BS ISO/IEC 30115-1:2022

BSI Standards Publication

Information technology — Redfish scalable platforms management API specification

Part 1: Redfish Specification v1.13.0

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

National foreword

This British Standard is the UK implementation of ISO/IEC 30115-1:2022. It supersedes BS ISO/IEC 30115:2018, which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee ICT/1, Information systems co-ordination.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2022
Published by BSI Standards Limited 2022

ISBN 978 0 539 19645 0

ICS 35.020

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 September 2022.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

INTERNATIONAL STANDARD

ISO/IEC 30115-1

First edition
2022-08

Information technology — Redfish scalable platforms management API specification —

Part 1:
Redfish Specification v1.13.0

Reference number
ISO/IEC 30115-1:2022(E)

Documents
Document Sharing Hub

© ISO/IEC 2022

Full document details: BS ISO/IEC 30115-1:2022

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2022

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by DMTF (as DMTF Redfish Specification, DSP0266, Version 1.13.0) and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

This first edition cancels and replaces the first edition (ISO/IEC 30115:2018), which has been technically revised.

The main changes are as follows:

— a number of clauses have been added expanding the functionality of the protocol, including additions for controlling software-defined composable systems, for performing software or firmware updates and for aggregating multiple service instances;

— the protocol itself has been enhanced to provide a number of optional query parameters to enable more efficient transactions;

— the specification and its data model have been enhanced to provide compatibility with the OpenAPI v3.0 specification allowing the use of the OpenAPI toolchain for both documentation and code generation.

A list of all parts in the ISO/IEC 30115 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Documents
Document Sharing Hub

© ISO/IEC 2022 – All rights reserved

Full document details: BS ISO/IEC 30115-1:2022

Copyright Notice

Copyright © 2015-2021 DMTF. All rights reserved.

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.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

CONTENTS

1 Foreword {-} . . . 11
1.1 Acknowledgments {-} . . . 11
2 Introduction {-} . . . 13
3 Scope . . . 14
4 Normative references . . . 15
5 Terms, definitions, symbols, and abbreviated terms . . . 17
5.1 Hardware terms . . . 17
5.1.1 baseboard management controller (BMC) . . . 17
5.1.2 IPMI . . . 17
5.1.3 KVM-IP . . . 18
5.1.4 NIC . . . 18
5.1.5 PCI . . . 18
5.1.6 PCIe . . . 18
5.2 Web development terms . . . 18
5.2.1 CORS . . . 18
5.2.2 CRUD . . . 18
5.2.3 CSRF . . . 18
5.2.4 event . . . 18
5.2.5 excerpt . . . 18
5.2.6 HTTP . . . 19
5.2.7 HTTPS . . . 19
5.2.8 hypermedia API . . . 19
5.2.9 IP . . . 19
5.2.10 JSON . . . 19
5.2.11 member . . . 19
5.2.12 message . . . 19
5.2.13 OData . . . 19
5.2.14 OData service document . . . 19
5.2.15 operation . . . 20
5.2.16 parent resource . . . 20
5.2.17 property . . . 20
5.2.18 request . . . 20
5.2.19 response . . . 20
5.2.20 subscription . . . 20
5.2.21 task . . . 20
5.2.22 task monitor . . . 20
5.2.23 TCP . . . 20
5.2.24 TLS . . . 21
5.2.25 XSS . . . 21
5.3 Redfish terms . . . 21
5.3.1 collection . . . 21

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

5.3.2 Redfish client . . . 21
5.3.3 Redfish protocol . . . 21
5.3.4 Redfish schema . . . 21
5.3.5 Redfish service . . . 21
5.3.6 resource . . . 21
5.3.7 resource collection . . . 22
5.3.8 resource tree . . . 22
5.3.9 resource type . . . 22
5.3.10 service root . . . 22
5.3.11 subordinate resource . . . 22
6 Typographical conventions . . . 23
7 Overview . . . 24
7.1 Goals . . . 24
7.2 Design tenets . . . 25
7.3 Limitations . . . 25
7.4 Additional design background and rationale . . . 26
7.4.1 REST-based interface . . . 26
7.4.2 Data-oriented . . . 26
7.4.3 Separation of protocol from data model . . . 26
7.4.4 Hypermedia API service root . . . 26
7.4.5 OpenAPI v3.0 support . . . 26
7.4.6 OData conventions . . . 27
7.5 Service elements . . . 27
7.5.1 Synchronous and asynchronous operation support . . . 27
7.5.2 Eventing mechanism . . . 27
7.5.3 Actions . . . 28
7.5.4 Service discovery . . . 28
7.5.5 Remote access support . . . 28
7.6 Security . . . 28
8 Protocol details . . . 29
8.1 Universal Resource Identifiers . . . 29
8.2 HTTP methods . . . 31
8.3 HTTP redirect . . . 32
8.4 Media types . . . 32
8.5 ETags . . . 32
8.6 Protocol version . . . 33
8.7 Redfish-defined URIs and relative reference rules . . . 34
9 Service requests . . . 36
9.1 Request headers . . . 36
9.2 GET (read requests) . . . 39
9.2.1 GET (read requests) overview . . . 39
9.2.2 Resource collection requests . . . 39
9.2.3 Service root request . . . 40

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

9.2.4 OData service and metadata document requests . . . 40
9.3 Query parameters . . . 41
9.3.1 Query parameter overview . . . 41
9.3.2 The $expand query parameter . . . 43
9.3.3 The $select query parameter . . . 45
9.3.4 The $filter query parameter . . . 46
9.4 HEAD . . . 47
9.5 Data modification requests . . . 48
9.5.1 Data modification requests overview . . . 48
9.5.2 Modification success responses . . . 48
9.5.3 Modification error responses . . . 49
9.6 PATCH (update) . . . 49
9.7 PATCH on array properties . . . 50
9.8 PUT (replace) . . . 51
9.9 POST (create) . . . 51
9.10 DELETE (delete) . . . 52
9.11 POST (action) . . . 52
9.12 Operation apply time . . . 54
9.13 Deep operations . . . 57
10 Service responses . . . 61
10.1 Response headers . . . 61
10.2 Link header . . . 63
10.3 Status codes . . . 63
10.4 OData metadata responses . . . 66
10.4.1 OData metadata responses overview . . . 66
10.4.2 OData $metadata . . . 66
10.4.2.1 Referencing other schemas . . . 67
10.4.2.2 Referencing OEM extensions . . . 67
10.4.3 OData service document . . . 68
10.5 Resource responses . . . 68
10.6 Error responses . . . 69
11 Data model . . . 71
11.1 Resources . . . 71
11.2 Resource types . . . 71
11.3 Resource collections . . . 72
11.4 OEM resources . . . 72
11.5 Common data types . . . 73
11.5.1 Primitive types . . . 73
11.5.2 Empty string values . . . 73
11.5.3 GUID and UUID values . . . 74
11.5.4 Date-Time values . . . 74
11.5.5 Duration values . . . 74
11.5.6 Reference properties . . . 75

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

11.5.7 Non-resource reference properties . . . 75
11.5.8 Array properties . . . 76
11.5.9 Structured properties . . . 76
11.5.10 Message object . . . 77
11.5.10.1 Overview . . . 77
11.5.10.2 MessageId format . . . 78
11.6 Properties . . . 79
11.6.1 Properties overview . . . 79
11.6.2 Resource identifier (@odata.id) property . . . 79
11.6.3 Resource type (@odata.type) property . . . 79
11.6.4 Resource ETag (@odata.etag) property . . . 80
11.6.5 Resource context (@odata.context) property . . . 80
11.6.6 Id . . . 81
11.6.7 Name . . . 81
11.6.8 Description . . . 81
11.6.9 MemberId . . . 81
11.6.10 Count (Members@odata.count) property . . . 81
11.6.11 Members . . . 82
11.6.12 Next link (Members@odata.nextLink) property . . . 82
11.6.13 Links . . . 82
11.6.13.1 Reference to a related resource . . . 82
11.6.13.2 References to multiple related resources . . . 83
11.6.14 Actions property . . . 83
11.6.14.1 Action representation . . . 83
11.6.14.2 Action responses . . . 84
11.6.15 Oem . . . 84
11.6.16 Status . . . 85
11.7 Naming conventions . . . 85
11.7.1 Naming rules . . . 85
11.7.2 URI naming rules . . . 86
11.8 Extending standard resources . . . 87
11.8.1 Extending standard resources overview . . . 87
11.8.2 OEM property format and content . . . 87
11.8.3 OEM-specified object naming . . . 87
11.8.4 OEM resource types . . . 88
11.8.5 OEM registries . . . 88
11.8.6 OEM URIs . . . 89
11.8.7 OEM property examples . . . 89
11.8.8 OEM actions . . . 90
11.9 Payload annotations . . . 91
11.9.1 Payload annotations overview . . . 91
11.9.2 Allowable values . . . 91
11.9.3 Extended information . . . 92

