://www.ietf.org/rfc/rfc4122.txt

4 Terms and definitions

In this document, some terms have a specific meaning beyond the normal English meaning. Those terms are defined in this clause.

The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"), "may", "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 7. The terms in parenthesis are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Clause 7 specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 6.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

The following additional terms are used in this document.

Term	Definition
Host	The Computer System that is managed by a Redfish Service
Host Software	The software running on the Host Computer System, including Operating System and its Software components (such as drivers or applications), as well as pre-boot software such as UEFI or BIOS drivers and applications.
Redfish Service	Also referred to as the "Service". The collection of functionality that implements the protocols, resources, and functions that deliver the interface defined by the Redfish API specification and its associated behaviors for one or more managed systems.
Redfish Service Entry Point	Also referred to as "Service Entry Point". The interface through which a particular instance of a Redfish Service is accessed. A Redfish Service may have more than one Service Entry Point.
Redfish Manager	Also referred to as "Manager". The entity that manages a Computer System and other peripherals through a Redfish Service.

5 Symbols and abbreviated terms

The following additional abbreviations are used in this document.

Term	Definition
BIOS	Basic I/O System. Name for system firmware typically used for initialization and launching the boot of an ISA (Industry Standard Architecture), aka 'x86' or 'PC', architecture-based computer system.
BSP	Board Support Package. Name for system firmware typically used for initialization and launching the boot of Linux in a computer system that uses a non-ISA architecture, but may be used for booting other types of operating systems or runtime software.
SMBIOS	System Management BIOS. Refers to DSP0134. Defines memory mapped tables, typically implemented by system firmware/BIOS and mapped into system firmware/BIOS memory space, that provide inventory and management information for the computer system.
UEFI	Unified Extensible Firmware Interface. A modern firmware standard that defines the interfaces between hardware and Operating Systems in a Computer System. UEFI is supported on multiple processor architectures, including x86, x64, ia64, and AARCH64.
HI	Host Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over TLS
IP	Internet Protocol
IPMI	Intelligent Platform Management Interface
NIC	Network Interface Card
PCI	Peripheral Component Interconnect
PCIe	PCI Express
TCP	Transmission Control Protocol
UUID	Universally Unique Identifier

6 Introduction

Redfish is a flexible system management tool that can be successfully applied to various system architectures. One important architecture consists of one or more CPUs assigned to the system application (the Host CPUs) and a separate CPU or CPUs assigned solely to management including publishing the Redfish interface. In many management schemes, it is necessary to provide standardized Redfish-based communication between the Host CPU and the Redfish service in the Management unit. This communication is in addition to the Redfish services available via the external network. Implementation of the Redfish Host Interface is optional for the system designer. If provided, this interface may be used in both the pre-boot (firmware) stage and by drivers and applications within the Host Operating System and is designed to be available without use of external networking. This specification provides design details for several methods of Host-to-Service communication. Additional methods may be added in future revisions of this specification.

6.1 Scope

This specification is targeted to system manufacturers that are providing Redfish Host Interfaces within computer systems, system and component manufactures that are providing devices or firmware that include or support Redfish Host Interfaces, and system firmware and software writers that are creating software or firmware that uses Redfish Host Interfaces.

The specification covers Host accessible physical and logical communication paths and protocols that are used to access the Redfish Service that manages that Host.

The specification also defines certain supporting elements in the Host, such as SMBIOS extensions, that enable inventory and discovery functions.

The specification does not seek to place specific hardware implementation requirements; however, it does in some cases specify how hardware-specific interfaces are identified for Host Software (e.g., SMBIOS structures).

The specification defines connectivity between a Redfish Service and a Host. Any network routing or other connectivity beyond the Redfish Service or other networks is out of scope.

6.2 Goals

The following are the goals for the Redfish Host Interface:

- Implementable with existing management controller technology
- Easily integrated into products
- Host Interface and out-of-band API must be the same (where possible) so that client apps have minimal (if any) change to adapt

- Support authentication, confidentiality, and integrity:
  - Support environments where users do not want to solely rely on Host/OS access control mechanisms
  - Provide mechanism to optionally (if configured) pass credentials to an OS Kernel for sensor monitoring (with configurable privilege)
- Support multi-service to multi-host architectures:
  - Blade system with Chassis Manager as well as sled management controller on each blade, each with a Host Interface
- Support security requirements with authentication and confidentiality

7 Protocol details

A Redfish Host Interface shall support one of the following protocols:

- Network HI -- Redfish HTTP requests/responses over a TCP/IP network connection between a Host and Redfish Service

