Document Identifier: DSP0270

Date: 2018-11-29

Version: 1.1.0a

Redfish Host Interface Specification

Information for Work-in-Progress version:

IMPORTANT: This document is not a standard. It does not necessarily reflect the views of the DMTF or its members. Because this document is a Work in Progress, this document may still change, perhaps profoundly and without notice. This document is available for public review and comment until superseded.

Provide any comments through the DMTF Feedback Portal: http://www.dmtf.org/standards/feedback

Document Class: Normative

Document Status: Work in Progress

Document Language: en-US

CONTENTS

Foreword

The Redfish Host Interface Specification was prepared by the Scalable Platforms Management 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.

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.

1. 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.

2. 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

3. 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, Annex H. 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, Annex H 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 5.

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 preboot 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.

4. 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,

Term	Definition
	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

5. 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 preboot (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. 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.

6.1. 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 multiservice to multihost 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.
- 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 Kernel authentication 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.

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 3 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	Varies	Management Controller Interface Type. --Network Host Interface = 40h
05h	Interface Specific Data Length (n)	BYTE	Varies	Interface-specific Data as specified by the Interface type. if 0, there is no Interface specific data
06h	Interface Specific data	n BYTEs	Varies	Defined by Interface Type. See Table-2 below.
06h+n	Protocol count	BYTE	Varies	Number of protocols defined for the Host Interface (typically 1)
07h+n	Protocol Records	m Bytes	Varies	Include a Protocol Record for each protocol supported. See Table-4 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 five types of Device Descriptors defined (see Table-3); however, only one may be used in specific Type 42 table.

Offset	Name	Length	Value	Description
X	Device Type	BYTE	Enum	USB Network Interface = 02h, PCI/PCIe Network Interface = 03h, USB Network Interface v2 = 04h, PCI/PCIe Network Interface v2 = 05h, OEM = 80h-FFh other values reserved
X+1	Device Descriptors	n-1 Bytes	Varies	Device descriptor data formatted based on Device Type. See Table-3

8.4. Table-3: Device descriptor data

The following table defines the specific Device Descriptor data (referenced in Table-2) for each defined Device Type:

Device Type enum value	Device Type Name	Length	Value	Description
02h	USB Network Interface	Varies	Varies	Device Descriptors for USB Device Type: - idVendor (2 bytes) - idProduct (2 bytes) - iSerialNumber: --- bLength (1 byte) --- bDescriptorType (1 byte) --- bString (Varies)
03h	PCI/PCIe Network Interface	8 bytes	Varies	Device Descriptors for PCI/PCIe Device Type: - Vendor ID (2 bytes) - Device ID (2 bytes) - Subsystem Vendor ID (2 bytes) - Subsystem ID (2 bytes)
04h	USB	Varies	Varies	Device Descriptors for USB Device Type v2:

Device Type enum value	Device Type Name	Length	Value	Description
	Network Interface v2			- Length of this structure, including type and length fields (1 byte) - Vendor ID (2 bytes) - Product ID (2 bytes) - String Number for the Serial Number (1 byte) - MAC Address (6 bytes)
05h	PCI/PCIe Network Interface v2	Varies	Varies	Device Descriptors for PCI/PCIe Device Type v2: - Length of this structure, including type and length fields (1 byte) - Vendor ID (2 bytes) - Device ID (2 bytes) - Subsystem Vendor ID (2 bytes) - Subsystem ID (2 bytes) - MAC Address (6 bytes) - Segment Group Number (2 bytes) --- Segment Group Number is defined in the PCI Firmware Specification. The value is 0 for a single-segment topology - Bus Number (1 byte) - Device/Function Number (1 byte): --- Bits 7:3 - Device Number --- Bits 2:0 - Function Number
80h-FFh	OEM	Varies	Varies	Device Descriptors for OEM Device Type: - Vendor IANA (4 bytes) - OEM defined data

For USB devices (types 02h and 04h):

- idVendor (Vendor ID), idProduct (Product ID), and iSerialNumber (Serial Number) originate from the USB descriptor for the device.
- For type 02h, within iSerialNumber, bDescriptorType is always 0x03 and bString is a Unicode string without a NULL terminator.
    - This string does not follow typical string patterns found elsewhere in SMBIOS definitions.
- For type 04h, the string referenced by the string number after the descriptor is converted from Unicode to ASCII and is NULL terminated per SMBIOS design patterns.