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

11.9.3.1 Extended object information . . . 92
11.9.3.2 Extended property information . . . 93
11.9.4 Action info annotation . . . 93
11.9.5 Settings and settings apply time annotations . . . 94
11.9.6 Operation apply time and operation apply time support annotations . . . 94
11.9.7 Maintenance window annotation . . . 94
11.9.8 Collection capabilities annotation . . . 95
11.9.9 Requested count and allow over-provisioning annotations . . . 97
11.9.10 Zone affinity annotation . . . 97
11.9.11 Supported certificates annotation . . . 98
11.9.12 Deprecated annotation . . . 98
11.10 Settings resource . . . 98
11.11 Special resource situations . . . 101
11.11.1 Overview . . . 101
11.11.2 Absent resources . . . 101
11.12 Registries . . . 101
11.13 Schema annotations . . . 102
11.13.1 Schema annotations overview . . . 102
11.13.2 Description annotation . . . 102
11.13.3 Long description annotation . . . 103
11.13.4 Resource capabilities annotation . . . 103
11.13.5 Resource URI patterns annotation . . . 103
11.13.6 Additional properties annotation . . . 104
11.13.7 Permissions annotation . . . 105
11.13.8 Required annotation . . . 105
11.13.9 Required on create annotation . . . 105
11.13.10 Units of measure annotation . . . 105
11.13.11 Expanded resource annotation . . . 105
11.13.12 Owning entity annotation . . . 106
11.13.13 Deprecated annotation . . . 106
11.14 Versioning . . . 106
11.15 Localization . . . 107
12 File naming and publication . . . 108
12.1 Registry file naming . . . 108
12.2 Profile file naming . . . 108
12.3 Dictionary file naming . . . 108
12.4 Localized file naming . . . 108
12.5 DMTF Redfish file repository . . . 109
13 Schema definition languages . . . 111
13.1 OData Common Schema Definition Language . . . 111
13.1.1 OData Common Schema Definition Language overview . . . 111
13.1.2 File naming conventions for CSDL . . . 111
13.1.3 Core CSDL files . . . 111

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

13.1.4 CSDL format . . . 112
13.1.4.1 Referencing other CSDL files . . . 112
13.1.4.2 CSDL data services . . . 113
13.1.5 Elements of CSDL namespaces . . . 113
13.1.5.1 Qualified names . . . 114
13.1.5.2 Entity type and complex type elements . . . 114
13.1.5.3 Action element . . . 115
13.1.5.4 Action element for OEM actions . . . 116
13.1.5.5 Action with a response body . . . 116
13.1.5.6 Property element . . . 117
13.1.5.7 Navigation property element . . . 118
13.1.5.8 Enum type element . . . 118
13.1.5.9 Annotation element . . . 119
13.2 JSON Schema . . . 122
13.2.1 JSON Schema overview . . . 122
13.2.2 File naming conventions for JSON Schema . . . 122
13.2.3 Core JSON Schema files . . . 122
13.2.4 JSON Schema format . . . 123
13.2.5 JSON Schema definitions body . . . 123
13.2.5.1 Resource definitions in JSON Schema . . . 123
13.2.5.2 Enumerations in JSON Schema . . . 124
13.2.5.3 Actions in JSON Schema . . . 125
13.2.5.4 OEM actions in JSON Schema . . . 126
13.2.5.5 Action with a response body . . . 127
13.2.6 JSON Schema terms . . . 128
13.3 OpenAPI . . . 129
13.3.1 OpenAPI overview . . . 129
13.3.2 File naming conventions for OpenAPI schema . . . 129
13.3.3 Core OpenAPI schema files . . . 129
13.3.4 openapi.yaml . . . 130
13.3.5 OpenAPI file format . . . 132
13.3.6 OpenAPI components body . . . 132
13.3.6.1 Resource definitions in OpenAPI . . . 132
13.3.6.2 Enumerations in OpenAPI . . . 133
13.3.6.3 Actions in OpenAPI . . . 133
13.3.6.4 OEM actions in OpenAPI . . . 135
13.3.7 OpenAPI terms used by Redfish . . . 135
13.4 Schema modification rules . . . 136
14 Service details . . . 138
14.1 Eventing . . . 138
14.1.1 Eventing overview . . . 138
14.1.2 POST to subscription collection . . . 138
14.1.3 Open an SSE connection . . . 139

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

14.1.4 EventType-based eventing . . . 140
14.1.5 Subscribing to events . . . 140
14.1.6 Event formats . . . 141
14.1.7 OEM extensions . . . 142
14.2 Asynchronous operations . . . 142
14.3 Resource tree stability . . . 144
14.4 Discovery . . . 144
14.4.1 Discovery overview . . . 144
14.4.2 UPnP compatibility . . . 145
14.4.3 USN format . . . 145
14.4.4 M-SEARCH response . . . 145
14.4.5 Notify, alive, and shutdown messages . . . 146
14.5 Server-sent events . . . 146
14.5.1 General . . . 146
14.5.2 Event service . . . 147
14.5.2.1 Event message SSE stream . . . 149
14.5.2.2 Metric report SSE stream . . . 150
14.6 Update service . . . 151
14.6.1 Overview . . . 151
14.6.2 Software update types . . . 151
14.6.2.1 Simple updates . . . 151
14.6.2.2 Multipart HTTP push updates . . . 151
15 Security details . . . 154
15.1 Transport Layer Security (TLS) protocol . . . 154
15.1.1 Transport Layer Security (TLS) protocol overview . . . 154
15.1.2 Cipher suites . . . 154
15.1.3 Certificates . . . 155
15.2 Sensitive data . . . 155
15.3 Authentication . . . 155
15.3.1 Authentication overview . . . 155
15.3.2 Authentication requirements . . . 156
15.3.2.1 Resource and operation authentication requirements . . . 156
15.3.2.2 HTTP header authentication requirements . . . 156
15.3.2.3 Authentication failure requirements . . . 156
15.3.3 HTTP Basic authentication . . . 157
15.3.4 Redfish session login authentication . . . 157
15.3.4.1 Redfish login sessions . . . 157
15.3.4.2 Session login . . . 158
15.3.4.3 Session lifetime . . . 159
15.3.4.4 Session termination or logout . . . 159
15.4 Authorization . . . 159
15.4.1 Authorization overview . . . 159
15.4.2 Privilege model . . . 160

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

15.4.2.1 Roles . . . 160
15.4.2.2 Restricted roles and restricted privileges . . . 161
15.4.2.3 OEM privileges . . . 162
15.4.3 Redfish service operation-to-privilege mapping . . . 162
15.4.3.1 Why specify operation-to-privilege mapping? . . . 162
15.4.3.2 Representing operation-to-privilege mappings . . . 162
15.4.3.3 Operation map syntax . . . 163
15.4.3.4 Mapping overrides syntax . . . 164
15.4.3.5 Property override example . . . 164
15.4.3.6 Subordinate override . . . 165
15.4.3.7 Resource URI override . . . 166
15.4.3.8 Privilege AND and OR syntax . . . 167
15.5 Account service . . . 168
15.5.1 Account service overview . . . 168
15.5.2 Password management . . . 168
15.5.3 Password change required handling . . . 169
15.6 Asynchronous tasks . . . 169
15.7 Event subscriptions . . . 169
16 Redfish Host Interface . . . 170
17 Redfish composability . . . 171
17.1 Composition requests . . . 172
17.1.1 Composition requests overview . . . 172
17.1.2 Specific composition . . . 172
17.1.3 Constrained composition . . . 173
17.1.4 Expandable resources . . . 174
17.2 Updating a composed resource . . . 174
18 Aggregation . . . 175
18.1 Classes of aggregators . . . 175
18.1.1 Implicit and complex aggregators . . . 175
18.1.2 Use cases . . . 176
18.2 Aggregation service . . . 176
18.2.1 Aggregation service overview . . . 176
18.2.2 Aggregator requirements . . . 176
18.2.3 Aggregates . . . 177
18.2.4 Aggregation sources and connection methods . . . 177
19 ANNEX A (informative) Change log . . . 179
20 Bibliography . . . 192

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

1 Foreword {-}

The Redfish Forum of the DMTF develops the Redfish standard.

