文件编号：DSP0266

日期：2017 年 9 月 22 日

版本：1.3.0

Redfish 可扩展平台管理 API 规范

取代版本：1.2.1

文件分级：规范文件

文件状态：已出版

文件语言：中文（中国）

DMTF 是一家非营利性组织，其行业成员的主要目的在于促进企业和系统的管理与互通性。成员和非成员都可以复制 DMTF 的规范和文件，但在引用相关文件时必须明确说明其具体出处。DMTF 规范文件可能会随时进行修改，因此用户必须随时注意相关文件的具体版本号和发行日期。

实施本标准或建议标准中的部分要素可能会涉及到第三方的专利权，包括临时性专利权（简称为“专利权”）。DMTF 不负责向标准用户告知或说明上述专利权，也不负责识别、披露。或者确定任何或所有上述第三方专利权、所有者或索赔人；也不对在确定或者披露上述权利、所有者或索赔人过程中出现的不完整或不准确信息负责。在任何情况下，DMTF 都不对未能识别、披露或者确定上述第三方专利权的行为、也不对相关方依赖标准文件，或者将其纳入到其产品、协议或测试流程中的行为承担任何形式的责任。DMTF 不对实施本标准文件的任何一方承担责任，不论相关实施行为是否可以被预见；也不对任何专利所有者或索赔人承担责任；也不对标准在出版之后被撤销或者进行修改导致的任何成本或损失承担责任。对于实施本标准的任何方而言，如果其在实施标准的过程中侵犯了任何专利权，则其必须针对专利所有者提出的索赔要求向 DMTF 进行赔偿，并保证 DMTF 免受任何不利影响。

更多关于向 DMTF 发出通知，认为其持有的专利权可能与 DMTF 标准相关、或者会影响标准实施的持有专利权的第三方的信息，请参阅下列网站中的相关内容：http://www.dmtf.org/about/policies/disclosures.php。

本标准文件使用的语言为英语，但允许将其翻译为其他语言。

目录

序言

Redfish 可控展平台管理 API（“Redfish”）是由 DMTF 的可扩展平台管理论坛编制的。

DMTF 是一家非营利性组织，其行业成员的主要目的在于促进企业和系统的管理与互通性。

更多关于 DMTF 的信息，请参阅：http://www.dmtf.org。

致谢

DMTF 在此感谢下列人员在本文件编制过程中做出的贡献：

- 杰夫.奥特尔：惠普公司；
- 帕特里克.博伊德：戴尔公司；
- 大卫.布罗克豪斯：艾默生公司；
- 理查德.布伦纳：威睿公司；
- 李.卡尔科特：希捷科技公司；
- P.强德拉塞卡：戴尔公司；
- 克里斯.达文波特：惠普公司；
- 伽马.迪恩：艾默生公司；
- 丹尼尔.杜弗雷斯内：戴尔公司；
- 萨摩尔.埃尔哈吉.穆罕默德：联想公司、惠普公司；
- 乔治.埃里克森：戴尔公司；
- 瓦西姆.费耶德：微软公司；
- 麦克.加雷特：惠普公司；
- 史蒂夫.格芬：艾默生公司；
- 乔.韩德兹克：惠普公司；
- 乔.哈斯：戴尔公司；
- 杰夫.西兰德：惠普公司；
- 克里斯.霍夫曼：艾默生公司；
- 史蒂夫.克里格：英特尔公司；
- 约翰.梁：英特尔公司；
- 贾根.莫来蒂：戴尔公司；
- 米拉娜.纳塔诺夫：微软公司；
- 迈克尔.皮佐：微软公司；
- 克里斯.波夫莱特：戴尔公司；
- 迈克尔. 拉伊内里：戴尔公司；
- 伊琳娜.萨尔凡：微软公司；
- 何马尔.沙哈：博通公司；
- 吉姆.谢尔顿：艾默生公司；
- 汤姆.史莱特：英特尔公司；
- 东尼.斯特金：艾默生公司；
- 鲍威尔.西曼斯基：英特尔公司；
- 保罗.范希尔：戴尔公司；
- 琳达.吴：超微电脑公司。

1. 摘要

Redfish 可扩展平台管理 API（“Redfish”）是一种新的规范，使用 RESTful 界面语义来访问模型格式中定义的数据，并执行带外系统管理。此规范适合于多种范围很广的服务器，包括独立式服务器、机架式服务器、以及刀片服务器，在经过扩展之后，还可以适用于大规模的云环境。

当前行业内存在多种带外系统管理标准（事实上和法律意义上的标准）。所有这些标准要么在实施过程中有着很大变化，是针对嵌入式环境中的单一服务器开发的；要么与预期使用的软件模型 schema 有关。当前，尚没有一种使用简单、立足于新兴的编程标准、对嵌入式友好、且能够满足大型数据中心和云需求的单一的行业标准。

2. 参考标准

在本文件的应用和实施过程中，下列参考文件是必不可少的。对于带有日期或者版本号的参考文件而言，只有被应用的版本适用（包括其勘误表和 DMTF 更新版本在内）。对于不带有日期或者版本号的参考文件而言，最新发布的相关版本的参考文件适用（包括其勘误表和 DMTF 更新版本在内）。

- IETF RFC 3986 T. Berners-Lee et al，统一资源标识符（URI）：通用语法，http://www.ietf.org/rfc/rfc3986.txt；
- IETF RFC 4627, D. Crockford，Javascrip 对象标记（JSON）媒体类型的应用，http://www.ietf.org/rfc/rfc4627.txt；
- IETF RFC 5789, L. Dusseault et al，HTTP 的 PATCH 方法，http://www.ietf.org/rfc/rfc5789.txt；
- IETF RFC 5280, D. Cooper et al，网页链接，http://www.ietf.org/rfc/rfc5280.txt；
- IETF RFC 5988, M. Nottingham，网页链接，http://www.ietf.org/rfc/rfc5988.txt；
- IETF RFC 6585, M. Nottingham, et al，其他 HTTP 状态代码，http://www.ietf.org/rfc/rfc6585.txt；
- IETF RFC 6901, P. Bryan, Ed. et al，Javascrip 对象标记（JSON）指针，http://www.ietf.org/rfc/rfc6901.txt；
- IETF RFC 6906, E. Wilde，“配置”链接关系类型，http://www.ietf.org/rfc/rfc6906.txt；
- IETF RFC 7230, R. Fielding et al.，超文本传输协议（HTTP/1.1）：消息语法和路径选择，http://www.ietf.org/rfc/rfc7230.txt；
- IETF RFC 7231, R. Fielding et al.，超文本传输协议（HTTP/1.1）：语义和内容，

http://www.ietf.org/rfc/rfc7231.txt；

- IETF RFC 7232, R. Fielding et al.，超文本传输协议（HTTP/1.1）：条件请求，http://www.ietf.org/rfc/rfc7232.txt；
- IETF RFC 7234, R. Fielding et al.，超文本传输协议（HTTP/1.1）：缓存，http://www.ietf.org/rfc/rfc7234.txt；
- IETF RFC 7235, R. Fielding et al.，超文本传输协议（HTTP/1.1）：认证，http://www.ietf.org/rfc/rfc7235.txt；
- ISO/IEC 指令，第二部分：国际标准结构和起草规则，http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtypeH；
- JSON schema，重要术语和定义，第 4 稿，http://tools.ietf.org/html/draft-zyp-json-schema-04.txt；
- JSON schema，交互式和非交互式验证，第 4 稿，http://tools.ietf.org/html/draft-fge-json-schema-validation-00.txt；
- OData 4.0 版，第一部分：协议，2014 年 2 月 24 日，http://docs.oasis-open.org/odata/odata/v4.0/os/part1-protocol/odata-v4.0-os-part1-protocol.html；
- OData 4.0 版，第二部分：URL 公约，2014 年 2 月 24 日，http://docs.oasis-open.org/odata/odata/v4.0/os/part2-url-conventions/odata-v4.0-os-part2-url-conventions.html；
- OData 4.0 版，第三部分：通用 schema 定义语言（CSDL），2014 年 2 月 24 日，http://docs.oasis-open.org/odata/odata/v4.0/os/part3-csdl/odata-v4.0-os-part3-csdl.html；
- OData 4.0 版，核心词汇表，2014 年 2 月 24 日，http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Core.V1.xml；
- OData 4.0 版，JSON 格式，2014 年 2 月 24 日，http://docs.oasis-open.org/odata/odata-json-format/v4.0/os/odata-json-format-v4.0-os.html；
- OData 4.0 版，检测单位词汇表，2014 年 2 月 24 日，http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Measures.V1.xml；
- 简单服务发现协议/1.0，1999 年 10 月 28 日，http://tools.ietf.org/html/draft-cai-ssdp-v1-03；
- 检测单位统一编码，http://www.unitsofmeasure.org/ucum.html；

- W3C 跨域资源共享建议，2014 年 1 月 16 日，http://www.w3.org/TR/cors/；
- 储存系统 SNIA TLS 规范，2014 年 11 月 20 日，http://www.snia.org/tls/；
- DMTF DSP0270，Redfish 主机界面规范，http://www.dmtf.org/sites/default/files/standards/documents/DSP0270_1.0.pdf。

3. 术语和定义

在本文件中，部分术语的具体含义可能不同于正常的英语语义。这些术语将在本节中进行定义。

本文件中使用的“必须”(“要求”)、“不得”、“应该”(“建议”)、“不应该”(“不建议”)、“可以”、“不需要”(“不要求”)、“能够”和“不能够”等词语的含义，应该根据 ISO/IEC 指令第二部分附件 H 中的规定进行解释。其中括号内的术语是前面术语的替代，可以在由于语言学原因无法使用前面的术语时使用。需要注意 ISO/IEC 指令第二部分附件 H 中规定的其他替代用词。在出现上述其他替代用词时，应该按照正常的英语语义进行解释。

本文件中使用的“条”、“款”、“段落”和“附件”含义应该根据 ISO/IEC 指令第二部分第 5 条规定进行解释。

本文件中使用的“强制性”和“供参考”术语的含义应该根据 ISO/IEC 指令第二部分第 3 条规定进行解释。在本文件中，带有“(供参考)”标识的条、款或者附件中不含有任何强制性内容。本文件中的备注和实例均为供参考内容。