7.1 Network Host Interface protocol details

Implementations that support the "Network Host Interface" protocol shall implement the following requirements:

- Implementations shall provide a TCP/IP network connection that route TCP/IP traffic between a Redfish client executing on the Host and the Redfish Service.
- Any link-level driver and interconnect that implements a TCP/IP network connection between a Host and a Redfish Service may be used. Example implementations include:
  - A USB Network Connection between a Host and a Redfish Service
  - A Host PCIe NIC that connects to a Manager NIC
  - A Host PCIe NIC that connects to a management LAN that connects to a Redfish Service
- Authentication, and privilege authorization equivalent to the out-of-band Redfish API as specified in DSP0266 shall be supported by the implementation when enabled from the Redfish Service configuration.
  - Authentication credentials that are valid on the normal out-of-band Redfish network interface shall also be valid on the HI.
  - Implementations may optionally support a configurable AuthNone authentication mode (no authentication required) that can be configured on the Redfish Service for use on the Host Interface. If implemented, enablement of AuthNone shall be configurable, and the RoleId assumed by AuthNone requests shall be configurable as described by the Redfish schema.
  - In addition to standard credentials, implementations may optionally support auto-generation and delivery of HI-only credentials that may be used by the Firmware or OS to authenticate.
  - If supported, auto-generated credentials for the Host shall be delivered by using UEFI-based mechanism described in a later section of this document.
  - The permissions granted to any auto-generated credentials shall be configurable with a defined RoleId assigned.
- Services shall require HTTPS encryption for the Network Host Interface with same requirements as via out-of-band network interfaces:
  - Session login POSTs shall use HTTPS

  - Patches that contain sensitive data shall use HTTPS
  - Basic Auth requests shall require HTTPS

- Implementations that support SMBIOS shall provide an SMBIOS Type 42 structure that describes each Host Interface as defined by the SMBIOS standard and the SMBIOS support clause of this document.

DEPRECATED

- Implementations that support automatically generating and sending credentials to the Host OS kernel and/or firmware using UEFI runtime variables shall be implemented as defined within the Delivery of credentials via UEFI runtime variables clause of this document.
  - If the Kernel Authentication Interface is implemented, the Redfish Service shall implement a configuration option that allows customers to disable the Kernel Authentication as described by the Redfish schema.
  - If the Kernel Authentication Interface is implemented, the Redfish Service shall implement a configurable role for the Kernel Authentication Interface as described by the Redfish schema.

DEPRECATED

---

8 SMBIOS support

Information in the SMBIOS structure shall allow Host Software to discover the Redfish Service Entry Point supported and to initialize the driver stack on the Host.

For Network Host Interfaces, the mechanism that clients should use to discover/obtain the Redfish Service Entry Point IP address shall also be described in the structure.

All SMBIOS structures referenced in this specification shall 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. Unless otherwise noted, strings in these structures follow "Text Strings" pattern described in the SMBIOS Specification.

8.1 SMBIOS Type 42 structure general layout

The SMBIOS Type 42 structure is used to describe a Management Controller Host Interface. It consists of standard SMBIOS entry information, followed by interface descriptors (which detail the physical interface to the Redfish Service), and protocol descriptors (which describe the supported payload encoding between the Host and Redfish Service). The following table shows the general format of the SMBIOS Type 42 structure:

```
--------------------------
Type 42 Header
--------------------------
Interface Specific Data
  - Device Description
  - (1 of 5 types)
--------------------------
  Protocol Record Header
--------------------------
  - Protocol Specific Data
--------------------------
```

Further details about the SMBIOS Type 42 structure can be found in the SMBIOS Specification.

The remaining sections document how the SMBIOS Type 42 structure is defined for use by the Redfish Host Interface.

8.2 Table 1: SMBIOS Type 42 structure definition for Redfish Host Interfaces

The following describes the SMBIOS Management Controller Host Interface (Type 42) structure. Offset 00h-04h is the Type 42 Header. Starting at Offset 05h is the Interface-specific Data.

Offset	Name	Length	Value	Description
00h	Type	BYTE	42	Management Controller Host Interface structure indicator
01h	Length	BYTE	Varies	Length of the structure, a minimum of 09h
02h	Handle	WORD	Varies	
04h	Interface Type	BYTE	ENUM	Management Controller Interface Type. Network Host Interface is 40h.
05h	Interface Specific Data Length (N)	BYTE	N	Number of bytes (N) in the Interface Specific Data field. If 0, there is no Interface specific data.
06h	Interface Specific Data	N BYTEs	Varies	Management Controller Host Interface Data as specified by the Interface Type. See Table 2 below for Network Host Interface definitions.
06h+N	Protocol Count	BYTE	X	X number of Protocol Records for this Host Interface Type (typically 1).
07h+N	Protocol Records	M BYTEs	Varies	Protocol Records for each protocol supported. See Table 10 below record format.