DMTF is a not-for-profit association of industry members that promotes enterprise and systems management and interoperability. For information about the DMTF, see DMTF.

This version supersedes version 1.12.1. For a list of the changes, see ANNEX A (informative) Change log.

1.1 Acknowledgments {-}

The DMTF acknowledges the following individuals for their contributions to the Redfish standard, including this document and Redfish schemas, interoperability profiles, and message registries:

- Rafiq Ahamed - Hewlett Packard Enterprise
- Richelle Ahlvers - Broadcom Inc.
- Jeff Autor - Hewlett Packard Enterprise
- David Black - Dell Inc.
- Jeff Bobzin - Insyde Software Corp.
- Patrick Boyd - Dell Inc.
- David Brockhaus - Vertiv
- Richard Brunner - VMware Inc.
- Sean Byland - Cray Inc.
- Lee Calcote - Seagate Technology
- Keith Campbell - Lenovo
- P Chandrasekhar - Dell Inc.
- Barbara Craig - Hewlett Packard Enterprise
- Chris Davenport - Hewlett Packard Enterprise
- Gamma Dean - Vertiv
- Daniel Dufresne - Dell Inc.
- Samer El-Haj-Mahmoud - Arm Limited, Lenovo, Hewlett Packard Enterprise
- George Ericson - Dell Inc.
- Wassim Fayed - Microsoft Corporation
- Kevin Ferguson - Vertiv
- Mike Garrett - Hewlett Packard Enterprise
- Steve Geffin - Vertiv
- Joe Handzik - Hewlett Packard Enterprise
- Jon Hass - Dell Inc.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

- Jeff Hilland - Hewlett Packard Enterprise
- Chris Hoffman - Vertiv
- Cactus Jiang - Vertiv
- Barry Kittner - Intel Corporation
- Steven Krig - Intel Corporation
- Jennifer Lee - Intel Corporation
- John Leung - Intel Corporation
- Magnus Lundmark - Ericsson AB
- Steve Lyle - Hewlett Packard Enterprise
- Gunnar Mills - IBM
- Jagan Molleti - Dell Inc.
- Milena Natanov - Microsoft Corporation
- Scott Phuong - Cisco Systems, Inc.
- Michael Pizzo - Microsoft Corporation
- Chris Poblete - Dell Inc.
- Michael Raineri - Dell Inc.
- Joseph Reynolds - IBM
- Irina Salvan - Microsoft Corporation
- Bill Scherer - Hewlett Packard Enterprise
- Hemal Shah - Broadcom Inc.
- Jim Shelton - Vertiv
- Tom Slaight - Intel Corporation
- Josiah Smith - Eaton
- Donnie Sturgeon - Vertiv
- Pawel Szymanski - Intel Corporation
- Paul Vancil - Dell Inc.
- Joseph White - Dell Inc.
- Linda Wu - NVIDIA Corporation, Super Micro Computer, Inc.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

2 Introduction {-}

Redfish is a standard that uses RESTful interface semantics to access a schema based data model to conduct management operations. It is suitable for a wide range of devices, from stand-alone servers, to composable infrastructures, and to large-scale cloud environments.

The initial Redfish scope targeted servers. The DMTF and its alliance partners expanded that scope to cover most data center IT equipment and other solutions, and both in- and out-of-band access methods.

Additionally, the DMTF and other organizations that use Redfish as part of their industry standard or solution have added educational material.

This document defines the RESTful interface protocol and the various concepts and services necessary to implement a Redfish interface. The definition of the schema based data model and standard messages for the Redfish interface are covered separately in the following documents:

- DMTF DSP8010, *Redfish Schema Bundle*, https://www.dmtf.org/dsp/DSP8010 contains the individual schema definition files in multiple schema description languages.
- DMTF DSP0268, *Redfish Schema Supplement*, https://www.dmtf.org/dsp/DSP0268 contains the normative descriptions and example payloads for all standard Redfish schema in a single reference guide.
- DMTF DSP8011, *Redfish Standard Registries Bundle*, https://www.dmtf.org/dsp/DSP8011 contains the message registries used for error reporting and event messages.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

3 Scope

This specification defines the required protocols, data model, behaviors, and other architectural components for an interoperable, multivendor, remote, and out-of-band capable interface. This interface meets the cloud-based and web-based IT professionals' expectations for scalable platform management. While large and hyperscale environments are the primary focus, clients can use the specification for individual system management.

The specification defines the required elements for all Redfish implementations, and the optional elements that system vendors and manufacturers can choose. This specification also defines at which points an implementation can provide OEM-specific extensions.

The specification sets normative requirements for *Redfish services* and associated materials, such as Redfish schema files. In general, the specification does not set requirements for Redfish clients but indicates how a client can successfully and effectively access and use a Redfish service.

The specification does not require that implementations of the Redfish interfaces and functions require particular hardware or firmware.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

4 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- DMTF DSP0270, *Redfish Host Interface Specification*, https://www.dmtf.org/sites/default/files/standards/documents/DSP0270_1.0.0.pdf
- Redfish Schema: RedfishExtensions v1.0.0, https://redfish.dmtf.org/schemas/v1/RedfishExtensions_v1.xml
- *Transport Layer Security (TLS) Parameters*, https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml
- *JSON Schema: A Media Type for Describing JSON Documents draft-handrews-json-schema-01*, https://tools.ietf.org/html/draft-handrews-json-schema-01
- *JSON Schema Validation: A Vocabulary for Structural Validation of JSON draft-handrews-json-schema-validation-01*, https://tools.ietf.org/html/draft-handrews-json-schema-validation-01
- IETF RFC1738, T. Berners-Lee et al, *Uniform Resource Locators (URL)*, https://tools.ietf.org/html/rfc1738
- IETF RFC3986, T. Berners-Lee et al, *Uniform Resource Identifier (URI): Generic Syntax*, https://tools.ietf.org/html/rfc3986
- IETF RFC5280, D. Cooper et al, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*,
- IETF RFC6585, M. Nottingham et al, *Additional HTTP Status Codes*, https://tools.ietf.org/html/rfc6585
- IETF RFC6901, P. Bryan, Ed. et al, *JavaScript Object Notation (JSON) Pointer*, https://tools.ietf.org/html/rfc6901
- IETF RFC7230, R. Fielding et al, *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*, https://tools.ietf.org/html/rfc7230
- IETF RFC7231, R. Fielding et al, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*, https://tools.ietf.org/html/rfc7231
- IETF RFC7232, R. Fielding et al, *Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests*, https://tools.ietf.org/html/rfc7232
- IETF RFC7234, R. Fielding et al, *Hypertext Transfer Protocol (HTTP/1.1): Caching*, https://tools.ietf.org/html/rfc7234
- IETF RFC7525, Y. Sheffer et al, *Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, https://tools.ietf.org/html/rfc7525
- IETF RFC7578, L. Masinter et al, *Returning Values from Forms: multipart/form-data*, https://tools.ietf.org/html/rfc7578
- IETF RFC7617, J. Reschke et al, *The 'Basic' HTTP Authentication Scheme*, https://tools.ietf.org/html/rfc7617
- IETF RFC8259, T. Bray, Ed., *The JavaScript Object Notation (JSON) Data Interchange Format*, https://tools.ietf.org/html/rfc7617
- IETF RFC8288, M. Nottingham, *Web Linking*, https://tools.ietf.org/html/rfc8288

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

- ISO 639-1:2002, *Codes for the representation of names of languages - Part 1: Alpha-2 code*, https://www.iso.org/standard/22109.html
- 24 February 2014, *OData Version 4.0 Part 1: Protocol*, https://docs.oasis-open.org/odata/odata/v4.0/os/part1-protocol/odata-v4.0-os-part1-protocol.html
- 24 February 2014, *OData Version 4.0 Part 3: Common Schema Definition Language (CSDL)*, https://docs.oasis-open.org/odata/odata/v4.0/os/part3-csdl/odata-v4.0-os-part3-csdl.html
- 10 March 2016, *OData Version 4.0 Plus Errata 03 OASIS Standard incorporating Draft 01 of Errata 03*, https://docs.oasis-open.org/odata/odata/v4.0/errata03/csd01/complete/vocabularies/Org.OData.Measures.V1.xml
- 20 November 2014, SNIA *TLS Specification for Storage Systems*, https://www.snia.org/tech_activities/standards/curr_standards/tls
- *The OpenAPI Specification*, https://swagger.io/specification/
- *The Unified Code for Units of Measure*, https://ucum.org/ucum.html
- 24 December 2020, *Fetch Living Standard*, https://fetch.spec.whatwg.org/
- *9.2 Server-sent events* in the *HTML Living Standard*, https://html.spec.whatwg.org/multipage/server-sent-events.html

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

5 Terms, definitions, symbols, and abbreviated terms

Some terms and phrases in this document have specific meanings beyond their typical English meanings. This clause defines those terms and phrases.