另外，在下文中将对本文件中使用的其他术语进行定义。

术语	定义
基板管理控制器	计算机系统内的一种嵌入式设备或服务，通常是独立的微处理器或者带有相关固件的片上系统，主要用于执行系统监控和管理相关任务，通常在带外执行。
集合	参阅“资源集合”一节相关内容。
CRUD	任何界面使用的基本固有操作：创建、读取、更新和删除。
事件	与单独的报警相对应的记录。
受管理系统	在本规范文件中，受管理系统是指通过 Redfish 基界面提供信息或状态、或者可以通过上述界面进行控制的系统。
成员	成员是指包含在资源集合中的一个单一的资源实例。
消息	完成的请求或响应，格式为 HTTP/HTTPS。协议基于 REST，每次请求都会产生响应的请求/响应协议。
操作	HTTP 请求方法，能够映射到基本的 CRUD 操作上，其中包括 POST、GET、PUT/PATCH、HEAD、以及 DELETE。

OData	开发数据协议，具体定义参阅“OData 协议”相关内容。
OData 服务文件	提供关于服务根目录相关信息的资源的名称。服务文件提供了枚举服务所使用的资源的标准格式，从而确保 OData 客户端能够浏览 Redfish 服务的资源。
Redfish 报警接收器	接收 Redfish 服务发出的报警信号的功能的名称。此功能通常是在远程系统上运行的软件，与受管理系统互相分离。
Redfish 客户端	与 Redfish 服务通信的功能的名称，能够访问服务的一项或多项资源或功能。
Redfish 协议	用于发现、连接 Redfish 服务，以及与 Redfish 服务通信的协议组。
Redfish schema	Redfish 资源的 schema 定义。根据 OData schema 表达进行定义，可以直接转译为 JSON schema 表达。
Redfish 服务	也可以简称为“服务”。是指为一套或多套受管理系统实施本规范中规定的协议、资源和功能，并提供相关界面的功能组合。
Redfish 服务接入点	也可以简称为“服务接入点”。是指通过其访问 Redfish 服务的具体实例的界面。Redfish 服务中可以具有多个服务接入点。
请求	由客户端向服务器发送的消息。其中包括请求行（其中包括操作）、请求头、一个空行、以及可选择的消息体。
资源	资源可以通过 URI 寻址，能够接收和处理消息。资源可以是一个单一的实体，也可以是作为其他多个实体的容器的集合。
资源集合	资源集合是指用作其他资源的容器的资源。资源集合中的成员通常具有类似的特性。容器能够处理发送到容器处的消息。容器中的成员只能够处理发送给该成员的消息，不会影响容器中的其他成员。
资源树	资源树是一种 JSON 编码资源的树状结构，能够通过著名的起始 URI 进行访问。客户端可以通过跟踪资源树顶部的超链接，来发现 Redfish 服务中可用的资源。 Redfish 客户端实施**备注**：尽管资源为树状结构，但资源之间的互相引用关系可能会导致图形而非树状结构。因此，客户端在浏览资源的过程中必须带有防止无限循环的逻辑。
响应	由服务器发送给客户端的消息，是对请求消息的响应。其中包括状态行、响应头、一个空行、以及可选的消息体。
服务根目录	服务根目录是指可以直接通过服务接入点访问的具体资源。此资源可以用作其他资源、以及共同构成 Redfish 服务实例的相关元数据寻址和访问的起点。
订阅	注册一个用于接收事件的目的地的行为。

4. 符号和缩写

在本文件中使用的符号和缩写的定义如下所述：

术语	定义
BMC	基板管理控制器
CRUD	创建、替换、更新和删除
CSRF	跨站点请求伪造
HTTP	超文本传输协议
HTTPS	以安全为目标的超文本传输协议
IP	互联网协议
IPMI	智能平台管理界面
JSON	JavaScript 对象标记
KVM-IP	基于互联网协议的键盘、显示器和鼠标设备
NIC	网络接口卡
PCI	外设连接接口
PCIe	外设连接快速接口
TCP	传输控制协议
XSS	跨站脚本攻击

5. 概述

Redfish 可扩展平台管理 API（“Redfish”）是一种管理标准，使用超媒体 RESTful 界面内部的数据模型表达。由于其以 REST 为基础，因此与很多其他解决方案相比，Redfish 更加容易使用和实施。由于其采用模型导向，因此能够表达现代系统中的部件之间的关系，以及系统内部语义和部件之间的关系。Redfish 也可以很便捷地进行扩展。通过对 REST 使用超媒体方法，Redfish 可以表达由多家供货商提供的各种不同的系统。通过要求 JSON 表达，可以通过非规范化的方式创建多种资源，这不但能够提高可扩展性，且其负载可以通过大多数编程环境进行解释，同时对于查看数据的人而言也相对较为直观。模型采用可以互通的 Redfish schema，通过 OData schema 进行表达，同时转译为 JSON schema 表达，其中消息的负载根据 OData-JSON 公约使用 JSON 进行表达。Redfish schema 能够使用外部主机，通过机器可读格式来定义资源，因此可以让 Redfish 服务在不需要枚举元数据的情况下，在元数据和数据之间建立关联关系，从而能够在很多数据中心和云环境中发现更先进的客户端。

5.1 适用范围

本规范文件对一种具有可互通性、跨供货商、远程和带外能力，满足云环境和网页基 IT 专家对可扩展平台管理期待的界面所需的协议、数据模型、和行为方式及其他 schema 部件进行了定义。尽管大规模系统是首要关注所在，但本规范文件同样可以用于很多传统系统平台管理的实施。

本规范规定了所有 Redfish 实施都必须采用的强制性要素，以及可以由系统供货商或生产商自由选择的可选要素。本规范同样对在某一实施情况下可提供原始设备制造商（系统供货商）的特定扩展点进行了定义。

本规范为 Redfish 服务及相关材料设置了强制性要求，例如 Redfish schema 文件。总体而言，本规范并未对 Redfish 客户端的具体要求做出规定，但会说明 Redfish 客户端需要做什么，才能够更加成功、有效地访问和使用 Redfish 服务。

本规范并未对实施 Redfish 界面和功能所用的特殊硬件或固件设置要求。

5.2 目标

作为一种 schema 和数据表达，以及在定义用于访问 Redfish 服务并与之互动的各种协议的过程中，Redfish 有着多个目的和目标。Redfish 希望提供能够满足下列目标要求的规范：

- 可扩展性：能够支持独立式服务器应用于云服务环境中设立的机架式服务器；
- 灵活性：能够支持当前使用的各种不同系统；
- 可扩充性：能够在数据模型的框架内，支持各种新的、或者供货商独有的具体功能；
- 向后兼容性：在保留本规范早期版本的内容的同时，能够不断加入新的功能；
- 可互通性：提供有用、要求的基准，能够确保不同供货商的功能和实施情况具有一致性；
- 以系统为中心：能够有效支持在可扩展环境中使用的最常用的平台硬件管理功能，同时能够管理当前的服务器环境；
- 以标准为基础：能够适应在当前环境中被广泛接受和使用的协议和标准，更加具体地讲，能够适应当前网页基客户端开发过程中被广泛使用的编程环境；
- 简单：软件开发师可以直接使用，不需要高超的编程技能或丰富的系统知识；
- 轻质：能够降低在管理系统上实施和验证 Redfish 服务的复杂性和成本。

5.3 设计理念

使用了下列设计理念和技术，以确保能够达到前文中所述的目标，获得相关特性：

- 使用 JSON 负载和实体数据模型来提供 RESTful 界面；

- 将协议与数据模型分离，确保可以对其进行独立修改；
- 规定了协议和 schema 的版本变更原则；
- 利用互联网协议标准的强度来满足 schema 要求，例如本文件中引用的 JSON、HTTP、OData 和 RFCs 协议；
- 关注带外访问：在现有的 BMC 和固件产品基础上实施；
- 对 schema 进行组织，在提供标准化项目的同时还提供增值功能；
- 明确数据定义文本；
- 维持实施过程中的灵活性：不将界面与任何具体的潜在实施 schema 绑定，“让界面而不是实施过程标准化”；
- 关注应用最广泛的“公分母”功能。避免引入只有少部分用户使用的功能，从而增加复杂度；
- 避免在管理控制器中引入复杂度，以确保客户端能够更有效地操作。

5.4 限制条件

Redfish 不保证客户端软件永远不需要更新。可能需要进行更新的实例包括：新型系统或其部件配置、数据模型更新等等。在应用过程中，系统优化总是需要架构的监督。尽管如此，Redfish 确实在试图努力降低强制客户端升级的情况，具体做法包括使用 Schemas, 严格的版本号、向前兼容原则、以及将协议与数据模型分离等。

可互通性并不意味着完全相同。Redfish 客户端可能需要实施由不同供货商提供的其他可选功能。特定供货商提供的具体产品的实施和配置情况可能也会发生变化。

例如：Redfish 不能让客户端读取资源树并将其写入到其他的 Redfish 服务中。对于超媒体 API 而言，这是不可能实现的。只有根对象才有一个已知的 URI。资源的拓扑结构反映的是其所代表的系统和装置的拓扑结构。因此，不同类型的服务器和装置可能会导致不同形状的资源树，即使其使用了由相同生产商提供的相同系统。

另外，不是所有的 Redfish 资源都是简单的读/写资源。在实施过程中，可能需要采用本规范后文中将介绍的其他互动模式。例如：不能简单地从一个服务中读取用户凭证或证书，再将其转移到另一个服务中。另一个示例是对读出的同一资源用设置数据代替写入。

最后，在实施过程中，资源与其他要素之间的超链接的值会发生变化。客户端不能假设在 Redfish 服务的不同实施过程中，上述超链接可以回用。

5.5 其他设计背景和理由

5.5.1 REST 基础

本规范中规定了一种 RESTful 界面。很多服务应用都将在 RESTful 界面中提供。

规定 RESTful 界面具有多种原因和理由，具体如下所述：

- 实施轻量化，提高经济性（传输的数据量小于 SOAP、协议层数比 WS-Man 更少）；
- 是一种在行业中被普遍使用的访问方法；
- 更加容易学习和记录；
- REST 可以使用多种工具套装和开发环境；
- 能够支持数据模型语义，能够便捷地映射到常用的 CRUD 操作上；
- 与我们的“简洁”的设计理念相符合；
- 能够同时适用于软件应用以及嵌入式环境，因此能够促进管理生态系统中部件代码的集合和共享；
- 具有 schema 兼容性，能够很好地适应任何一种建模语言；
- 在使用此界面之后，Redfish 能够进一步提高和促进行业中现有的安全机制和发现机制。