8.3 Table 2: Interface Specific Data (for Interface Type 40h)

Interface Specific Data starts at offset 06h of the SMBIOS Type 42 structure. This table defines the Interface Specific data for Interface Type 40h. There are several types of Device Descriptors defined (see Table 3); however, only one may be used in specific Type 42 table.

Offset	Name	Length	Value	Description
00h	Device Type	BYTE	ENUM	The Device Type of the interface. The value of this field determines how the Device Descriptor Data is decoded. Table 3.
01h	Device Descriptor Data	N-1 BYTEs	Varies	The Device Descriptor Data formatted based on Device Type. See Table 4, Table 5, Table 6, Table 7, and Table 8.

8.3.1 Table 3: Device Type values

The following table defines the possible values for the Device Type field found in Table 2.

Value	Description
02h	USB Network Interface. See Table 4.
03h	PCI/PCIe Network Interface. See Table 5.
04h	USB Network Interface v2. See Table 6.
05h	PCI/PCIe Network Interface v2. See Table 7.
80h-FFh	OEM. See Table 8.
Others	Reserved

8.3.2 Table 4: Device Descriptor Data for Device Type 02h (USB Network Interface)

The following table defines the Device Descriptor Data format for when the Device Type field is 02h, which indicates the device is a USB Network Interface.

Offset	Name	Length	Value	Description
00h	Vendor ID	WORD	Varies	The Vendor ID of the device, as read from the idVendor field of the USB descriptor (LSB first).
02h	Product ID	WORD	Varies	The Product ID of the device, as read from the idProduct field of the USB descriptor (LSB first).
04h	Serial Number Descriptor Length	BYTE	Varies	The number of bytes of the Serial Number, Serial Number Descriptor Type, and Serial Number Descriptor Length, as read from the iSerialNumber.bLength field of the USB descriptor. This field has a minimum value of 0x02.
05h	Serial Number Descriptor Type	BYTE	0x03	The Descriptor Type of the Serial Number, as read from the iSerialNumber.bDescriptorType field of the USB descriptor. This is always 0x03 to indicate that the descriptor is a string.
06h	Serial Number	Varies	Varies	The Serial Number of the device as a Unicode string without a NULL terminator, as read from the iSerialNumber.bString field of the USB descriptor. This string does not follow typical string patterns found elsewhere in SMBIOS definitions.

Examples type 02h descriptors:

- Vendor ID is 0xAABB, Product ID is 0xCCDD, and the Serial Number is "SN00001": `0xBB 0xAA 0xDD 0xCC 0x10 0x03 0x53 0x00 0x4E 0x00 0x30 0x00 0x30 0x00 0x30 0x00 0x30 0x00 0x31 0x00`
- Vendor ID is 0xAABB, Product ID is 0xCCDD, but there is no Serial Number: `0xBB 0xAA 0xDD 0xCC 0x02 0x03`

8.3.3 Table 5: Device Descriptor Data for Device Type 03h (PCI/PCIe Network Interface)

The following table defines the Device Descriptor Data format for when the Device Type field is 03h, which indicates the device is a PCI/PCIe Network Interface.

Offset	Name	Length	Value	Description
00h	Vendor ID	WORD	Varies	The Vendor ID of the PCI/PCIe device as read from its PCI configuration space (LSB first).
02h	Device ID	WORD	Varies	The Device ID of the PCI/PCIe device as read from its PCI configuration space (LSB first).
04h	Subsystem Vendor ID	WORD	Varies	The Subsystem Vendor ID of the PCI/PCIe device as read from its PCI configuration space (LSB first).
06h	Subsystem ID	WORD	Varies	The Subsystem ID of the PCI/PCIe device as read from its PCI configuration space (LSB first).

Examples type 03h descriptors:

- Vendor ID is 0xAABB, Product ID is 0xCCDD, Subsystem Vendor ID is 0x0011, and Subsystem ID is 0x2233: `0xBB 0xAA 0xDD 0xCC 0x11 0x00 0x33 0x22`

8.3.4 Table 6: Device Descriptor Data for Device Type 04h (USB Network Interface v2)

The following table defines the Device Descriptor Data format for when the Device Type field is 04h, which indicates the device is a USB Network Interface, version 2.