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 term "deprecated" in this document is to be interpreted as material that is not recommended for use in new development efforts. Existing and new implementations may use this material, but they should move to the favored approach. Deprecated material may be implemented in order to achieve backwards compatibility. Deprecated material should contain references to the last published version that included the deprecated material as normative material and to a description of the favored approach. Deprecated material may be removed from the next major version of the specification.

This document defines these additional terms:

5.1 Hardware terms

5.1.1 baseboard management controller (BMC)

embedded device or service

Note 1 to entry: Typically an independent microprocessor or system-on-chip with associated firmware in a computer system that completes out-of-band systems monitoring and management-related tasks.

5.1.2 IPMI

Intelligent Platform Management Interface

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

5.1.3 KVM-IP

keyboard, video, mouse redirection over IP

5.1.4 NIC

network interface controller

5.1.5 PCI

Peripheral Component Interconnect

5.1.6 PCIe

Peripheral Component Interconnect Express

5.2 Web development terms

5.2.1 CORS

cross-origin resource sharing

5.2.2 CRUD

basic **C**reate, **R**ead, **U**pdate, and **D**elete operations that any interface can support

5.2.3 CSRF

cross-site request forgery

5.2.4 event

data structure that corresponds to one or more alerts

5.2.5 excerpt

subset of data that is copied from one resource and presented in another resource

Note 1 to entry: An excerpt provides data in convenient locations without duplication of entire resources.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

5.2.6 HTTP

Hypertext Transfer Protocol

5.2.7 HTTPS

Hypertext Transfer Protocol Secure

Note 1 to entry: TLS secures HTTP.

5.2.8 hypermedia API

API that enables you to navigate through URIs that a service returns

5.2.9 IP

Internet Protocol

5.2.10 JSON

JavaScript Object Notation

5.2.11 member

single resource instance in a resource collection

5.2.12 message

complete HTTP-formatted or HTTPS-formatted request or response

Note 1 to entry: In the REST-based Redfish protocol, every request results in a response.

5.2.13 OData

Open Data Protocol (OData), as defined in OData Version 4.0 Part 1: Protocol

5.2.14 OData service document

resource that provides information about the service root for generic OData clients

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

5.2.15 operation

HTTP `POST`, `GET`, `PUT`, `PATCH`, `HEAD`, and `DELETE` request methods that map to generic CRUD operations

5.2.16 parent resource

parent to another resource if the initial segment of the resource URI is the same as the URI of the other resource, but is at least one level higher

Note 1 to entry: For example, `/redfish/v1/Chassis/A88` is a parent resource of `/redfish/v1/Chassis/A88/Assembly`.

5.2.17 property

name-value pair in a Redfish-defined request or response

Note 1 to entry: A property can be any valid JSON data type.

5.2.18 request

message from a client to a service

5.2.19 response

message from a service to a client in response to a request message

5.2.20 subscription

registration of a destination to receive events

5.2.21 task

representation of a long-running operation

5.2.22 task monitor

opaque service-generated URI that the client who initiates the request can use to monitor an asynchronous operation

5.2.23 TCP

Transmission Control Protocol

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

5.2.24 TLS

Transport Layer Security

5.2.25 XSS

cross-site scripting

5.3 Redfish terms

5.3.1 collection

see resource collection

5.3.2 Redfish client

communicates with a Redfish service and accesses one or more of the service's resources or functions

5.3.3 Redfish protocol

discovers, connects to, and inter-communicates with a Redfish service

5.3.4 Redfish schema

a set of human and machine-readable documents that define Redfish resources using one or more of the supported schema definition languages

5.3.5 Redfish service

implementation of the protocols, resources, and functions that deliver the interface that this specification defines and its associated behaviors for one or more managed systems

Note 1 to entry: Also known as the *service*.

5.3.6 resource

URI-addressable Redfish data structure

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

5.3.7 resource collection

set of similar resources where the number of instances can shrink or grow

5.3.8 resource tree

tree structure of resources accessible through a well-known starting URI

Note 1 to entry: A client can discover the available resources on a Redfish service by following the resource hyperlinks from the base of the tree.

5.3.9 resource type

set of definitions for properties and actions contained within a resource and documented in the Redfish schema files

5.3.10 service root

starting-point resource for locating and accessing the other resources and associated metadata that make up an instance of a Redfish service

5.3.11 subordinate resource

is subordinate to another resource if the initial segment of the resource URI is the same as the URI of the other resource, but is at least one level deeper

Note 1 to entry: For example, `/redfish/v1/Chassis/A88/Assembly` is a subordinate resource of the `Chassis` resource named `A88`.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

6 Typographical conventions

The following typographical convention indicates deprecated material:

DEPRECATED

Deprecated material appears here.

END DEPRECATED

In places where this typographical convention cannot be used, such as tables or figures, the "DEPRECATED" label is used alone.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

7 Overview

Redfish is a management standard that uses a data model representation with a RESTful interface.

Being RESTful, Redfish is easier to use and implement.

Being model-oriented, it can express the relationships between components and the semantics of the Redfish services and components within them. The model is also easy to extend.

By requiring JSON representation, Redfish enables easy integration with programming environments. It is also easy to interpret by humans.

An interoperable Redfish schema defines this model, which is freely available and published in OpenAPI YAML, OData CSDL, and JSON Schema formats.

7.1 Goals

As an architecture, data model, and set of protocols that enable a client to access Redfish services, Redfish has these goals.

Table 1 describes these goals:

Goal	Purpose
Scalable	Can scale on stand-alone machines or racks of equipment.
Flexible	Can implement through existing hardware or entirely as a software service.
Extensible	Can easily add new and vendor-specific capabilities to the data model.
Backward-compatible	Can add capabilities while preserving investments in earlier implementations.
Interoperable	Provides consistent functionality across multiple vendor implementations.
Standards-based	Built on ubiquitous and secure protocols. Leverages other standards where applicable.
Simple	Easy-to-use without the need for highly specialized programming skills or systems knowledge.
Lightweight	Designed to reduce complexity and implementation costs. Minimizes the required footprint for implementations.

: **Table 1 — Redfish goals**

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

7.2 Design tenets

To deliver these goals, Redfish:

- Provides a RESTful interface by using a JSON payload and a data model.
- Separates the protocol from the data model, which enables the independent revision and use of each.
- Specifies versioning rules for protocols and schema.
- Leverages strength of ubiquitous standards where it meets architectural requirements, such as JSON, HTTP, OData, OpenAPI, and the RFCs that this document references.
- Organizes the data model so that it provides clearly demarcated and value-add features in the same payload as standardized items.
- Makes data in payloads as obvious in context as possible.
- Maintains implementation flexibility. Does not tie the interface to any particular underlying implementation or architecture.
- Focuses on widely used capabilities. To avoid complexity, does not add functions that only a small percentage of users value.

7.3 Limitations

Redfish minimizes the need for clients to complete upgrades by using strict versioning and forward-compatibility rules, and separation of the protocols from the data model. However, Redfish does not guarantee that clients never need to update their software. For example, clients might need to upgrade to manage new system or component types, or update the data model.

Interoperable does not mean identical. Many elements of Redfish are optional. Clients should be prepared to discover the optional elements by using the built-in discovery methods.