5.5.2 遵守 OData 公约

伴随 RESTful API 的流行，RESTful 界面的数量几乎和应用程序的数量相当。尽管遵守 REST 模式能够促进良好的实践规范，但由于设计上的差异，很多 RESTful API 之间完全不具备互通性。

OData 中规定了一组常用的 RESTful 公约，在采用之后，能够在不同的 API 之间提供互通性。采用 OData 公约来描述 Redfish schema、URL 公约和 JSON 负载中常用属性的命名和结构，不但能够概括 RESTful API 的最佳实践方式，还能够进一步在逐渐增长的通用型客户端库、应用、以及工具系统中实施 Redfish 服务。

5.5.3 模型导向

Redfish 模型是针对系统管理而建立的。所有资源都将在 OData schema 表达中进行定义，并将其转译为 JSON schema 表达。OData 是一种行业标准，概括了 RESTful 服务的最佳实践方式，并为不同类型服务之间提供可互通性。JSON 在多个领域内被广泛利用，具有大量的工具和编程语言，在使用这些方法之后能够加快开发的速度。

5.5.4 协议与数据模型分离

协议的操作与数据模型相独立。协议的版本也与数据模型相独立。使用这种设计理念的理由是：协议的版本变化频率不高，但数据模型版本可以根据需要随时变化。这就意味着创新主要将发生在数据模型上，而不是协议上。这种设计方式允许根据需要对数据模型进行扩展和变更，同时不需要调整协议或 API 版本。另一方面，在将协议与数据模型分离之后，还可以在不对数据模型进行重大修改的情况下，对协议进行调整和变更。

5.5.5 超媒体 API 服务端点

和其他超媒体 API 一样，Redfish 也具有单一的服务端点 URI，其他所有资源都可以通过从根目录引用的不透明 URI 进行访问。通过访问根服务、或任一从根服务引用的服务或资源建立超链接，由此所发现的资源都将遵守根服务支持的协议版本。

5.6 服务要素

5.6.1 同步和非同步操作支持

尽管 schema 中大多数操作在本质上都具有同步的性质，但部分操作需要较长的时间执行，甚至超过客户端通常希望等待的时间。由于这个原因，服务可能会自行决断，对部分操作采取非同步模式。非同步操作的请求部分与同步操作的请求部分没有任何区别。
使用 HTTP 响应代码能够让客户端选择使用同步模式还是非同步模式来执行相关操作。更多详细信息，请参阅本规范“任务”条款。

5.6.2 事件机制

在部分情况下，服务在请求/响应机制之外为客户端提供信息可能会非常有用。这些信息通常称为事件，主要供服务器用于以非同步的方式为客户端提供一些重要信息，例如重要的状态变化或错误条件，事件通常具有时间关键性质。
在本规范中，当前只对一种事件做出了规定，即推送式事件。在推送式事件中，服务器检测是否需要发送事件，使用 HTTP 的发送请求，将事件消息推送给客户端。客户端可以通过在事件服务中创建订阅条目来接收事件消息；或者，由管理员创建订阅作为 Redfish 服务配置部分。所有订阅都具有固定的配置设置。
在本规范“事件”条款中，将进一步对事件机制进行详细的介绍。

5.6.3 动作

操作可以分为两类，即内在操作和外在操作。内在操作通常是指 CRUD，可以映射到 HTTP 方法上。尽管如此，协议还可以支持外在操作：这些操作不能简单地映射到 CRUD 上。外在操作指成套操作时更好执行的操作（出于可扩展性、界面的简洁性、服务器端语义保留、或者类似原因）或与 CRUD 操作之间没有天然映射关系的操作。其中一个示例是系统重置。可以将多个操作整合到一个单独的动作中。在建模过程中，系统重置可以建模为状态更新，但从语义的角度看，客户端实际上需要执行状态变更、而不是简单地变更状态的数值。
在 Redfish 中，这些外在操作通常称为**动作**，将在本规范的不同部分进行详细介绍。
Redfish schema 中规定了一些与通用 Redfish 资源相关的特定标准动作。对于这些标准动作而言，Redfish schema 中含有适用于动作行为的规范语言。尽管如此，也允许原始设备制造商对 Redfish schema 进行扩展，包括明确现有资源动作。

5.6.4 服务接入点发现

尽管服务本身具有知名 URI，但服务主机却不能被发现。和 UPnP 一样，Redfish 使用 SSDP 进行发现。很多设备都支持 SSDP，例如打印机。SSDP 具有简单、轻质、兼容 IPv6、以及适合在嵌入环境中实施等优点。Redfish 还在研究其他服务接入点发现（例如：DHCP 基机制）方法。
更多详细信息，请参阅本规范“发现”条款。

5.6.5 远程访问支持

此 schema 能够支持多种远程访问和重新定向服务。对于带外环境而言非常重要的服务包括：支持串行控制台访问，键盘-显示器-鼠标重新定向（KVM-IP），命令窗口（即：命令行界面），以及远程虚拟媒体的机制。对串行控制台、命令窗口、KVM-IP 和虚拟媒体的支持都将包括在本规范中，并在 Redfish schema 中得到表达。本标准中没有规定访问这些装置和服务的具体协议和访问机制。Redfish schema 为这些服务提供了表达和配置，在这些服务和其操作状态之间建立了关联关系。尽管如此，这些协议本身的规范并不在本标准的范围之内。

5.7 安全性

远程界面所面临的安全性方面的挑战是一个程序设计问题，必须确保与 Redfish 互动的界面、以及所交换数据的安全性。这就意味着必须围绕界面设计合理的安全控制机制，以确保用于

数据交换的通道的安全性。在这方面，需要采取一些具体的行动，其中包括在通信通道内定义和使用最低水平的加密技术等。

6. 协议细节

Redfish 可扩展平台管理 API 是以 REST 为基础，遵守 OData 互通性公约（定义参阅“OData 协议”相关内容）、JSON 负载（定义参阅“OData-JSON”相关内容）、以及 schema 的机器可读格式表达（定义参阅“OData 协议”相关内容）。OData schema 表达中包括可以直接转译为 JSON schema 表达的注释，以便进行验证、使用支持 JSON schema 的工具。通过遵守这些同意的标准和公约，可以增加可互通性，促进现有工具链的使用。

对于使用最小元数据的客户端而言，Redfish 还遵守 OData 的最小合规性水平。

在本规范文件中，我们将指出 Redfish 具有能够映射到数据模型的协议。更加准确地讲，使用 HTTP 协议作为传输消息的应用协议，使用 TCP/IP 作为传输协议。RESTful 界面可以映射到消息协议上。为了确保简洁性，我们将会把 RESTful 映射到 HTTP、TCP/IP、以及 Redfish 协议中的其他协议、传输和消息层面上。

Redfish 协议是围绕网页服务基界面模型设计的，主要设计目的是确保用户界面（UI）和自动使用的网络和互动效率。界面专门使用 REST 模式的语义进行设计。

Redfish 协议使用 HTTP 方法来执行通用的 CRUD 操作和数据头信息检索。

使用动作来执行超出 CRUD 范围的其他操作，但在使用过程中将受到一定的限制。

使用媒体类型来确定在消息本体中发送的数据的类型。

使用 HTTP 状态代码来显示服务器处理请求的意图。使用扩展错误处理来返回比 HTTP 错误代码提供信息更多的信息。

能够发送安全信息的功能是非常重要的。在本规范文件中的“安全性”一条中，将对具体的 TLS 要求进行介绍。

部分操作所需要的时间可能比同步返回语义所需时间更长。因此，在 schema 中还包括了确定性的非同步语义。

6.1 使用 HTTP

HTTP 与 RESTful 界面之间具有完美的匹配关系。在本条中，将介绍如何在 Redfish 界面中使用 HTTP，以及需要在 HTTP 的顶层加入哪些限制条件，以确保 Redfish 实施的可互通性。

- 应该通过使用超文本传输协议 1.1 版实施的网页服务端点来接入 Redfish 界面（RFC7230、RFC7231、RFC7232）。

6.1.1 统一资源标识符

使用统一资源标识符（URI）来识别资源，其中包括基本服务和所有 Redfish 资源。

- 每个独特的资源实例都将通过一个 URI 来进行识别；
- 客户端应该将 URI 处理为不透明状态，因此除了 RFC3986 文件第 5 条-“指代消解”中规定的标准指代消解规则之外，客户端不得视图理解或者消解 URI。

在开始操作之前，客户端必须知道一个资源的 URI。

- 执行 GET 操作将会产生一个资源表达，其中含有相关资源的属性和超链接。

基本资源的 URI 是已知的，是根据协议版本确定的。通过之前操作响应过程中返回的相关资源的超链接，可以发现其他资源的 URI。这种通过服务返回的 URI 来浏览资源的 API 即为超媒体 API。

Redfish 将采用 RFC3986 中规定的 URI 的三个部分。

第一部分包括 URI 的机制和授权部分。第二部分包括根服务和版本。第三部分是一个独特的资源识别符。

例如，在下列 URL 中：

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

- 第一部分是机制和授权部分（https://mgmt.vendor.com）；
- 第二部分是根服务和版本（/redfish/v1/）；
- 第三部分是独特的资源路径（Systems/1）。

URI 中的机制和授权部分不得视为资源的独特识别符部分。这是因为重新定向功能和本地操作都可能会导致连接部分发生变化。URI 中的剩余部分（服务和资源路径）是在特定 Redfish 服务中能够独特识别资源的识别符部分。

- 在实施过程中，URI 中的独特识别符部分必须保持独特性；
- 在同一个实施过程中，可以使用负载（消息体或者 HTTP 数据头）中的相对 URI 来识别资源；
- 在不同的实施过程中，可以使用负载（消息体或者 HTTP 数据头）中的绝对 URI 来识别资源。绝对 URI 的定义参阅 RFC3986 相关内容。

例如：在执行 POST 操作后，可能会在响应的位置数据头中返回下列 URI（表示通过 POST 操作创建的新的资源）：

```
Example: /redfish/v1/Systems/2
```

假设客户端是通过名为“mgmt.vendor.com”的设备进行连接的，则访问上述新资源的完整 URI 将为：

```
https://mgmt.vendor.com/redfish/v1/Systems/2.
```