Offset	Name	Length	Value	Description
00h	Length	BYTE	Varies	Length of the structure, including Device Type and Length fields. Length dependent on version of the Redfish Host Interface Specification supported: • 11h for 1.3. • 0Dh for 1.2 and older.
01h	Vendor ID	WORD	Varies	The Vendor ID of the device, as read from the idVendor field of the USB descriptor (LSB first).
03h	Product ID	WORD	Varies	The Product ID of the device, as read from the idProduct field of the USB descriptor (LSB first).
05h	Serial Number	BYTE	STRING	The string number for the Serial Number of the device. The string data is read from the iSerialNumber.bDescriptorType field of the USB descriptor, and is converted from Unicode to ASCII and is NULL terminated. See "Text Strings" in the SMBIOS Specification for how this field is constructed.
06h	MAC Address	6 BYTEs	Varies	The MAC address of the USB network device (most significant octet first).

Offset	Name	Length	Value	Description
0Ch	Device Characteristics	WORD	Varies	Additional characteristics for the device. See Table 9.
0Eh	Credential Bootstrapping Handle	WORD	Varies	Handle of the interface to be used for credential bootstrapping via IPMI commands. The value is 0xFFFF if not supported.

8.3.5 Table 7: Device Descriptor Data for Device Type 05h (PCI/PCIe Network Interface v2)

The following table defines the Device Descriptor Data format for when the Device Type field is 05h, which indicates the device is a PCI/PCIe Network Interface, version 2.

Offset	Name	Length	Value	Description
00h	Length	BYTE	Varies	Length of the structure, including Device Type and Length fields. Length dependent on version of the Redfish Host Interface Specification supported: • 18h for 1.3. • 14h for 1.2 and older.
01h	Vendor ID	WORD	Varies	The Vendor ID of the PCI/PCIe device as read from its PCI configuration space (LSB first).
03h	Device ID	WORD	Varies	The Device ID of the PCI/PCIe device as read from its PCI configuration space (LSB first).
05h	Subsystem Vendor ID	WORD	Varies	The Subsystem Vendor ID of the PCI/PCIe device as read from its PCI configuration space (LSB first).
07h	Subsystem ID	WORD	Varies	The Subsystem ID of the PCI/PCIe device as read from its PCI configuration space (LSB first).
09h	MAC Address	6 BYTEs	Varies	The MAC address of the PCI/PCIe network device (most significant octet first).
0Fh	Segment Group Number	WORD	Varies	The Segment Group Number of the PCI/PCIe device as defined in the PCI Firmware Specification. The value is 0 for a single-segment topology.
11h	Bus Number	BYTE	Varies	The Bus Number of the PCI/PCIe device.
12h	Device/Function Number	BYTE	Varies	The Device/Function Number of the PCI/PCIe device. Bits 7:3 - Device Number Bits 2:0 - Function Number.
13h	Device Characteristics	WORD	Varies	Additional characteristics for the device. See Table 9.
15h	Credential Bootstrapping Handle	WORD	Varies	Handle of the interface to be used for credential bootstrapping via IPMI commands. The value is 0xFFFF if not supported.

8.3.6 Table 8: Device Descriptor Data for Device Types 80h-FFh (OEM)

The following table defines the Device Descriptor Data format for when the Device Type field is 80h-FFh, which indicates the device is an OEM device.

Offset	Name	Length	Value	Description
00h	Vendor IANA	4 BYTEs	Varies	The IANA code for the vendor (MSB first).
04h	Vendor Data	Varies	Varies	OEM defined data.

8.3.7 Table 9: Device Characteristics

The following table defines the bit field for the Device Characteristics field found in Table 6 and Table 7.

Word Bit Position	Meaning
Bit 0	Credential bootstrapping via IPMI commands is supported.
Bits 1:15	Reserved.

8.4 Table 10: Protocol Records data format

Protocol Records start at offset 07h+N of the SMBIOS Type 42 structure. The following table defines the general Protocol Record layout specific data for Redfish over IP protocol:

Offset	Name	Length	Value	Description
00h	Protocol Type	BYTE	ENUM	The Protocol Type. "Redfish over IP" is 04h.
01h	Protocol Type Specific Data Length	BYTE	P	The length of the Protocol Specific Record Data field.
02h	Protocol Specific Record Data	P BYTEs	Varies	The data for the protocol as defined by the Protocol Type. See Table 11 for "Redfish over IP" protocol.