The *resource tree* reflects the topology of the system and its devices. Consequently, different hardware or device types result in different resource trees, even for identical systems from the same manufacturer. References between resources may result in a graph instead of a tree. Clients that traverse the resource tree should provide logic to avoid infinite loops.

Additionally, not all Redfish resources use simple REST read-and-write semantics. Different use cases may follow other types of client logic. For example, clients cannot simply read user credentials or certificates from one service and write them to another service.

Finally, the hyperlink values between resources and other elements can vary across implementations. Clients should not assume that they can reuse hyperlinks across different Redfish service instances.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

7.4 Additional design background and rationale

7.4.1 REST-based interface

Redfish exposes many service applications as RESTful interfaces. This document defines a RESTful interface.

Redfish defines a RESTful interface because it:

- Enables a lightweight implementation, using fewer layers than previous standards.
- Is a prevalent access method in the industry.
- Is easy to learn, document, and implement in modern programming languages.
- Has a number of development environments and a healthy tooling ecosystem.
- Fits with the design goal of simplicity.
- Equally applies to software application space as it does to embedded environments, which enables convergence and sharing of code within the management ecosystem.
- Adapts well to any data modeling language.
- Has industry-provided security and discovery mechanisms.

7.4.2 Data-oriented

The Redfish data model is developed by focusing on the contents of the payload. By concentrating on the contents of the payload first, Redfish payloads are easily mapped to schema definition languages and encoding types. The data model is defined in various schema languages, including OpenAPI YAML, OData CSDL, and JSON Schema.

7.4.3 Separation of protocol from data model

Redfish separates the protocol operations from the data model and versions the protocol independently from the data model. This enables clients to extend and change the data model as needed without requiring the protocol version to change.

7.4.4 Hypermedia API service root

Redfish has a single service root URI and clients can discover all resources through referenced URIs. The *hypermedia API* enables the discovery of resources through hyperlinks.

7.4.5 OpenAPI v3.0 support

The OpenAPI v3.0 provides a rich ecosystem of tools for using RESTful interfaces that meet the design requirements of that specification. Starting with *Redfish Specification v1.6.0*, the Redfish schemas support the OpenAPI YAML file format and URI patterns that conform to the OpenAPI Specification were defined. Conforming Redfish services that

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

support the Redfish protocol version v1.6.0 or later implement those URI patterns to enable use of the OpenAPI ecosystem.

For details, see *OpenAPI Specification v3.0*.

7.4.6 OData conventions

With the popularity of RESTful APIs, there are nearly as many RESTful interfaces as there are applications. While following REST patterns helps promote good practices, due to design differences between the many RESTful APIs there few common conventions between them.

To provide for interoperability between APIs, OData defines a set of common RESTful conventions and annotations. Redfish follows OData conventions for describing schema, URL conventions, and definitions for typical properties in a JSON payload.

7.5 Service elements

7.5.1 Synchronous and asynchronous operation support

Some operations can take more time than a client typically wants to wait. For this reason, some operations can be asynchronous at the discretion of the service. The request portion of an asynchronous operation is no different from the request portion of a synchronous operation.

To determine whether an operation was completed synchronously or asynchronously, clients can review the HTTP status codes. For more information, see the Asynchronous operations clause.

7.5.2 Eventing mechanism

Redfish enables clients to receive messages outside the normal request and response paradigm. The service uses these messages, or *events*, to asynchronously notify the client of a state change or error condition, usually of a time critical nature.

This specification defines two styles of eventing:

- Push-style eventing.

  When the service detects the need to send an event, it calls HTTP `POST` to push the event message to the client. Clients can enable reception of events by creating a *subscription* entry in the event service, or an administrator can create subscriptions as part of the Redfish service configuration.

- Server-sent events (SSE)-style eventing.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

The client opens an SSE connection to the service through a `GET` on the `ServerSentEventUri`-specified URI in the event service.

For information, see the Eventing clause.

7.5.3 Actions

Actions are Redfish operations that do not easily map to RESTful interface semantics. These types of operations may not directly affect properties in the Redfish resources. The Redfish schema defines certain standard actions for common Redfish resources. For these standard actions, the Redfish schema contains the normative language on the behavior of the action.

7.5.4 Service discovery

While the service itself is at a well-known URI, clients need to discover the network address of the service. Like UPnP, Redfish uses SSDP for discovery. A wide variety of devices, such as printers and client operating systems, support SSDP. It is simple, lightweight, IPv6 capable, and suitable for implementation in embedded environments.

For more information, see the Discovery clause.

7.5.5 Remote access support

Remote management functionality typically includes access mechanisms for redirecting operator interfaces such as serial console, keyboard video and mouse (KVM-IP), command shell, or command-line interface, and virtual media. While these mechanisms are critical functionality, they cannot be reasonably implemented as a RESTful interface.

Therefore, this standard does not define the protocols or access mechanisms for those services but encourages implementations that leverage existing standards. However, the Redfish schema includes resources and properties that enable client discovery of these capabilities and access mechanisms to enable interoperability.

7.6 Security

The challenge of remote interface security is to protect both the interface and exchanged data. To accomplish this, Redfish provides authentication and encryption. As part of this security, Redfish defines and requires minimum levels of encryption.

For more information, see the Security details clause.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

8 Protocol details

In this document, the Redfish protocol refers to the RESTful mapping to HTTP, TCP/IP, and other protocol, transport, and messaging layer aspects. HTTP is the application protocol that transports the messages and TCP/IP is the transport protocol. The RESTful interface is a mapping to the message protocol.

The Redfish protocol is designed around a web service-based interface model, which provides network and interaction efficiency for both user interface (UI) and automation usage. Specifically, the protocol can leverage existing tool chains.

Table 2 describes the items that the Redfish protocol uses:

Item	Description
HTTP methods	Maps to common CRUD operations.
Actions	Expands operations beyond CRUD-type operations.
Media types	Negotiates the type of data sent in the message body.
HTTP status codes	Indicates the success or failure of the server's request.
Error responses	Returns more information than HTTP status codes.
TLS	Secures messages. See Security details.
Asynchronous semantics	Manages long-running operations.

: **Table 2 — Redfish protocol**

A Redfish interface shall be exposed through a web service endpoint implemented by using HTTP version 1.1. See RFC7230, RFC7231, and RFC7232.

The subsequent clauses describe how the Redfish interface uses and adds constraints to HTTP to ensure interoperability of Redfish implementations.

8.1 Universal Resource Identifiers

A Universal Resource Identifier (URI) identifies a resource, including the service root and all Redfish resources.

- A URI shall identify each unique instance of a resource.
- URIs shall not include any RFC1738-defined unsafe characters.
  - For example, the `{`, `}`, `<`, `|`, `^`, `~`, `[`, `]`, `` ` ``, and `\` characters are unsafe because gateways and other transport agents can sometimes modify these characters.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

  - Do not use the `#` character for anything other than the start of a fragment.
- URIs shall not include any percent-encoding of characters. This restriction does not apply to the query parameters portion of the URI.

A `GET` operation on a URI returns a representation of the resource with properties and hyperlinks to associated resources. The service root URI is well known and is based on the protocol version.

To discover the URIs to additional resources, extract the associated resource hyperlinks from earlier responses. The *hypermedia API* enables the discovery of resources through hyperlinks.

Redfish considers the RFC3986-defined scheme, authority, service root, and version, and unique resource path component parts of the URI.

For example, this URI:

```
https://mgmt.vendor.com/redfish/v1/Systems/1
```

Contains these component parts:

- `https:` is the scheme.
- `mgmt.vendor.com` is the authority to which to delegate the URI.
- `redfish/v1` is the service root and version.
- `Systems/1` is the unique resource path.

In a URI:

- The scheme and authority component parts are not part of the unique resource path because redirection capabilities and local operations may cause the connection portion to vary.
- The service root and resource path component parts *uniquely identify* the resource in a Redfish service.

In an implementation:

- The resource path component part shall be unique.
- A relative reference in the body and HTTP headers payload can identify a resource in that same implementation.
- An absolute URI in the body and HTTP headers payload can identify a resource in a different implementation.

For the absolute URI definition, see RFC3986.

For example, a `POST` operation may return the `/redfish/v1/Systems/2` URI in the `Location` header of the response, which points to the `POST`-created resource.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

Assuming that the client connects through the `mgmt.vendor.com` appliance, the client accesses the resource through the `https://mgmt.vendor.com/redfish/v1/Systems/2` absolute URI.