在 RFC3986 中规定的 URI 中，还可以包括一个查询（?query）和碎片（#frag）部分。其中

查询部分将在“查询参数”一条中进行介绍。在用作提交操作的 URI 时，碎片（frag）部分将被服务器忽略。

如果响应中的某个属性是对资源中的另一个属性的引用，则 RFC6901 中规定的“URI 碎片识别符表达”格式将被使用。如果在 schema 内该属性被定义为参考属性，则上述碎片应该视为有效的资源识别符。例如，下列碎片表示 resource/redfish/v1/Chasis/MultiBladeEncl/Thermal 中 Fans 数组中编号为 0 的属性：

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

6.1.2 HTTP 方法

RESTful 界面的另一个重要功能是只支持有限数量的操作类型。下表中将列出这些基本操作与 HTTP 方法之间的基本映射关系。如果表格中“要求”一栏中的数值为“是”，则说明 Redfish 界面能够支持此 HTTP 方法。

HTTP 方法	界面语义	要求
POST	目标创建、目标动作、事件	是
GET	目标检索	是
PUT	目标替换	否
PATCH	目标更新	是
DELETE	目标删除	是
HEAD	目标数据头检索	否
OPTIONS	数据头检索，CORs 预检	否

对于其他不允许的 HTTP 方法而言，将会返回 405 响应。

6.1.3 HTTP 重新定向

HTTP 重新定向是指服务将一个请求重新定向到另一个 URL 上。除了其他方面之外，此功能让 Redfish 资源能够重新命名数据模型中的区域。

- 所有 Redfish 客户端都必须能够正确处理 HTTP 重新定向。

备注：HTTP 重新定向导致的安全性方面的问题，请参阅“安全性”一条相关内容。

6.1.4 媒体类型

部分资源可能具有超过一种的表达类型。资源的表达类型通过媒体类型显示。

在 HTTP 消息中，媒体类型将在内容-类型数据头中做出规定。通过设置 HTTP 接受数据头，在其中加入可接受媒体类型的列表，客户端可以向服务器说明其希望接受的响应所采用的特定的媒体类型。

- 所有资源都应该使用 JSON 媒体类型“application/json”；
- Redfish 服务必须根据 RFC4627 中规定的 JSON 规则表达每种可用资源。接收设备不得拒收消息，因为消息是使用 JSON 编码的。接受设备必须提供至少一个根据 JSON 表达的响应表达。在具体实施中，可以使用非 JSON 媒体类型的其他表达。

客户端可以在请求的“接受-编码数据头”中做出规定，要求进行压缩处理。

- 在客户端提出请求时，服务器必须能够支持 gzip 压缩操作。

6.1.5 ETags

为了减少 RESTful 对资源进行不必要的访问，Redfish 服务应该支持为每个资源提供一个独立的 ETag。

- 在实施过程中，每个资源都必须支持返回 ETag 属性；
- 在实施过程中，代表单个资源的每次响应都必须能够支持返回 ETag 数据头；
- 在实施过程中，必须能够支持针对管理者账号（ManagerAccount）资源的 GET 请求返回 ETag 数据头。

ETag 将作为资源负载的一部分生成和提供，这是因为服务器处于最佳位置，能够判断目标的新版本是否具有充分、能够被视为实质性的差异。共有两种类型的 ETag：弱和强。

- 弱模式：在 ETag 编制过程中，只包括了目标中的“重要”部分。例如，诸如最新修改时间等元数据将不会被包括到 ETag 生成中。能够确定 ETag 变动的“重要”属性包括可写入设置，以及诸如 UUID、FRU 数据、序列号等可变更属性；
- 强模式：在 ETag 编制过程中，包括了目标中的所有部分。

在本规范中，并不会对创建 ETag 的具体算法做出强制性规定，但 ETag 应该具有较高的无冲突性。ETag 可以为散列、生成的 ID、时间戳、或者能够在目标发生变化时会随之变化的其他数值。

如果客户端对资源执行 PUT 或者 PATCH 操作，则在 ETag 中应该纳入一个通过之前 GET 指令获得的 HTTP If-Match/If-None-Match 数据头。如果服务支持返回资源的 ETag 数据头，且如果对相同资源的 PUT/PATCH 请求中没有 If-Match/If-None-Match 数据头，则响应将会返回状态代码 428，更多信息参阅 RFC6585 相关内容。

除了返回每种资源的 ETag 属性之外：

- 在客户端执行 PUT/ POST/ PATCH 操作时，Redfish 服务应该返回 ETag 数据头；
- 在针对某种资源执行 GET 操作时，Redfish 服务应该返回 ETag 数据头。

ETag 数据头的格式为：

```
ETag: W/"<string>"
```

6.2 协议版本

协议版本与其支持的资源版本或 Redfish schema 版本相分离。

Redfish 协议的每个版本都具有很强的类型性。这是通过将 Redfish 服务的 URI 和在该 URI 处获得的资源结合后得到的，这成为服务根目录。

此版本 Redfish 协议的根目录 URI 应该为“/redfish/v1/”。

尽管协议的主要版本将在 URI 中进行表达，协议的主要版本、次要版本和勘误版本将在服务根目录资源的版本属性中进行表达，具体将通过 Redfish 对该资源进行定义实现。协议版本是一个具有下列形式的字符串：

主要版本.次要版本.勘误版本

其中：

- 主要版本=整数：以向后兼容的方式，对分级中的部分内容进行了变更；
- 次要版本=整数：可以加入新功能，但不能删除任何内容。应该保留与之前次要版本之间的兼容性；
- 勘误版本=整数：之前版本中的部分内容损坏，需要进行勘误和修改。

任何通过访问服务根目录发现的超链接发现的资源，或者通过引用服务根目录发现的任何资源或服务，都必须与服务根目录所支持的协议具有相同的版本号。

对资源“/redfish”执行 GET 操作，将会返回下列消息体：

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

6.3 Redfish 绝对 URI 和相对 URI 规则

Redfish 是一个超媒体 API，具有一组绝对 URI。所有其他资源都可以通过引用服务根目录下的不透明的 URI 来进行访问。Redfish 服务必须能够支持如下所述的各项 Redfish 绝对 URI：

URI	描述
/redfish	此 URI 用于返回版本信息。
/redfish/v1/	此 URI 用于访问 Redfish 服务根目录。
/redfish/v1/odata	此 URI 用于访问 Redfish OData 服务文件。

/redfish/v1/$metadata	此 URI 用于访问 Redfish 元数据文件。

除此之外，下列没有尾部斜杠的 URI 要么可以重新定向到下表中所示的相关 Redfish 绝对 URI 上，要么会被服务视为与相关 Redfish 绝对 URI 相同的 URI。

URI	相关 Redfish 绝对 URI
/redfish/v1	/redfish/v1/

服务使用的所有相对 URI 都必须以一个双斜杠（“//”）开始，其中包括一个授权部分（例如：“//mgmt.vendor.com/redfish/v1/Systems”）；或者以一个单斜杠（“/”）开始，其中包括一个绝对路径（例如：“/redfish/v1/Systems”）。

6.4 请求

本条中将对可以向 Redfish 服务发送的请求进行介绍。

6.4.1 请求数据头

在请求消息中可以使用 HTTP 数据头。在下表中，将对这些数据头、以及对 Redfish 的要求做出规定。需要注意的的一点是，这些要求是针对 Redfish 服务的，而不是针对发送 HTTP 请求的客户端的。

- 如果下表中“要求”一栏中的数值为“是”，则 Redfish 服务必须能够理解和处理下表中列出的，使用 HTTP 1.1 规范确定的数据头；
- 如果下表中“要求”一栏中的数值为“有条件”，则在“描述”一栏中的条件得到满足的情况下，Redfish 服务必须能够理解和处理下表中列出的，使用 HTTP 1.1 规范确定的数据头；
- 如果下表中“要求”一栏中的数值为“否”，则 Redfish 服务应该能够理解和处理下表中列出的，使用 HTTP 1.1 规范确定的数据头。

数据头	要求	支持数值	描述
Accept	是	RFC 7231	向服务器说明客户端准备接收什么媒体类型。服务必须支持的带有 Accept 数据头的资源请求包括：application/json 或者 application/json;charset=utf-8。服务必须支持的带有 Accept 数据头的元数据请求包括：application/xml 或者 application/xml;charset=utf-8。服务必须支持的带有 Accept 数据头的所有资源请求包括：application/*、或者 application/*;charset=utf-8,*/*；或者*/*; charset=utf-8。
Accept-	否	RFC 7231	说明客户端是否可以处理 gzip 编码。如果请求中带有

Encoding			Accept-Encoding 数据头，但服务器不能发送能够被 Accept-Encoding 数据头所接受的响应，则服务器应该使用 406 状态代码进行响应。如果请求中没有 Accept-Encoding 数据头，则服务器不能返回 gzip 编码的响应。
Accept-Language	否	RFC 7231	此数据头用于说明响应需要采用的语言。如果此数据头中没有做出明确规定，则将使用设备的默认区域设置。
Content-Type	有条件	RFC 7231	用于描述消息体中采用的表达类型。在带有请求消息体的请求中，应该使用 Content-Type 数据头。服务必须能够接受数值为 application/json 或者 application/json;charset=utf-8 的 Content-Type 数值。
Content-Length	否	RFC 7231	用于描述消息体的长度。另外一种说明消息体长度的可选方法为使用 Transfer-Encoding：分块编码，从而不需要使用 Content-Length 数据头。如果服务器不支持 Transfer-Encoding，而需要使用 Content-Length 进行替代，则服务其将会使用状态代码 411 进行响应。
OData-Maxversion	否	4.0	用于说明客户端能够理解的最高的 OData 版本号。
OData-Version	是	4.0	规定了不支持版本的 OData 的请求将被服务器拒绝。如果服务器遇到其不支持的版本号，则服务器将通过状态代码 [412]（#status-412）拒绝。如果客户端没有规定 OData-Version 数据头，则上述客户端将不在本规范适用范围之内。
Authorization	有条件	RFC 7235 第 4.2 节	“基本验证”时需要。
User-Agent	是	RFC 7231	在跟踪产品令牌和其版本时需要。可以在其中列出多个产品令牌。
Host	是	RFC 7230	在允许支持使用单一 IP 地址的多来源主机时需要。
Origin	是	W3C CORS 第 5.7 节	用于允许网页基应用使用 Redfish 服务，同时防止 CSRF 攻击。
Via	否	RFC 7230	用于说明网络的层级结构，识别消息循环。每次通过时都会插入其 VIA 值。
Max-Forwards	否	RFC 7231	限制网关和跳过代理。防止消息在网络中无限停留。
If-Match	有条件	RFC 7232	对于服务器需要返回 ETags 的资源的 PUT 和 PATCH 操作而言，必须支持 If-Match 数据头，以确保客户的能够从已