8.4.1 Table 11: "Redfish over IP" Protocol Specific Record Data

Protocol Specific Record Data starts at offset 02h of the Protocol Record structure. The following table defines the protocol specific data for the "Redfish Over IP" protocol:

Offset	Name	Length	Value	Description
00h	Service UUID	16 BYTEs	Varies	Same as Redfish Service UUID in Redfish Service Root resource; set to all 0s if the UUID is not supported or unknown.
10h	Host IP Assignment Type	BYTE	ENUM	The method for how the host IP address is assigned. See Table 12.
11h	Host IP Address Format	BYTE	ENUM	The format of the Host IP Address. See Table 13.
12h	Host IP Address	16 BYTEs	Varies	Used for Static and AutoConfigure. For IPv4, use the first 4 Bytes and zero fill the remaining bytes.
22h	Host IP Mask	16 BYTEs	Varies	Used for Static and AutoConfigure. For IPv4, use the first 4 Bytes and zero fill the remaining bytes.
32h	Redfish Service IP Discovery Type	BYTE	ENUM	The method for how the Redfish Service IP address is discovered. See Table 12.
33h	Redfish Service IP Address Format	BYTE	ENUM	The format of the Redfish Service IP Address. See Table 13.
34h	Redfish Service IP Address	16 BYTEs	Varies	Used for Static and AutoConfigure. For IPv4, use the first 4 Bytes and zero fill the remaining bytes.
44h	Redfish Service IP Mask	16 BYTEs	Varies	Used for Static and AutoConfigure. For IPv4, use the first 4 Bytes and zero fill the remaining bytes.
54h	Redfish Service IP Port	WORD	Varies	Used for Static and AutoConfigure.
56h	Redfish Service VLAN ID	DWORD	Varies	Used for Static and AutoConfigure.
5Ah	Redfish Service Hostname Length	BYTE	Varies	The length in bytes of the "Redfish Service Hostname" field, including any NULL characters in the field.
5Bh	Redfish Service Hostname	Varies	Varies	Hostname of Redfish Service; this string may end with zero or more NULL characters. This string does not follow typical string patterns found elsewhere in SMBIOS definitions.

The UUID is 128 bits long. Its format is described in RFC4122. Although RFC4122 recommends network byte order for all fields, the PC industry (including the ACPI, UEFI, and Microsoft specifications) has consistently used little-endian byte encoding for the first three fields: time_low, time_mid, time_hi_and_version. The same encoding, also known as wire format, should also be used for the SMBIOS representation of the UUID.

- The UUID {00112233-4455-6677-8899-AABBCCDDEEFF} would thus be represented as: `0x33 0x22 0x11 0x00 0x55 0x44 0x77 0x66 0x88 0x99 0xAA 0xBB 0xCC 0xDD 0xEE 0xFF`

In the above table, the fields "Host IP Address", "Host IP Mask", "Redfish Service IP Address", and "Redfish Service IP Mask" shall be stored in network byte order.

- IPv4 Example: 10.12.110.57 will be stored, from lowest offset first, as `0x0A 0x0C 0x6E 0x39 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00`
- IPv6 Example: 2001:db8:63b3:1::3490 will be stored, from lowest offset first, as `0x20 0x01 0x0D 0xB8 0x63 0xB3 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x34 0x90`

8.4.2 Table 12: Assignment/Discovery Type values

The following table defines the possible values for the Assignment Type and Discovery Type fields found in Table 11.

Value	Description
00h	Unknown
01h	Static
02h	DHCP
03h	AutoConfigure
04h	HostSelected
Others	Reserved

8.4.3 Table 13: IP Address Format values

The following table defines the possible values for the Host IP Address Format and Redfish Service IP Address Format fields found in Table 11.

Value	Description
00h	Unknown
01h	IPv4
02h	IPv6
Others	Reserved

8.5 Multiple Protocol Records

The SMBIOS Type 42 structure allows for multiple Protocol Records to be specified within a given Type 42 entry. For Redfish, this could be used to describe both IPv4 and IPv6 information for a single Redfish Service. However, due to the size of the Redfish over IP Protocol, it is likely not possible to use two Protocol Records in a single Type 42 entry without going beyond the limit of the Length field of the Type 42 structure found at offset 01h. If multiple Protocol Records are needed to describe a single Redfish Service, there should be a Type 42 entry for each of the Protocol

Records. The Interface Specific Data portion of each Type 42 entry shall be identical if they map to the same Redfish Service. Clients should interpret multiple Type 42 entries with identical Interface Specific Data as a single Redfish Service with multiple protocols.