Examples of USB devices (type 02h):

- 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`

For PCI/PCIe devices (types 03h and 05h):

- Vendor ID, Device ID, Subsystem Vendor ID, and Subsystem ID originate from the PCI configuration space for the device.

Examples of PCI/PCIe devices (type 03h):

- 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`

8.5. Table-4: Protocol Records data format:

The following table defines the general Protocol Record layout specific data for Redfish over IP protocol:

Offset	Name	Length	Value	Description
X	Protocol Identifier	BYTE	Varies	Protocol identifier --"Redfish over IP" = 04h
X+1	Length	BYTE	Varies	Length of protocol specific data for Redfish over IP protocol
X+2	Protocol specific record data	p Bytes	Varies	Defined by protocol. See Table-5 below for "Redfish over IP" protocol

8.6. Table-5: "Redfish over IP" Protocol-specific record data

The following table defines the protocol-specific data for the "Redfish Over IP" protocol:

Offset	Name	Length	Value	Description
X+0	Service UUID	16BYTEs	Varies	Same as Redfish Service UUID in Redfish Service Root resource; set to all 0s if the UUID is not supported or unknown.

Offset	Name	Length	Value	Description
X+16	Host IP Assignment Type	BYTE	Enum	Unknown=00h, Static=01h, DHCP=02h, AutoConfigure=03h, HostSelected=04h, other values reserved
X+17	Host IP Address Format	BYTE	Enum	Unknown=00h, IPv4=01h, IPv6=02h, other values reserved
X+18	Host IP Address	16BYTEs	Varies	Used for Static and AutoConfigure. For IPv4, use the first 4 Bytes and zero fill the remaining bytes.
X+34	Host IP Mask	16BYTEs	Varies	Used for Static and AutoConfigure. For IPv4, use the first 4 Bytes and zero fill the remaining bytes.
X+50	Redfish Service IP Discovery Type	BYTE	Enum	Unknown=00h, Static=01h, DHCP=02h, AutoConfigure=03h, HostSelected=04h, other values reserved
X+51	Redfish Service IP Address Format	BYTE	Enum	Unknown=00h, IPv4=01h, IPv6=02h, other values reserved
X+52	Redfish Service IP Address	16BYTEs	Varies	Used for Static and AutoConfigure. For IPv4, use the first 4 Bytes and zero fill the remaining bytes.
X+68	Redfish Service IP Mask	16BYTEs	Varies	Used for Static and AutoConfigure. For IPv4, use the first 4 Bytes and zero fill the remaining bytes.
X+84	Redfish Service IP Port	WORD	Varies	Used for Static and AutoConfigure.

Offset	Name	Length	Value	Description
X+86	Redfish Service VLAN ID	DWORD	Varies	Used for Static and AutoConfigure.
X+90	Redfish Service Hostname Length	BYTE	Varies	The length in bytes of the "Redfish Service Hostname" field, including any NULL characters in the field
X+91	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.

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`

9. Delivery of kernel authentication information via UEFI runtime variables

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:

9.1. 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 logon credentials for use by Firmware Preboot 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 preboot prior to ExitBootServices() call. This may include local system ROM firmware or utility firmware applications downloaded from external sources.

9.2. 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

9.3. 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.

9.3.1. 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"
```

9.3.2. Related definitions

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

9.3.3. 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.

9.3.4. 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

10. ANNEX A (informative)

10.1. Change log

Version	Date	Description
1.1.0a	2018-11-29	Clarified the byte ordering in SMBIOS structures.

Version	Date	Description
		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.
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.