			知的状态开始更新资源。
If-None-Match	否	RFC 7232	如果请求中带有此数据头，则只有在资源当前的 ETag 与数据头中规定的 ETag 不匹配时，服务器才会返回请求的资源。如果资源当前的 ETag 与数据头中规定的 ETag 相一致，则服务器将会返回 304 代码。

- 如果下表中“要求”一栏中的数值为“是”，则 Redfish 服务必须能够理解和处理下表中列出的数据头。

数据头	要求	支持数值	描述
X-Auth-Token	是	不透明的编码八字节字符串	用于用户会话的认证。令牌值必须为不能辨别的随机值。

6.4.2 读取请求（GET）

读取（GET）方法主要用于检索资源的表达。在满足本规范文件“媒体类型”一条中相关要求的情况下，服务将会使用 Accept 数据头中规定的一种媒体类型返回资源表达。如果请求中没有 Accept 数据头，则服务将会通过 application/json 返回资源表达：

- HTTP GET 方法可以用于检索资源，同时不会产生任何副作用；
- 服务器将会忽略 GET 请求消息体中的内容；
- 在资源没有发生外部变化的情况下，GET 操作必须具有幂等性。

6.4.2.1 服务根目录请求

Redfish 第 1 版的服务根目录 URL 为“/redfish/v1/”。
返回 ServiceRoot 资源的服务器根目录的 URL 将在本规范文件中做出规定。
在检索服务器根目录和“/redfish”文件时，服务器不需要进行认证。

6.4.2.2 元数据文件请求

Redfish 服务必须在“/redfish/v1/$metadata”资源中设置一份元数据文件，用于描述服务。此元数据文件应该描述根目录下可用的资源，同时还应该引用服务器可以访问的、用于描述整套资源类型的其他元数据文件。
在检索元数据文件时，服务器不需要进行认证。

6.4.2.3 OData 服务文件请求

Redfish 服务必须在“/redfish/v1/odata”资源中设置一份 OData 服务文件。此服务文件将提供对服务器可访问资源进行枚举的标准格式，确保普通超媒体 OData 客户端能够浏览服务器中的资源。

在检索上述服务文件时，服务器不需要进行认证。

6.4.2.4 资源检索请求

客户端可以通过向每个具体资源的 URI 发送 GET 指令来进行资源检索。资源的 URI 可以通过之前请求返回的资源识别符属性获得（例如：在之前返回资源中的链接属性内）。服务器可以（但不强制要求）支持检索资源具体属性的公约，但需要补充一个区段，在其中列出属性的名称和资源的 URI。

6.4.2.4.1 查询参数

如果资源位于资源集合内，则客户端可以使用下列分页查询功能，明确说明需要返回的资源集合中的子集成员。这些分页查询功能只适用于资源集合内部的“成员”的数组属性。

属性	描述	示例
$skip	整数：表示在开始检索第一个资源之前，需要跳过的资源集合中的成员数量。	http://resourcecollection?$skip=5
$top	整数：表示在响应中需要包括的成员的数量。此参数的最小值应该为 1，默认行为是否返回资源集合中的所有成员。	http://resourcecollection?$top=30

- 服务器应该支持$skip 和$top 查询参数；
- 在实施之后，应该返回 501 代码。在未实施的情况下，每个查询参数的状态代码都将以“$”开始，同时应该返回扩展错误代码，表示此资源不支持请求的查询参数；
- 在实施过程中，未知或者不支持、且不是以“$”开头的查询参数将被忽略。

6.4.2.4.2 检索资源集合

通过向资源的 URI 发送 HTTP GET 请求可以执行资源集合检索操作。响应中将会包括资源集合的属性，包括其成员数组。使用客户端分页查询功能，可以返回资源集合的一个成员子集。

在实施过程中，不要求在一定时间内，通过持续不断使用分页查询参数发送一系列请求之后，需要返回资源集合中的全部成员。因为在使用分页参数通过多次 GET 指令进行检索时，可能会有部分成员被遗漏、也可能有部分成员重复。

- 客户端不能假设资源集合内每个成员的具体 URI；
- 在检索到的资源集合中，应该随时包括一个计数（count）属性，在其中说明其“成员”数组中的总条目数量；
- 不论是否进行分页，在查看部分结果时，在计数属性中，必须说明返回的成员数组中所引用的资源的总数量。

6.4.3 HEAD

HEAD 方法与 GET 方法的区别在于：其绝对不允许返回消息体内的信息。尽管如此，在执行 HEAD 操作之后，将会返回与使用 GET 指令后相同的 HTTP 数据头中的元信息和状态代码，包括授权检查信息：

- 服务器可以支持 HEAD 方法，以便以 HTTP 响应数据头的方式返回元信息；
- 服务器可以支持 HEAD 方法，以便验证超链接的有效性；
- 服务器可以支持 HEAD 方法，以便验证资源的可访问性；
- 服务器不得支持 HEAD 方法的其他任何用途；
- 在资源没有发生外部变化的情况下，HEAD 方法必须具有幂等性。

6.4.4 数据修改请求

客户端可以执行适宜的创建（Create）、更新（Update）、替换（Replace）或者删除（Delete）操作，或者通过调用对资源的动作，来创建、修改或者删除资源。如果相关的资源存在、但不支持请求的操作，则服务器将会返回 405 状态代码。如果客户端（4xx）或者服务器（5xx）返回了状态代码，则说明相关操作将不能修改资源。

6.4.4.1 更新（PATCH）

PATCH 方法是用于对现有资源进行更新的优选方法。通过请求的消息体，说明需要对资源进行的具体变更。在请求消息体中没有规定的属性，将不能通过 PATCH 方法直接修改。上述请求的响应要么是空响应，要么是更新完成之后的资源的表达。在实施过程中，可以根据其具体的政策规定，拒绝执行部分特定领域内的更新操作。在这种情况下，此更新方法将不会适用于这些领域内的资源。

- 服务器可以支持 PATCH 方法来更新资源。如果相关资源不能被更新，则服务器应该返回 405 状态代码；
- 在服务器端的资源发生变化之后，服务器可以在响应消息体中返回相关资源的表达；
- 如果请求内的某个属性不能被更新，例如其中的只读属性，则服务器将会随资源表达一同返回一个 200 状态代码，其中含有一条注释，说明无法更新的具体属性。在成功更新之后，资源中的其他属性将被更新；
- 如果客户端对资源集合提出的 PATCH 请求，则服务器应该返回 405 状态代码；
- 在资源没有发生外部变化的情况下，GET 操作必须具有幂等性，即使初始的 ETag 值已经不再相同；
- 服务器可以接受对空的 JSON 目标提出的 PATCH 指令。本段中所述的空的 JSON

目标是指没有请求对资源进行任何修改。

服务器中可以设置 JSON 数组属性空条目，以显示客户端在提出 PATCH 请求之后允许使用的条目数量。在 PATCH 请求中，JSON 数组中未变更的成员可以使用空的 JSON 目标进行规定，同时从 JSON 数组中清除掉具有空条目的成员。

在更新过程中，将会忽略 OData 标记（资源识别符、类型、etag 和链接属性）。

6.4.4.2 替换（PUT）

PUT 方法主要用于完全替换资源。请求消息体中忽略的、或者通常由服务器提供的，在定义资源过程中所必须的资源属性将会由服务器添加到最终替换后的资源中。

- 服务器可以支持使用 PUT 方法将资源全部替换。如果服务器不支持此方法，则将会返回 405 状态代码；
- 在服务器端的资源发生变化之后，服务器可以在响应消息体中返回相关资源的表达；
- 对于其中没有包括资源定义（schema）所必须的属性的请求而言，服务器可以拒绝；
- 如果客户端对资源集合提出的 PUT 请求，则服务器应该返回 405 状态代码；
- 在资源没有发生外部变化的情况下，GET 操作必须具有幂等性，可能出现的例外情况是：在 PUT 操作结束之后，资源的 ETAG 数值可能会发生变化。

6.4.4.3 创建（POST）

POST 方法用于创建新资源。POST 请求将会提交给新资源所属的资源集合。

向资源集合提交 POST 请求，相当于向资源集合中的成员属性提交了相同的请求。支持在资源集合内新增新成员的服务器，必须同时支持上述两种形式的请求。

- 服务器必须支持使用 POST 指令创建资源。如果资源中不能任何被创建的东西，则必须返回 405 状态代码；
- 服务器必须支持对资源集合实例中引用的 URL 执行 POST 操作；
- 服务器应该支持对动作中引用的 URL 执行 POST 操作（参阅“动作（POST）一节相关内容”）；
- POST 操作不得具有幂等性。

创建请求的消息体中应该包括有需要创建的资源的表达。服务器可以忽略请求中的任何服务器控制属性（例如：id），强制覆盖这些属性。服务器必须将位置数据头写入到新创建资源的 URI 中。在资源成功创建之后，应该使用 201 代码（已创建）进行响应，同时在其响应消息体中应该根据新创建资源的 schema 要求，纳入新创建资源的表达。

6.4.4.4 删除（DELETE）

DELETE 方法用于删除资源：

- 对于可以被删除的资源而言，服务器必须支持 DELETE 方法。如果相关资源不能

被删除，则必须返回 405 状态代码；

- 在响应消息体中，服务器可以返回被删除的资源的表达；
- 如果客户端对资源集合提出的 DELETE 请求，则服务器应该返回 405 错误代码。

如果相关资源被成功删除，则服务器可以返回 404 状态代码或者成功代码。

6.4.4.5 动作（POST）

POST 方法用于对资源发起操作（例如：动作）：

- 服务器必须支持发送动作请求的 POST 方法；
- POST 操作可以不具有幂等性。

通过向动作的 URI 发送 HTTP POST 指令，可以对资源执行常规动作。如果资源内的动作属性中没有规定目标属性，则动作的 URI 应该为下列形式：

ResrouceUri/Actions/QualifiedActionName

其中：

- ResrouceUri 是支持相关调用动作的资源的 URI；
- Actions 是含有相关资源动作的属性名称，具体将在本规范文件中做出规定；
- QualifiedActionName 是相关动作的命名空间、或者其他合格的名称。