This methodology also allows for multiple Redfish Services to be described by multiple Type 42 entries. Clients should interpret Type 42 entries with different Interface Specific Data as different Redfish Services.

9 Credential bootstrapping via IPMI commands

Services may support provisioning a temporary account for the host interface using IPMI commands. This is to allow a host with no initial knowledge of the Redfish interface to create an initial account from which it can create a permanent account for the life of the system.

Services that support credential bootstrapping via IPMI commands shall:

- Implement the `CredentialBootstrapping` property defined in the `HostInterface` schema.
- Implement the `CredentialBootstrappingRole` property in the `Links` property defined in the `HostInterface` schema.

9.1 IPMI command definitions

The following clause defines IPMI commands added to the "Group Extension" NetFn (2Ch, 2Dh). The defining body code for Redfish is 52h.

The service shall implement the IPMI commands defined in this clause. The IPMI commands defined in this clause shall be rejected on interfaces that are not the system interface.

9.1.1 Get Manager Certificate Fingerprint (NetFn 2Ch, Command 01h)

This command is used to get the fingerprint for the manager's TLS certificate for the host interface. In most cases, the manager will only have one certificate available. Clients should use the response data from this command when connecting over the host interface to verify the service's certificate matches.

Request data:

Byte	Data field
1	Group extension identification (52h).
2	Certificate number (1 based).

Response data:

Byte	Data field
1	Completion code: • 00h: Command completed normally. • 80h: Credential bootstrapping via IPMI commands is disabled. • CBh: Certificate number is invalid.
2	Group extension identification (52h).
3	Fingerprint hash algorithm: • 01h: SHA-256. The length of the fingerprint will be 32 bytes. • Others: Reserved.
4:N	Fingerprint of the manager's TLS certificate.

9.1.2 Get Bootstrap Account Credentials (NetFn 2Ch, Command 02h)

This command is used to get the user name and password of the bootstrap account. When the manager receives this command, it shall generate a new user name and password for the bootstrap account, store the credentials in a manner consistent with other Redfish accounts, and provide the credentials in the response data to the host. The manager shall generate a unique user name, and shall randomly generate the password.

The user name and password shall:

- Be valid UTF-8 strings consisting of printable characters.
- Not include single quote (') or double quote (") characters.
- Not include backslash (\) characters.
- Not include whitespace characters.
- Not include control characters.
- Be at most 16 characters long.

Upon normal completion of this command, the `Enabled` property within the `CredentialBootstrapping` property of the host interface resource shall be set to `false`, unless the "Disable credential bootstrapping control" parameter of the command contains the value `A5h`.

Request data:

Byte	Data field
1	Group extension identification (52h).

Byte	Data field
2	Disable credential bootstrapping control: • A5h: Keep credential bootstrapping enabled. • Other values: Disable credential bootstrapping upon normal completion of this command.

Response data:

Byte	Data field
1	Completion code: • 00h: Command completed normally. • 80h: Credential bootstrapping via IPMI commands is disabled.
2	Group extension identification (52h).
3:18	The user name as a UTF-8 string. Strings with fewer than 16 characters are terminated with a null (00h) character and 00h padded to 16 bytes.
19:34	The password as a UTF-8 string. Strings with fewer than 16 characters are terminated with a null (00h) character and 00h padded to 16 bytes.

9.2 Account life cycle

If the `Enabled` property within the `CredentialBootstrapping` property of the host interface resource is `false`, the Get Manager Certificate Fingerprint and Get Bootstrap Account Credentials commands shall be rejected and respond with the completion code 80h.

If the Get Bootstrap Account Credentials command has been issued and responds with the completion code 00h, a bootstrap account shall be added to the manager's account collection and enabled. If the Get Bootstrap Account Credentials command is sent subsequent times and responds with the completion code 00h, a new account shall be created based on the newly generated credentials. Any existing bootstrap accounts shall remain active.

Bootstrap accounts shall be usable only on the host interface.

Services shall delete all bootstrap accounts:

- On any reset of the service.
- On any reset of the host. Implementers may need to provision for the necessary mechanisms in order to detect a host reset, which could be hardware-based, UEFI-based, or a mixture of the two.

If the `EnableAfterReset` property within the `CredentialBootstrapping` property of the host interface resource is

`true`, services shall set the `Enabled` property within the `CredentialBootstrapping` property of the host interface resource to `true`:

- On any reset of the service.
- On any reset of the host.