URIs that conform to RFC3986 may also contain the query, `?query`, and frag, `#frag`, components. For information about queries, see Query parameters. When a URI includes a fragment (`frag`) to submit an operation, the server ignores the fragment.

If a property in a response references another property within a resource, use the RFC6901-defined URI fragment identifier representation format. If the property is a reference property in the schema, the fragment shall reference a valid resource identifier. For example, the following fragment identifies a property at index 0 of the `Fans` array in the `/redfish/v1/Chassis/MultiBladeEncl/Thermal` resource:

```
{
   "@odata.id": "/redfish/v1/Chassis/MultiBladeEncl/Thermal#/Fans/0"
}
```

For requirements on constructing Redfish URIs, see the resource URI patterns annotation clause.

8.2 HTTP methods

Table 3 describes the mapping of HTTP methods to the Redfish-supported operations. If the **Required** column contains **Yes**, a Redfish interface shall support the HTTP method. If the **Required** column contains **No**, a Redfish interface may support the HTTP method.

HTTP method	Interface semantic	Required
`POST`	Create resource Resource action Eventing	Yes
`GET`	Retrieve resource	Yes
`PUT`	Replace resource	No
`PATCH`	Update resource	Yes
`DELETE`	Delete resource	Yes
`HEAD`	Retrieve resource header	No
`OPTIONS`	Retrieve header Cross-origin resource sharing (CORS) preflight	No

: **Table 3 — Mapping of HTTP methods to Redfish-supported operations**

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

For HTTP methods that the Redfish service does not support or that Table 3 omits, the Redfish service shall return the HTTP `405 Method Not Allowed` status code or the HTTP `501 Not Implemented` status code.

8.3 HTTP redirect

HTTP redirect enables a service to redirect a request to another URL. Among other things, HTTP redirect enables Redfish resources to alias areas of the data model.

All Redfish clients shall correctly handle HTTP redirect.

The service for the redirected resource shall enforce the authentication and authorization requirements for the redirected resource.

8.4 Media types

Some resources may be available in more than one type of representation. The media type indicates the representation type.

In HTTP messages, the media type is specified in the `Content-Type` header. To tell a service to return the response through certain media types, the client sets the HTTP `Accept` header to a list of the media types.

- All resources shall be available through the JSON `application/json` media type.
- Redfish services shall make every resource available in a JSON-based representation as a JSON object, as specified in RFC8259. Receivers shall not reject a JSON-encoded message, and shall offer at least one JSON-based response representation. An implementation may offer additional non-JSON media type representations.

To request compression in the response body, clients specify an `Accept-Encoding` request header.

8.5 ETags

To reduce unnecessary RESTful accesses to resources, the Redfish service should support the association of a separate entity tag (ETag) with each resource.

- Implementations should support the return of ETag properties for each resource.
- Implementations should support the return of ETag headers for each single-resource response.
- Implementations shall support the return of ETag headers for `GET` requests of `ManagerAccount` resources.

Because the service knows whether the new version of the object is substantially different, the service generates and provides the ETag as part of the resource payload.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

The ETag mechanism supports both ***strong*** and ***weak*** validation. If a resource supports an ETag, it shall use the RFC7232-defined ETag.

This specification does not mandate a particular algorithm for ETag creation, but ETags should be highly collision-free.

An ETag can be:

- A hash
- A generation ID
- A time stamp
- Some other value that changes when the underlying object changes

If a client calls `PUT` or `PATCH` to update a resource, it should include an ETag from a previous `GET` in the HTTP `If-Match` or `If-None-Match` header. If a service supports the return of the ETag header on a resource, it may respond with the HTTP `428 Precondition Required` status code if the `If-Match` or `If-None-Match` header is missing from the `PUT` or `PATCH` request for the same resource, as specified in RFC6585.

In addition to the return of the ETag property on each resource, a Redfish service should return the ETag header on:

- A client `PUT`, `POST`, or `PATCH` operation
- A `GET` operation for an individual resource

The format of the ETag header is:

```
ETag: <string>
```

8.6 Protocol version

The protocol version is separate from the resources' version or the Redfish schema version that the resources support.

Each Redfish protocol version is strongly typed by using the URI of the Redfish service in combination with the resource obtained at that URI, called the `ServiceRoot` resource.

The root URI for this version of the Redfish protocol shall be `/redfish/v1/`.

The URI defines the major version of the protocol.

The `RedfishVersion` property of the `ServiceRoot` resource defines the protocol version, which includes the major version, minor version, and errata version of the protocol, as defined in the Redfish schema for that resource.

The protocol version is a string in the format:

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

`<MajorVersion>.<MinorVersion>.<ErrataVersion>`

where

- `<MajorVersion>` is an integer that represents the major version. Indicates a backward-incompatible change.
- `<MinorVersion>` is an integer that represents the minor version. Indicates a minor update. Redfish introduces functionality but does not remove any functionality. The minor version preserves compatibility with earlier minor versions.
- `<ErrataVersion>` is an integer that represents the errata version. Indicates a fix to the earlier version.

Any resource that a client discovers through hyperlinks that the service root or any service root-referenced service or resource returns shall conform to the same protocol version that the service root supports.

A `GET` operation on the `/redfish` resource shall return this response body:

```
{
   "v1": "/redfish/v1/"
}
```

8.7 Redfish-defined URIs and relative reference rules

Table 4 describes the Redfish-defined URIs that a Redfish service shall support:

URI	Returns	Note
`/redfish`	Version. A major update that does not preserve compatibility with earlier minor versions.	Services shall support this URI.
`/redfish/v1/`	Redfish service root.	Services shall support this URI.
`/redfish/v1/odata`	Redfish OData service document.	Services shall support this URI.
`/redfish/v1/$metadata`	Redfish metadata document.	Services shall support this URI.
`/redfish/v1/openapi.yaml`	Redfish OpenAPI YAML document.	Services should support this URI.

: **Table 4 — Redfish-defined URIs**

In addition, Table 5 describes the URIs that services shall process without a trailing slash in one of these ways:

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

- Redirect it to the associated Redfish-defined URI.
- Treat it as the equivalent URI to the associated Redfish-defined URI.

URI	Associated Redfish-defined URI
`/redfish/v1`	`/redfish/v1/`
`/redfish/`	`/redfish`

: **Table 5 — Redfish-defined URIs without trailing slashes**

All other Redfish service-supported URIs shall match the resource URI patterns definitions, except the supplemental resources that the `@Redfish.Settings`, `@Redfish.ActionInfo`, and `@Redfish.CollectionCapabilities` payload annotations reference. The client shall treat the URIs for these supplemental resources as opaque.

All Redfish-defined URIs and URIs starting with `/redfish` are reserved for future standardization by DMTF and DMTF alliance partners, except OEM extension URIs, which shall conform to the requirements of the OEM URIs clause.

All relative references that the service uses shall start with either:

- A double forward slash (`//`) and include the authority (network-path), such as `//mgmt.vendor.com/redfish/v1/Systems`.
- A single forward slash (`/`) and include the absolute-path, such as `/redfish/v1/Systems`.

For details, see RFC3986.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

9 Service requests

This clause describes the requests that clients can send to Redfish services.

9.1 Request headers

The *HTTP Specification* defines headers for request messages. Table 6 defines those headers and their requirements for Redfish services and clients.

For Redfish services:

- Redfish services shall process the headers in Table 6 as defined by the *HTTP 1.1 Specification* if the **Service requirement** column contains **Yes** or **Conditional**.
- Redfish services should process the headers in Table 6 and Table 7 as defined by the *HTTP 1.1 Specification* if the **Service requirement** column contains **No**.