绑定函数中的第一个参数是对其调用相关动作的资源，其余参数应该表达为请求消息体内的名称/数值组合。

客户端可以直接对资源进行查询操作，确定相关资源可用的动作、以及这些动作的有效参数数值。部分参数信息可能需要客户端检查 Redfish schema 与资源之间的相关性。

例如：如果在 Redfish schema 文件 http://redfish.dmtf.org/schemas/v1/ComputerSystem_v1.xml 中，在命名空间 ComputerSystem 中定义了重置动作，则应该将其绑定到 ComputerSystem.v1_0_0.Actions 类型上。在此示例中：

```
<Schema Name="ComputerSystem">
  ...

  <Action Name="Reset" IsBound="true">
    <Parameter Name="Resource" Type="ComputerSystem.v1_0_0.Actions"/>
    <Parameter Name="ResetType" Type="Resource.ResetType"/>
  </Action>
  ...
</Schema>
```

在计算机系统资源中所包含的动作属性如下所述：

```
{}{
    "Actions": {
        "#ComputerSystem.Reset": {
            "target":"/redfish/v1/Systems/1/Actions/ComputerSystem.Reset",
            "ResetType@Redfish.AllowableValues": [
                "On",
                "ForceOff",
                "GracefulRestart",
                "GracefulShutdown",
                "ForceRestart",
                "Nmi",
                "ForceOn",
                "PushPowerButton"
            ]
        }
    },
    ...
}
```

之后，下列内容代表可以提出的动作请求：

```
POST /redfish/v1/Systems/1/Actions/ComputerSystem.Reset HTTP/1.1
Content-Type: application/json;charset=utf-8
Content-Length: <computed length>
OData-Version: 4.0

{}{
    "ResetType": "On"
}
```

6.5 响应

Redfish 中规定了下列四种响应：

- 元数据响应：描述通用客户端可以在服务器中访问的资源和类型；
- 资源响应：具体资源的 JSON 表达；
- 资源集合响应：代表资源集合的资源的 JSON 表达；
- 错误响应：顶级 JSON 响应，在出现 HTTP 错误时提供额外信息。

6.5.1 响应数据头

在响应消息中可以使用 HTTP 数据头。在下表中，将对这些数据头、以及对 Redfish 的要求做出规定。

- 如果下表中“要求”一栏中的数值为“是”，则 Redfish 服务必须能够返回下表中列出的，使用 HTTP 1.1 规范确定的数据头；
- 如果下表中“要求”一栏中的数值为“否”，则 Redfish 服务应该能够返回下表中列出的，使用 HTTP 1.1 规范确定的数据头；

- Redfish 客户端必须能够理解并处理下表中列出的，使用 HTTP 1.1 规范确定的数据头。

数据头	要求	支持数值	描述
OData-Version	是	4.0	描述响应消息中负载所遵守的 OData 协议版本号。
Content-Type	是	RFC 7231	用于描述消息体所使用的资源表达的媒体类型。在以 JSON 形式返回资源时，应该规定 Content-Type 为 application/json；在以 XML 形式返回资源时应该将其规定为 application/xml；另外，如果在请求的 Accept 数据头在规定的特定的媒体类型，则还应该在数据头的后面加上;charset=utf-8。
Content-Encoding	否	RFC 7231	用户描述对相关媒体类型执行的编码操作。
Content-Length	否	RFC 7231	用于描述消息体的长度。另外一种说明消息体长度的可选方法为使用 Transfer-Encoding：分块编码，从而不需要使用 Content-Length 数据头。如果服务器不支持 Transfer-Encoding，而需要使用 Content-Length 进行替代，则服务其将会使用状态代码 411 进行响应。
ETag	有条件	RFC 7232	一个用于识别资源具体版本的识别符，通常是消息摘要。在对针对 ManagerAccount 目标的 GET 请求的响应中，应该包括 ETag。
Server	是	RFC 7231	在跟踪产品令牌和其版本时需要。可以在其中列出多个产品令牌。
Link	是	参阅“Link 数据头”一条	在“Link 数据头”一条中所规定的情况下，在响应中应该返回 Link 数据头。
Location	有条件	RFC 7231	表示可以用于请求资源表达的 URI。在新资源被创建的情况下，应该返回此数据头。在创建用户会话的响应中，应该返回 Location 和 X-Auth-Token 数据头。
Cache-Control	是	RFC 7234	必须包括此数据头，此数据头用于说明该响应是否可以被缓存。
Via	否	RFC 7230	用于说明网络的层级结构，识别消息循环。每次通过时都会插入其 VIA 值。
Max-Forwards	否	RFC 7231	限制网关和跳过代理。防止消息在网络中无限停留。
Access-Control-	是	W3C CORS 第	用于防止或者允许从初始域中提出的请求，用于防止

Allow-Origin		5.1 节	CSRF 攻击。
Allow	是	POST、PUT、PATCH、DELETE、HEAD	必须在响应中返回 405 代码（方法不允许），同时说明具体请求 URI 允许使用的有效方法。对于任何 GET 或 HEAD 操作请求而言，应该返回此数据头，说明此资源允许使用的其他方法。
WWW-Authenticate	是	RFC 7235 第 4.1 节	在实施了基本和其他可选验证机制时需要提供。更多详细信息参阅“安全性”一条相关内容。
X-Auth-Token	是	不透明的编码八字节字符串	用于用户会话的认证。令牌值必须为不能辨别的随机值。
Retry-After	否	RFC 7231 第 7.1.3 节	用于告知客户端，在再次发出任务信息请求之前需要等待多长的时间。

6.5.1.1 Link 数据头

在对 HEAD 或 GET 操作的响应中，Link 数据头用于提供关于被访问资源的元数据信息。除了来自于资源的超链接之外，在针对资源的 JSON schema 的 URL 中应该带有 rel=describedby。针对注释 JSON schema 的 URL 中不应该带有 rel=describedby。如果引用的 JSON schema 是一个带版本号的 schema，则应该与资源返回的@odata.id 属性数值中所包含的版本号一致。

下面是一个具有管理员权限、带有设置注释的 ManagerAccount 的 Link 数据头示例：

- 第一种 Link 数据头是一个来自于资源的超链接示例。其中描述了资源内部的超链接。此种类型的数据头不在本规范文件的适用范围之内。
- 第二种 Link 数据头是注释 Link 数据头的示例，因为其引用了用于描述注释的 JSON schema，因此其中不带有 rel=describedby。此示例中引用了 DMTF 的 Redfish schema 库中的注释的公开副本。
- 第三种 Link 数据头是用于描述实际资源的 JSON schema 示例。
- 需要注意的一点是，在 URL 中可以引用不带版本号的 JSON schema（因为资源中的@odata.type 属性将会显示正确的版本号）、也可以引用带有版本号的 JSON schema（根据前文中的强制性要求，此版本号必须与资源中的@odata.type 属性中规定的版本号相一致）。

```
Link: </redfish/v1/AccountService/Roles/Administrator>; path=/Links/Role
Link: <http://redfish.dmtf.org/schemas/Settings.json>
Link: </redfish/v1/JsonSchemas/ManagerAccount.v1_0_2.json>; rel=describedby
```

针对 GET 和 HEAD 请求必须返回一个针对 HEAD 请求的 Link 数据头，以及一个满足 rel=describedby 要求的数据头；或者返回一个满足注释要求的 Link 数据头。

6.5.2 状态代码

在响应消息中可以返回 HTTP 状态代码。

如果相关 HTTP 状态代码表示的是错误状态，则响应消息体中应该含有一个扩展错误资源，以便为客户端提供更多更有意义和确定性的错误语义。

- 在返回编号为 400 或以上的状态代码时，服务器必须根据本规范相关规定，在响应消息体中返回扩展错误资源。在返回其他编号的状态代码时，如果允许在响应消息体中发送相关代码和操作，则服务可以根据本规范相关规定，在响应消息体中返回扩展错误资源。
- 在认证失败的情况下，扩展错误资源中绝对不得提供任何特权信息。

备注：更多关于扩展错误安全性方面的信息，请参与本规范“安全性”一条相关内容。

下表中列出了一部分常用的 HTTP 状态代码。在适宜的情况下，服务器也可能返回其他状态代码。相关状态代码的具体描述、以及本规范文件中提出的一些额外要求，请参阅表格中“描述”一栏相关内容。

- 客户端必须能够理解并处理下表中列出的、根据 HTTP 1.1 规范定义的状态代码，同时必须接受本规范文件中提出的其他要求的限制；
- 服务器必须能够正确地使用这些状态代码进行响应；
- 操作中出现的异常情况必须映射到 HTTP 状态代码中；
- Redfish 服务不得返回 100 错误代码。除了上传大量数据的情况之外，应该尽可能避免将使用 HTTP 协议进行多通道数据传输。

HTTP 状态代码	描述
200，正常	相关请求成功完成，在其消息体中包括资源表达。
201，已创建	创建新资源的请求已经成功完成。其 Location 数据头应该设置为新创建资源的规定 URI。在响应消息体中应该包括新创建资源的表达。
202，已接受	相关请求已经被接受，正在处理，但处理过程尚未完成。其 Location 数据头应该设置为任务资源的 URI，以便在后续可以通过查询确定操作的状态。在响应消息体中应该包括任务资源的表达。
204，无内容	请求已成功，但在响应消息体中没有包含内容。
301，被永久移动	请求的资源对应另一个不同的 URI。
302，已发现	请求的资源临时对应另一个不同的 URI。
304，未修改	服务器执行了一次有条件 GET 请求，相关资源允许访问，但不能修改资源的内容。有条件请求应该使用 If-Modified-Since 以及/或者 If-None-Match（参阅 HTTP 1.1 第 14.25 节和第 14.26 节相关内容）数据头发起，以便在没有修改的情况下可以节省网络带宽。
400，错误请求	相关请求无法被处理，因为其中信息丢失或者含有不正确的信息（例