9.3 Recommendations for hosts

Anyone with root or administrative access to the host is able to make use of this mechanism for provisioning an account. However, it is recommended that this mechanism only be used during initial provisioning of a host. The overall workflow for using this interface is recommended to be:

1. Use this mechanism to create a bootstrap account. When issuing the Get Bootstrap Account Credentials command, do not set the "Disable credential bootstrapping control" parameter of the command to `A5h`.
2. Using the bootstrap account, create a permanent account for the host in the manager account collection found in the account service.
3. Perform a DELETE on the manager account representing the bootstrap account.

10 Delivery of credentials via UEFI runtime variables

> **DEPRECATED:** This entire clause has been deprecated in favor of other credential negotiation mechanisms described in this specification, such as credential bootstrapping via IPMI commands.

DEPRECATED

This clause defines an optional mechanism for automatically generating and sending credentials to the Host OS kernel and/or firmware by using UEFI runtime variables. Services that implement the kernel authentication mechanism shall comply with the following subclauses:

10.1 Credential generation and management for use by firmware and OS kernel

Opening a Redfish session on the Host Interface may be accomplished by use of any authorized Redfish credentials consisting of a valid username and password. However in some situations, it may be difficult to pre-provision the system with valid Redfish credentials for use by OS and firmware clients of Redfish. Examples of this include (a) early provisioning of new systems and (b) systems where firmware does not have secure storage to hide the credentials.

To provide for situations of this type, systems supporting the Redfish Service may optionally be configured to provide temporary login credentials for use by Firmware Pre-boot elements and/or OS. If implemented, the credentials shall follow these requirements:

- The credentials shall be auto-generated by the Redfish Service and provided to firmware and OS by using the UEFI variable interface variables described herein at the initiation of each system boot.
- The generation of both firmware and OS credentials shall be user-configurable with the option to disable or enable generation of the credentials separately for both firmware and OS kernel.
- The permissions of the resulting session shall be configurable through the Redfish Service configuration as described by the Redfish schema.
- The supplied credentials shall be in the form of a user id and password -- both auto-generated by the Redfish Service.
- Only one session using these auto-generated credentials shall be allowed at a time.
- The session associated with the auto-generated credentials shall not timeout or expire.
- The Redfish Service may close the session if it resets or for other policy reasons in which case the Host may re-open the session using the same credentials.
- Any open session started with firmware credentials shall be closed and the credentials invalidated at UEFI `ExitBootServices()` event.

- Any open session started with OS credentials shall be closed and new credential passwords generated when a Host restart is detected by the Redfish Service.
- The Firmware Credentials shall be made available for any agent or driver that operates within the UEFI pre-boot prior to ExitBootServices() call. This may include local system ROM firmware or utility firmware applications downloaded from external sources.

10.2 Security considerations for protecting auto-generated credentials

It is recommended that system designers protect the credentials from unauthorized access. The use of UEFI Secure Boot to protect access to credentials is recommended. Because of the difficulty of defining a security procedure for Legacy-booting OS, delivery of credentials to Legacy OS is not described by this specification and any Legacy OS support for this feature is OEM-specific.

The system OS is provided with a method of disabling further retrieval of the credentials after initial authorized retrieval. System designers are encouraged to implement such a scheme of retrieve, store, and disable to avoid unauthorized reading of the credential variables

10.3 UEFI implementation

Implementations that present a Redfish Host Interface for use by system firmware and OS shall use the UEFI Variables defined in this section to deliver credentials for the Redfish Host Interface.

The design of this delivery mechanism is compatible with any UEFI version starting with 2.3.1. Refer to the specifications available at www.uefi.org for details on using the UEFI variable calls described here.

10.3.1 Prototype

```
#define EFI_REDFISH_INFORMATION_GUID \
    {0x16faa37e, 0x4b6a, 0x4891, {0x90, 0x28, 0x24, 0x2d, 0xe6, 0x5a, 0x3b, 0x70 }}
#define EFI_REDFISH_INFORMATION_INDICATIONS         L"RedfishIndications"
#define EFI_REDFISH_INFORMATION_FW_CREDENTIALS      L"RedfishFWCredentials"
#define EFI_REDFISH_INFORMATION_OS_CREDENTIALS      L"RedfishOSCredentials"
```

10.3.2 Related definitions

```
#define EFI_REDFISH_INDICATIONS_FW_CREDENTIALS      0x00000001
#define EFI_REDFISH_INDICATIONS_OS_CREDENTIALS      0x00000002
```

10.3.3 Description

This GUID and these variable names are used when calling the UEFI Runtime Service `GetVariable()`. See the UEFI Specification for details on use of this interface. As described below, the `SetVariable()` interface can be used to disable further access to the credential information.