For Redfish clients (sending the HTTP requests):

- Redfish clients shall include the headers in Table 6 as defined by the *HTTP 1.1 Specification* if the **Client requirement** column contains **Yes** or **Conditional**.
- Redfish clients should transmit the headers in Table 6 and Table 7 as defined by the *HTTP 1.1 Specification* if the **Client requirement** column contains **No**.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

Header	Service requirement	Client requirement	Supported values	Description
`Accept`	Yes	No	RFC7231	Communicates to the server the media type or types that this client is prepared to accept. Services shall support resource requests with `Accept` header values of `application/json` or `application/json;charset=utf-8`. Services shall support XML metadata requests with `Accept` header values of `application/xml` or `application/xml;charset=utf-8`. Services shall support OpenAPI YAML schema requests with `Accept` header values of `application/yaml` or `application/yaml;charset=utf-8` or `application/vnd.oai.openapi` or `application/vnd.oai.openapi;charset=utf-8`. Services shall support SSE requests with `Accept` header values of `text/event-stream` or `text/event-stream;charset=utf-8`. Services shall support any request with `Accept` header values of `application/*`, `application/*;charset=utf-8`, `*/*`, or `*/*;charset=utf-8`.
`Accept-Encoding`	No	No	RFC7231	Indicates whether the client can handle gzip-encoded responses. If a service cannot return an acceptable response to a request with this header, it shall respond with the HTTP `406 Not Acceptable` status code. If the request omits this header, the service should not return gzip-encoded responses.
`Accept-Language`	No	No	RFC7231	The languages that the client accepts in the response. If the request omits this header, uses the service's default language for the response.
`Authorization`	Conditional	Conditional	RFC7617	Required for HTTP basic authentication. A client can access unsecured resources without this header on systems that support basic authentication.
`Content-Length`	No	No	RFC7231	The size of the message body. To indicate the size of the body, a client can use the `Transfer-Encoding: chunked` header. If a service needs to use `Content-Length` and does not support `Transfer-Encoding`, it responds with the HTTP `406 Not Acceptable` status code.
`Content-Type`	Conditional	Conditional	RFC7231	The request format. Required for operations with a request body. Services shall accept the `Content-Type` header set to either `application/json` or `application/json;charset=utf-8`. It is recommended that clients use these values in requests because other values can cause an error.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

Header	Service requirement	Client requirement	Supported values	Description
`Host`	Yes	No	RFC7230	Enables support of multiple origin hosts at a single IP address.
`If-Match`	Conditional	No	RFC7232	To ensure that clients update the resource from a known state, `PUT` and `PATCH` requests for resources for which a service returns ETags shall support `If-Match`. While not required for clients, it is highly recommended for `PUT` and `PATCH` operations.
`If-None-Match`	No	No	RFC7232	A service only returns the resource if the current ETag of that resource does not match the ETag sent in this header. If the ETag in this header matches the resource's current ETag, the `GET` operation returns the HTTP `304 Not Modified` status code.
`Last-Event-ID`	No	No	HTML5 SSE	The event source's last `id` field from the SSE stream. Requests history event data. See Server-sent events.
`Max-Forwards`	No	No	RFC7231	Limits gateway and proxy hops. Prevents messages from remaining in the network indefinitely.
`OData-MaxVersion`	No	No	4.0	The maximum OData version that an OData-aware client understands.
`OData-Version`	Yes	No	4.0	The OData version. Services shall reject requests that specify an unsupported OData version. If a service encounters an unsupported OData version, it should reject the request with the HTTP `412 Precondition Failed` status code.
`Origin`	Yes	No	Fetch Living Standard, 3.1. `Origin` header	Enables web applications to consume a Redfish service while preventing CSRF attacks.
`User-Agent`	Yes	No	RFC7231	Traces product tokens and their versions. The header can list multiple product tokens.
`Via`	No	No	RFC7230	Defines the network hierarchy and recognizes message loops. Each pass inserts its own `Via` header.

: **Table 6 — Request headers**

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

Redfish services shall understand and be able to process the headers in Table 7 as defined by this specification if the **Service requirement** column contains **Yes**.

Header	Service requirement	Client requirement	Supported values	Description
`X-Auth-Token`	Yes	Conditional	Opaque encoded octet strings	Authenticates user sessions. The token value shall be indistinguishable from random. While services shall support this header, a client can access unsecured resources without establishing a session.

: **Table 7 — Request headers part 2**

9.2 GET (read requests)

9.2.1 GET (read requests) overview

The `GET` operation retrieves resources from a Redfish service. Clients make a `GET` request to the individual resource URI. Clients may obtain the resource URI from published sources, such as the OpenAPI document, or from a resource identifier property in a previously retrieved resource response, such as the links property.

The service shall return the resource representation using one of the media types listed in the `Accept` header, subject to the requirements of the media types. If the `Accept` header is absent, the service shall return the resource's representation as `application/json`. Services may but are not required to support the convention of retrieving individual properties within a resource by appending a segment containing the property name to the URI of the resource.

- The HTTP `GET` operation shall retrieve a resource without causing any side effects.
- The service shall ignore the content of the body on a `GET`.
- The `GET` operation shall be idempotent in the absence of outside changes to the resource.

If supported by the service, clients can perform a conditional `GET` operation by specifying an `If-None-Match` request header that contains the ETag of the resource.

9.2.2 Resource collection requests

Clients retrieve a resource collection by making a `GET` request to the resource collection URI. The response includes the resource collection's properties and an array of its *members*.

No requirements are placed on implementations to return a consistent set of members when a series of requests that use paging query parameters are made over time to obtain the entire set of members. These calls can result in missed or duplicate elements if multiple `GET` requests use paging to retrieve the `Members` array instances.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

- Clients shall not make assumptions about the URIs for the members of a resource collection.
- Retrieved resource collections shall always include the count (`Members@odata.count`) property to specify the total number of entries in its `Members` array.
- Regardless of the next link (`Members@odata.nextLink`) property or paging, the count (`Members@odata.count`) property shall return the total number of resources that the `Members` array references.

A subset of the members can be retrieved using client paging query parameters.

A service might not be able to return all of the contents of a resource collection request in a single response body. In this case, the response can be paged by the service. If a service pages a response to a resource collection request, the following rules shall apply:

- Responses can contain a subset of the full resource collection's members.
- Individual members shall not be split across response bodies.
- A next link (`Members@odata.nextLink`) property annotation shall be supplied in the response body with the URI to the next set of members in the collection.
- The next link (`Members@odata.nextLink`) property shall adhere to the rules in the Next link property clause.
- `GET` operations on the next link (`Members@odata.nextLink`) property shall return the subsequent section of the resource collection response.

9.2.3 Service root request

The root URL for Redfish version 1.x services shall be `/redfish/v1/` .

The service returns the `ServiceRoot` resource, as defined by this specification, as a response for the root URL.

Services shall not require authentication to retrieve the service root and `/redfish` resources.

9.2.4 OData service and metadata document requests

Redfish services expose two OData-defined documents at specific URIs to enable generic OData clients to navigate the Redfish service.

- Service shall expose an OData metadata document at the `/redfish/v1/$metadata` URI.
- Service shall expose an OData service document at the `/redfish/v1/odata` URI.
- Service shall not require authentication to retrieve the OData metadata document or the OData service document.

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

9.3 Query parameters

9.3.1 Query parameter overview

To paginate, retrieve subsets of resources, or expand the results in a single response, clients can include the query parameters. Some query parameters apply only to resource collections.

Services:

- Shall only support query parameters on `GET` operations.
- Should support the `$top`, `$skip`, `only`, and `excerpt` query parameters.
- May support the `$expand`, `$filter`, and `$select` query parameters.
- Shall include the `ProtocolFeaturesSupported` object in the service root, if the service supports query parameters.
  - This object indicates which parameters and options have been implemented.
- Shall ignore unknown or unsupported query parameters that do not begin with `$`.
- Shall use the `&` operator to separate multiple query parameters in a single request.

Services shall return:

- The HTTP `501 Not Implemented` status code for any unsupported query parameters that start with `$`.
- An extended error that indicates the unsupported query parameters for this resource.
- The HTTP `400 Bad Request` status code for any query parameters that contain values that are invalid, or values applied to query parameters without defined values, such as `excerpt` or `only`.

Services should return:

- The HTTP `400 Bad Request` status code with the `QueryNotSupportedOnResource` message from the Base Message Registry for any implemented query parameters that are not supported on a resource in the request.
- The HTTP `400 Bad Request` status code with the `QueryNotSupportedOnResource` message from the Base Message Registry for any supported query parameters that apply only to resource collections but are used on singular resources. This includes query parameters such as `$filter`, `$top`, `$skip`, and `only`.
- The HTTP `400 Bad Request` status code with the `QueryNotSupportedOnOperation` message from the Base Message Registry for any supported query parameters on operations other than `GET`.