	如：输入栏内验证错误、要求的数值缺失等等）。在响应消息体中必须返回一个扩展错误，更多信息参阅“错误响应”一条相关内容。
401，未授权	此请求中的认证证书缺失或者不正确。
403，禁止	服务器承认请求中包含的证书，但该证书不具有执行此请求的授权。
404，未发现	请求中所指定的资源的 URI 不存在。
405，方法不允许	此请求 URI 不支持请求中提出的 HTTP 操作（例如：DELETE、GET、HEAD、POST、PUT、PATCH 等）。响应中必须包括 Allow 数据头，其中应该列出通过请求 URI 识别到的资源所支持的操作方法列表。
406，不接受	请求中规定了 Accept 数据头，但通过此请求识别到的资源不能生成对 Accept 数据头中规定的媒体类型相对应的资源表达。
409，冲突	创建或者更新请求不能完成，因为这可能平台支持的当前资源状态发生变化（例如：试图使用不兼容的数值设置以互相链接的方式运行的多个属性）。
410，消失	请求的资源不再保存在服务器上，且不知道其转递地址。如果预计这种情况将是永久性的，则具有超链接编辑能力的客户端应该在用户审批之后，删除对请求的 URI 的引用。如果服务器不知道该条件是否是永久性的，也没有相关的检测设施，则应该使用 404 代码（未发现）来进行替代。除非其他地方另有明确规定，否则此响应应该可以缓存。
411，要求的长度	请求中没有使用 Content-Length 数据头规定内容的长度（可能使用了 Transfer-Encoding：分块编码数据头替代）。但识别到的资源需要 Content-Length 数据头。
412，前提条件不满足	前提条件（例如：OData-Version、If-Match 或 If-Not-Match 数据头）检查失败。
415，不支持的媒体类型	请求中规定了不被支持的 Content-Type 数据头。
428，需要前提条件	请求中没有说明需要满足的前提条件，例如 If-Match 或 If-None-Match 数据头。
500，内部服务器错误	服务器遇到预料之外的情况，使其无法成功执行请求。在响应消息体中必须返回一个扩展错误，更多信息参阅“错误响应”一条相关内容。
501，未实施	服务器（当前）不支持请求中提出的功能。在服务器不支持请求中提出的方法、且不能支持任何资源的方法时，这是最适宜的响应。
503，服务不可用	用于临时性过载、或者服务器故障，服务器当前不能处理相关请求。服务器可以使用此代码来响应服务器对其执行初始化或其他资源维护的有效 URI 请求。

6.5.3 元数据响应

元数据的主要功能是用于向普通用户描述资源、资源集合、以及与服务相关的行为，包括对本规范文件没有具体了解的 OData 客户端工具和应用。如果其已经充分理解了目标服务，则客户端不需要提出元数据请求。例如，请求并解释本规范中规定的资源的 JSON 表达。

6.5.3.1 服务元数据

服务元数据能够根据 OData schema 要求，对顶层的资源和资源类型进行描述。Redfish 服务元数据使用 XML 文件进行表达，其中带有一个在“http://docs.oasis-open.org/odata/ns/edmx”中进行定义，名为“Edmx”的根元素；一个命名空间；以及一个取值为“4.0”的 OData 版本属性。

```
<edmx:Edmx xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx" Version="4.0">
  <!-- edmx:Reference and edmx:Schema elements go here -->
</edmx:Edmx>
```

6.5.3.1.1 引用其他 schema

服务元数据中包括每种 Redfish 资源类型的命名空间，以及“RedfishExtensions.v1_0_0”命名空间。这些引用可以使用主机 Redfish schema 的标准 URI（即：http://redfish.dmtf.org/schemas），或者与主机版本相同的本地版本的 Redirect schema 的 URI。

```
<edmx:Reference Uri="http://redfish.dmtf.org/schemas/v1/AccountService_v1.xml">
  <edmx:Include Namespace="AccountService"/>
  <edmx:Include Namespace="AccountService.v1_0_0"/>
</edmx:Reference>
<edmx:Reference Uri="http://redfish.dmtf.org/schemas/v1/ServiceRoot_v1.xml">
  <edmx:Include Namespace="ServiceRoot"/>
  <edmx:Include Namespace="ServiceRoot.v1_0_0"/>
</edmx:Reference>

...

<edmx:Reference Uri="http://redfish.dmtf.org/schemas/v1/VirtualMedia_v1.xml">
  <edmx:Include Namespace="VirtualMedia"/>
  <edmx:Include Namespace="VirtualMedia.v1_0_0"/>
</edmx:Reference>
<edmx:Reference
Uri="http://redfish.dmtf.org/schemas/v1/RedfishExtensions_v1.xml">
  <edmx:Include Namespace="RedfishExtensions.v1_0_0" Alias="Redfish"/>
</edmx:Reference>
```

服务器的元数据文件中应该包括一个实体容器，对顶层的资源和资源集合进行定义。在实施过程中，可以将服务容器扩展为定义为服务根目录 schema，作为实施的 OData 服务文件。

```
<edmx:DataServices>
  <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="Service">
    <EntityContainer Name="Service"
Extends="ServiceRoot.v1_0_0.ServiceContainer"/>
  </Schema>
</edmx:DataServices>
```

6.5.3.1.2 引用 OEM 扩展

元数据文件还可以引用其他 schema 文件，用于描述服务所使用的原始设备制造商的具体扩展，例如其他额外资源集合的自定义类型。

```
<edmx:Reference Uri="http://contoso.org/Schema/CustomTypes">
  <edmx:Include Namespace="CustomTypes"/>
</edmx:Reference>
```

6.5.3.1.3 注释

服务器可以使用 Redfish 规定的或者自定义的注释条目，对各种组合、类型、动作和参数进行注释。这些注释条目通常位于一个单独的注释文件中，使用 IncludeAnnotations 指令，从服务元数据文件中进行引用。

```
<edmx:Reference Uri="http://service/metadata/Service.Annotations">
  <edmx:IncludeAnnotations TermNamespace="Annotations.v1_0_0"/>
</edmx:Reference>
```

在注释文件中，应该说明需要注释的目标 Redfish schema 要素、实施的属性、以及每个属性的数值：

```
<Annotations Target="ComputerSystem.Reset/ResetType">
  <Annotation Term="Annotation.AdditionalValues">
    <Collection>
      <String>Update and Restart</String>
      <String>Update and PowerOff</String>
    </Collection>
  </Annotation>
</Annotations>
```

6.5.3.2 OData 服务文件

OData 服务文件的主要作用是为普通 OData 客户端提供顶层接入点。

```
{}{
    "@odata.context": "/redfish/v1/$metadata",
    "value": [
        {}{
            "name": "Service",
            "kind": "Singleton",
            "url": "/redfish/v1/"
        },
        {}{
            "name": "Systems",
            "kind": "Singleton",
            "url": "/redfish/v1/Systems"
        },
        {}{
            "name": "Chassis",
            "kind": "Singleton",
            "url": "/redfish/v1/Chassis"
        },
        {}{
            "name": "Managers",
            "kind": "Singleton",
            "url": "/redfish/v1/Managers"
        },
        ...
    ]
}
```


OData 服务文件应该使用 MIME 类型 application/json 以 JSON 对象的方式返回。

JSON 目标中应该含有一个名称为“@odata.context”的环境属性，其数值为“/redfish/v1/$metadata”。此环境属性可以让普通的 OData 客户端知道如何寻找描述服务器可以接触的资源类型的服务元数据。

JSON 对象中应该包括一个名词为“value”的属性，其取值为含有服务器根目录条目、以及服务网根目录直接子集的每种资源条目的 JSON 数组。

每个条目都必须使用一个 JSON 对象进行表达，其中应该包括一个“name”属性，其取值为资源的一个用户友好的名称；一个“kind”属性，对于普通资源（包括资源集合）而言其取值为“Singleton”、对于顶层资源集合而言，其取值为“EntitySet”；还有一个“url”属性，其取值为对应的顶层资源的 URL。

6.5.4 资源响应

资源响应将使用 MIME 型 application/json，以 JSON 消息负载的方式返回。资源属性名称必须与 schema 中规定的名称一致。

同时参阅“资源集合响应”一条相关内容。

6.5.4.1 环境属性

用于表达单个资源的响应中应该包括有一个环境属性，名称为“@odata.context”，用于描述负载的来源。环境属性的取值应该为根据 OData 协议描述资源的环境 URL。

资源的环境 URL 可以采用下列两种形式中的任何一种：

MetadaUrl#ResourceType

或者

MetadaUrl#ResourcePath/$entity

其中：

- MetadaUrl 是指服务器的元数据 URL（/redfish/v1/$metadata）；
- ResourceType 是指不带版本号的资源类型的完整限定名称；
- ResourcePath 是指从服务器根目录通往单例资源、或者含有资源的资源集合的路径；
- $entity 是一个标识符，用于响应来自于一个实体组合、或者通过浏览属性发现的资源。

例如，下列环境 URL 标识的是含有单个 ComputerSystem 资源的响应结果：

```
{}{
    "@odata.context":"/redfish/v1/$metadata#ComputerSystem.ComputerSystem",
    ...
}
```

6.5.4.2 资源识别符属性

在资源响应中，应该包括一个独特的识别符属性，其名称为“@odata.id”。识别符属性的取值应该为资源的独特识别符。

资源识别符应该根据 RFC 3986 第 3.3 节-“路径”中规定的 URI 路径规则，以字符串的形式在 JSON 负载中进行表达。与请求 URI 具有相同授权的资源应该使用上述规范中规定的绝对路径规则进行表达，即必须随时以单斜杠（“/”）开始。与请求 URI 具有不同授权的资源，必须随时以双斜杠（“//”）开始，之后加上资源路径和授权信息。

资源识别符是资源的典范 URL，在适当时可以用于检索或者编辑资源。

6.5.4.3 类型属性

所有资源响应中都应该带有一个名称为“@odata.type”的类型属性。响应中的所有嵌入对象中都应该包括一个名称为“@odata.type”的类型属性。类型属性的取值应该为规定了资源类型的 URL 碎片，具体将在元数据文件中进行定义、或者也可以引用元数据文件，其具体形式为：

#Namespace.TypeName

其中：

- Namespace 是指规定了资源类型的 Redfish schema 的完整命名空间名称。对于 Redfish 资源而言，将为带有版本号的命名空间名称；
- TypeName 是指资源类型的名称。