The variables defined in this section have the following attributes:

- `EFI_REDFISH_INFORMATION_INDICATIONS` and `EFI_REDFISH_INFORMATION_OS_CREDENTIALS` have attributes `EFI_VARIABLE_BOOTSERVICE_ACCESS` and `EFI_VARIABLE_RUNTIME_ACCESS`.
- `EFI_REDFISH_INFORMATION_FW_CREDENTIALS` has attribute `EFI_VARIABLE_BOOTSERVICE_ACCESS`

The variable `EFI_REDFISH_INFORMATION_INDICATIONS` shall return a 32-bit value, and provides information if any credentials are provided for the Host Software use. The bits defined with this variable shall be interpreted as follows:

- If `EFI_REDFISH_ INDICATIONS_FW_CREDENTIALS` bit is 1, the Redfish Host Interface is configured to provide credentials for use by system firmware.
- If `EFI_REDFISH_ INDICATIONS_OS_CREDENTIALS` bit is 1, the Redfish Host Interface is configured to provide a credentials for use by system OS.
- All other bits in `EFI_REDFISH_INDICATIONS_HOST_IF` are reserved.

When the Redfish implementation provides credentials for firmware use, the variable `EFI_REDFISH_INFORMATION_FW_CREDENTIALS` shall contain a UTF-8 character array formatted as described in the next clause. If this session is not available as defined by current system policy, this variable shall return `EFI_NOT_FOUND`.

When the Redfish implementation provides credentials for OS use, the variable `EFI_REDFISH_INFORMATION_OS_CREDENTIALS` a UTF-8 character array formatted as described in the next clause. If these credentials are not available as defined by current system policy, this variable shall return `EFI_NOT_FOUND`.

The password contained in these variables shall be recalculated so as to be unique and not easily predicted on each boot.

If the variables `EFI_REDFISH_INFORMATION_FW_CREDENTIALS` or `EFI_REDFISH_INFORMATION_OS_CREDENTIALS` are accessed using the `SetVariable()` function with a *DataSize* of zero, the variable contents shall be hidden until the next system restart and not be available for retrieval by future `GetVariable()` calls. After such `SetVariable()` access any `GetVariable()` attempt shall return `EFI_NOT_FOUND` error. Calls to `SetVariable()` with nonzero *DataSize* shall be processed as if *DataSize* is zero.

10.3.4 Variable format

The UEFI variables for delivery of temporary credentials shall contain an array of UTF-8 characters in the format Username:Password where the : character shall act as separator. The final byte of the array shall be 0x00 as terminator

and the size of the variable shall be length of Username plus length of Password plus 2. Characters shall be chosen from the set elsewhere defined as legal for Redfish Username and Password and neither field may contain the : character.

For convenience when identifying the auto-generated credentials when active and for the purpose of editing permissions, the following Username strings shall be used:

Usage	Username
Default Firmware Auto Username	HostAutoFW
Default OS Auto Username	HostAutoOS

DEPRECATED

11 ANNEX A (informative) Change log

Version	Date	Description
1.3.0	2020-08-04	Added clauses for bootstrapping credentials via IPMI.
		Deprecated credential reporting via UEFI variables.
		Extended USB Network Interface v2 and PCI/PCIe Network Interface v2 descriptors to include Device Characteristics and Credential Bootstrapping Handle fields.
1.2.0	2019-10-11	Added clause to allow for multiple SMBIOS Type 42 structures in order to specify multiple Protocol Record.
1.1.0	2019-05-16	Clarified the byte ordering in SMBIOS structures.
		Clarified the data shown in the Device Descriptor Table.
		Clarified the format of the Host Name field.
		Added example device descriptors.
		Added version 2 of the USB and PCI/PCI-e device descriptors.
		Clarified the format of the UUID.
1.0.1	2017-12-11	Errata release. Numerous terminology clarifications and typographical corrections.
		Terminology for 'host', 'manager' and 'service' were edited for consistency.
		Added additional wording about the SMBIOS Type 42 structure to describe its purpose.
		Added references to the UEFI Specification.
		Clarified byte ordering of IPv4 and IPv6 addresses in the SMBIOS Type 42 structure.
		Added missing case for what to use for the UUID in the SMBIOS Type 42 structure if it is unknown or not supported.
1.0.0	2016-12-30	Initial release.