Services shall process query parameters in this order:

- `$filter`
- `$skip`
- `$top`

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

- Apply server-side pagination
- `$expand`
- `excerpt`
- `$select`

Table 8 describes the query parameters:

Query parameter	Description and example
`excerpt`	Returns a subset of the resource's properties that match the defined `Excerpt` schema annotation. If no `Excerpt` schema annotation is defined for the resource, the entire resource is returned. **Example:** `https://resource?excerpt`

`$expand=<string>`	Returns a hyperlink and its contents in-line with retrieved resources, as if a `GET` call response was included in-line with that hyperlink.
	
	See The $expand query parameter.
	
	Example:
	
	`https://resource?$expand=*($levels=3)`
	
	`https://resourcecollection?$expand=.($levels=1)`

`$filter=<string>`	Applies to resource collections. Returns a subset of collection members that match the `$filter` expression. See The $filter query parameter. **Example:** `https://resourcecollection?$filter=SystemType eq 'Physical'`
`only`	Applies to resource collections. If the target resource collection contains exactly one member, clients can use this query parameter to return that member's resource. If the collection contains either zero members or more than one member, the response returns the resource collection, as expected. Services should return the HTTP `400 Bad Request` with the `QueryCombinationInvalid` message from the Base Message Registry if `only` is being combined with other query parameters. **Example:** `https://resourcecollection?only`

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

Query parameter	Description and example
`$select=<string>`	Returns a subset of the resource's properties that match the `$select` expression. See The $select query parameter. **Example:** `https://resource?$select=SystemType,Status`
`$skip=<integer>`	Applies to resource collections. Returns a subset of the members in a resource collection, or an empty set of members if the `$skip` value is greater than or equal to the member count. This paging query parameter defines the number of members in the resource collection to skip. **Example:** `https://resourcecollection?$skip=5`
`$top=<integer>`	Applies to resource collections. Defines the number of members to show in the response. Minimum value is `0`, though a value of `0` returns an empty set of members. **Example:** `https://resourcecollection?$top=30`

: **Table 8 — Query parameters**

9.3.2 The $expand query parameter

The `$expand` query parameter enables a client to request a response that includes not only the requested resource, but also includes the contents of the subordinate or hyperlinked resources. The definition of this query parameter follows the *OData Protocol Specification*.

The `$expand` query parameter has a set of possible options that determine which hyperlinks in a resource are included in the expanded response. Some resources may already be expanded due to the resource's schema annotation `AutoExpand`, such as the `Temperature` object in the `Thermal` resource.

Table 9 describes the Redfish-supported options for the `$expand` query parameter. The service may implement some of these options but not others. Any other supported syntax for `$expand` is outside the scope of this specification.

Option	Description	Example
asterisk (`*`)	Shall expand all hyperlinks, including those in payload annotations, such as `@Redfish.Settings`, `@Redfish.ActionInfo`, and `@Redfish.CollectionCapabilities`.	`https://resource?$expand=*`

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

Option	Description	Example
`$levels`	Number of levels the service should cascade the `$expand` operation. The default level shall be 1. For example, `$levels=2` expands both the hyperlinks in the current resource (level 1), and the hyperlinks in the resulting expanded resources (level 2).	`https://resourcecollection?$expand=.($levels=2)`
period (`.`)	Shall expand all hyperlinks **not** in any links property instances of the resource, including those in payload annotations, such as `@Redfish.Settings`, `@Redfish.ActionInfo`, and `@Redfish.CollectionCapabilities`.	`https://resourcecollection?$expand=.`
tilde (`~`)	Shall expand all hyperlinks found in all links property instances of the resource.	`https://resourcecollection?$expand=~`

: **Table 9 — The $expand query parameter options**

Examples of `$expand` usage include:

- `GET` of a `SoftwareInventoryCollection`.

  With `$expand`, the client can request multiple `SoftwareInventory` collection member resources in one request rather than fetching them one at a time.

- `GET` of a `ComputerSystem`.

  With `$levels`, a single `GET` request can include the subordinate resource collections, such as `Processors` and `Memory`.

- `GET` all UUIDs in members of the `ComputerSystem` collection.

  To accomplish this result, include both `$select` and `$expand` on the URI.

  The syntax is `GET /redfish/v1/Systems?$select=UUID&$expand=.($levels=1)`

When services execute `$expand`, they may omit some of the referenced resource's properties.

When clients use `$expand`, they should be aware that the payload may increase beyond what can be sent in a single response.

If a service cannot return the payload due to its size, it shall return the HTTP `507 Insufficient Storage` status code.

If a service cannot return the payload corresponding to an individual member of a resource collection, it should

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

return the `@odata.id` property for that member and should return extended information indicating the reason that member was not returned, such as when a provider internal to the service returns an error or times out.

The following example expands the `RoleCollection` resource with the level set to 1:

```
{
   "@odata.id": "/redfish/v1/AccountService/Roles",
   "@odata.type": "#RoleCollection.RoleCollection",
   "Name": "Roles Collection",
   "Members@odata.count": 3,
   "Members": [{
      "@odata.id": "/redfish/v1/AccountService/Roles/Administrator",
      "@odata.type": "#Role.v1_1_0.Role",
      "Id": "Administrator",
      "Name": "User Role",
      "Description": "Admin User Role",
      "IsPredefined": true,
      "AssignedPrivileges": ["Login", "ConfigureManager",
         "ConfigureUsers", "ConfigureSelf", "ConfigureComponents"]
   }, {
      "@odata.id": "/redfish/v1/AccountService/Roles/Operator",
      "@odata.type": "#Role.v1_1_0.Role",
      "Id": "Operator",
      "Name": "User Role",
      "Description": "Operator User Role",
      "IsPredefined": true,
      "AssignedPrivileges": ["Login", "ConfigureSelf",
         "ConfigureComponents"]
   }, {
      "@odata.id": "/redfish/v1/AccountService/Roles/ReadOnly",
      "@odata.type": "#Role.v1_1_0.Role",
      "Id": "ReadOnly",
      "Name": "User Role",
      "Description": "ReadOnly User Role",
      "IsPredefined": true,
      "AssignedPrivileges": ["Login", "ConfigureSelf"]
   }]
}
```

9.3.3 The $select query parameter

The `$select` query parameter indicates that the implementation should return a subset of the resource's properties that match the `$select` expression. If a request omits the `$select` query parameter, the response returns all properties by default. The definition of this query parameter follows the OData Protocol Specification.

The `$select` expression shall not affect the resource itself.

The `$select` expression defines a comma-separated list of properties to return in the response body.

© ISO/IEC 2022 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022

The syntax for properties in objects or properties in arrays of objects shall be the object and property names concatenated with a slash (`/`).

An example of `$select` usage is:

```
GET /redfish/v1/Systems/1?$select=Name,SystemType,Status/State
```

When services execute `$select`, they shall return all requested properties of the referenced resource. If a requested property is an object, the service shall return the entire object. The `@odata.id` and `@odata.type` properties shall be in the response payload and contain the same values as if `$select` was omitted. If the `@odata.context` property is supported, it shall be in the response payload and should be in the context property recommended format. If the `@odata.etag` property is supported, it shall be in the response payload and contain the same values as if `$select` was omitted.

Any other supported syntax for `$select` is outside the scope of this specification.

9.3.4 The $filter query parameter

The `$filter parameter` enables a client to request a subset of the resource collection's members based on the `$filter` expression. The definition of this query parameter follows the OData Protocol Specification.

The `$filter` query parameter defines a set of properties and literals with an operator.

A literal value can be:

- A string enclosed in single quotes.
- A number.
- A boolean value.

If the literal value does not match the data type for the specified property, the service should reject `$filter` requests with the HTTP `400 Bad Request` status code.

The `$filter` section of the OData ABNF Components Specification contains the grammar for the allowable syntax of the `$filter` query parameter, with the additional restriction that only built-in filter operations are supported.

Table 10 lists the Redfish-supported values for the `$filter` query parameter. Any other supported syntax for `$filter` is outside the scope of this specification.

Value	Description	Example
`()`	Precedence grouping operator.	`(Status/State eq 'Enabled' and Status/Health eq 'OK') or SystemType eq 'Physical'`

© ISO/IEC 2022 – All rights reserved

AVA Documents
Document Sharing Hub

Full document details: BS ISO/IEC 30115-1:2022