6.5.4.4 ETag 属性

ETag 能够有条件的检索或者更新资源。在资源中应该包括一个名称为“@odata.etag”的 ETag

属性。ETag 属性的取值为资源的 ETag。

6.5.4.5 原生属性

应该根据下表的相关要求，以 JSON 数值的方式返回原生属性：

类型	JSON 表达
Edm.Boolean	布尔值
Edm.DataTimeOffset	字符串，具体格式将在“日期时间数值”一条中做出规定
Edm.Demical	数值，其中可以带有小数点
Edm.Double	数值，其中可以带有小数点，还可以带有指数
Edm.Guid	字符串，其具体格式为 ([0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12})
Edm.Int64	不带有小数点的数值
Edm.String	字符串

在从客户端处接收到数值之后，服务器必须支持规定的 JSON 类型中的其他有效表达。更加具体地讲，服务器必须支持以指数形式表达的有效整数和小数，以及带有小数点和没有非零尾数的整数值。

6.5.4.5.1 日期时间数值

日期时间数值必须根据 ISO 8601 中规定的“扩展”格式，以 JSON 字符串的方式返回，其中应该带有时间补偿或者 UTC 后缀，应该采用下列形式：

```
*YYYY*-*MM*-*DD* T *hh*:*mm*:*ss*[.*SSS*] (Z | (+ | - ) *hh*:*mm*)
```

其中：

- SSS 是指一个或多个数字，表示一秒钟内小数部分，同时应该带有表示精度的数值；
- 分隔符“T”和后缀“Z”应该大写。

6.5.4.6 结构属性

定义为复数型或扩展资源类型的结构属性应该以 JSON 对象的方式返回。JSON 对象的具体类型将在 Redfish schema 对含有结构值的属性的定义中做出规定。

6.5.4.7 动作属性

资源可用的动作应该使用名称为“动作”的资源中的单个结构属性下的多个独立属性进行表达。

6.5.4.7.1 动作表达

动作应该使用“动作”资源中的属性进行表达，属性的名称应该为可以识别该动作的独特 URI。此 URI 应该采用下列形式：

#Namespace.ActionName

其中：

- Namespace 是指对动作进行定义的 Redfish schema 中使用的命名空间。对于 Redfish 资源而言，这是一个与版本号无关的命名空间；
- ActionName 是指动作的名称。

客户端可以使用此碎片，根据引用的 Redfish schema 文件中对相关命名空间的定义，来识别动作的定义。

属性的取值应该为一个 JSON 对象，其中含有一个名称为“target”的属性，其取值为用于调用相关动作的相对或绝对 URL。“target”属性将在 OData JSON 格式规范中进行定义。

用于表达可用动作的属性应该在“允许值”注释中进行必要注释，以便列出具体参数可以采取的允许值。

例如，下列属性用于表达重置（Reset）动作，在 ComputerSystem 的命名空间中应该进行下列定义：

```
{}{
    "#ComputerSystem.Reset": {
        "target":"/redfish/v1/Systems/1/Actions/ComputerSystem.Reset",
        "ResetType@Redfish.AllowableValues": [
            "On",
            "ForceOff",
            "GracefulRestart",
            "GracefulShutdown",
            "ForceRestart",
            "Nmi",
            "ForceOn",
            "PushPowerButton"
        ]
    },
    ...
}
```

在这种情况下，客户端可以在/redfish/v1/Systems/1/Actions/ComputerSystem 调用 POST 请求。使用下列消息体进行重置：

```
{}{
    "ResetType": "On"
}
```

6.5.4.7.2 允许值

用于表达可用动作的属性应该在“允许值”注释中进行必要注释，以便列出具体参数可以采取的允许值。

应该在属性中对属性的允许值做出规定，其名称即为在“@Redfish.AllowableValues”之后的参数的名称，其取值应该为代表相关参数允许值的一个 JSON 字符串数组。

6.5.4.8 链接属性

对其他资源的引用应该通过资源中的链接属性进行表达。

链接属性的名称应该为“Links”，其中应该针对每个未包含的引用属性各设置一个属性，具体根据 Redfish schema 对相关类型的定义确定。对于单个有数值的引用属性而言，此属性的取值应该为相关单一资源的 id。对于有数值的引用属性集合而言，此属性的取值应该为相关

资源的 id 数组。

另外，在 OEM 属性中也可以纳入链接属性，以便浏览供货商特有的超链接。

6.5.4.8.1 引用单一关联资源

对单一资源的引用应该以含有单一资源识别符属性的 JSON 对象的形式返回，该属性的名称为关系的名称，其取值为被引用资源的 URI。

```
{}{
    "Links" : {
        "ManagedBy": {
            "@odata.id":"/redfish/v1/Chassis/Encl1"
        }
    }
}
```

6.5.4.8.2 相关资源引用数组

对一组零或其他相关资源的引用，将以 JSON 对象数组的形式返回，其名称为关系的名称。数组中每个成员都是一个含有单一资源识别符属性的 JSON 对象，其取值为被引用资源的 URI。

```
{}{
    "Links" : {
        "Contains" : [

                {}{
                    "@odata.id":"/redfish/v1/Chassis/1"
                },
                {}{
                    "@odata.id":"/redfish/v1/Chassis/Encl1"
                }
            ]
        }
    }
```

6.5.4.9 OEM 属性

原始设备制造商的具体属性可以纳入到 OEM 属性中。

6.5.4.10 部分资源属性

用于表达单个资源的响应不得分为多个结果。更多关于此类响应格式的详细信息，参阅“部分结果”一节相关内容。

6.5.4.11 扩展信息

响应对象中可以包括扩展信息，例如关于不能被更新的属性的信息。此信息可以通过适用于 JSON 响应中的某个具体属性、或者适用于整个 JSON 对象的注释进行表达。

6.5.4.11.1 扩展对象信息

JSON 对象可以使用“@Message.ExtendedInfo”进行注释，以便规定对象层面的状态信息。

```
{}{
    "@odata.context": "/redfish/v1/$metadata#SerialInterface.SerialInterface",
    "@odata.id": "/redfish/v1/Managers/1/SerialInterfaces/1",
    "@odata.type": "#SerialInterface.v1_0_0.SerialInterface",
    "Name": "Managed Serial Interface 1",
    "Description": "Management for Serial Interface",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "InterfaceEnabled": true,
    "SignalType": "Rs232",
    "BitRate": "115200",
    "Parity": "None",
    "DataBits": "8",
    "StopBits": "1",
    "FlowControl": "None",
    "ConnectorType": "RJ45",
    "PinOut": "Cyclades",
    "@Message.ExtendedInfo" : [
        {}{
            "MessageId": "Base.1.0.PropertyDuplicate",
            "Message": "The property InterfaceEnabled was duplicated in the
request.",


             "RelatedProperties": [
                 "#/InterfaceEnabled"
             ],
             "Severity": "Warning",
             "Resolution": "Remove the duplicate property from the request body
 and resubmit the request if the operation failed."
         }
    ]
}
```


此属性的数值应该为一个消息对象数组。

6.5.4.11.2 扩展属性信息

JSON 对象内的具体属性可以使用“@Message.ExtendedInfo”，通过扩展信息进行注释，但需要预先考虑属性的名称。

```
{}{
    "@odata.context": "/redfish/v1/$metadata#SerialInterface.SerialInterface",
    "@odata.id": "/redfish/v1/Managers/1/SerialInterfaces/1",
    "@odata.type": "#SerialInterface.v1_0_0.SerialInterface",
    "Name": "Managed Serial Interface 1",
    "Description": "Management for Serial Interface",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "InterfaceEnabled": true,
    "SignalType": "Rs232",
    "BitRate": 115200,
    "Parity": "None",
    "DataBits": 8,
    "StopBits": 1,
    "FlowControl": "None",
    "ConnectorType": "RJ45",
    "PinOut": "Cyclades",
    "PinOut@Message.ExtendedInfo" : [
        {}{
            "MessageId": "Base.1.0.PropertyValueNotInList",
            "Message": "The value Contoso for the property PinOut is not in the
list of acceptable values.",
            "Severity": "Warning",
            "Resolution": "Choose a value from the enumeration list that the
implementation can support and resubmit the request if the operation failed."
        }
    ]
}
```

此属性的数值应该为一个消息对象数组。

6.5.4.12 其他注释

JSON 中的资源表达中也可以包括其他一些额外注释，使用名称为下列形式的属性进行表达：

[PropertyName]@NameSpace.TermName

其中：

- PropertyName 是指需要注释的属性的名称，在忽略此项的情况下，相关注释将适用于整个资源；
- NameSpace 是指对注释项做出定义的命名空间的名称。此命名空间必须被请求的环境 URL 中规定的元数据文件引用；
- TermName 是指适用于资源或者资源属性的注释项的名称。

客户端可以从服务元数据文件中调阅注释内容，也可以完全忽略注释内容。但客户端不能因为存在无法识别的注释，就导致无法读取资源，包括在 Redfish 命名空间内定义的新注释。

6.5.5 资源集合响应

资源集合将会以 JSON 对象的方式返回。JSON 对象中必须包括环境、资源计数、以及成员数组，还可以包括一个针对部分结果的下一链接属性。

6.5.5.1 环境属性

用于表达资源集合的响应中应该包括有一个环境属性，名称为“@odata.context”，用于描述

负载的来源。环境属性的取值应该为根据 OData 协议描述资源集合的环境 URL。

资源集合的环境 URL 可以采用下列两种形式中的任何一种：

MetadataUrl.#CollectionResourceTypeMetadataUrl.#CollectionResourcePath

其中：

- MetadaUrl 是指服务器的元数据 URL（/redfish/v1/$metadata）；
- CollectionResourceType 是指资源集合中不带版本号的资源类型的完整限定名称；
- CollectionResourcePath 是指从服务器根目录通往资源集合的路径。

6.5.5.2 计数属性

资源集合内可用的资源（成员）的总数量应该通过计数属性进行表达。计数属性的名称应该为“Members@odata.count”，其取值应该为资源集合内可用资源的总数量。此计数属性不得受到$top 或者$skip 等查询参数的影响。

6.5.5.3 成员属性

资源集合内的资源成员将会以 JSON 数组的方式返回，其中数组内的每个元素都是一个 JSON 对象，其类型将在描述每种所包含类型的 Redfish schema 文件中进行规定。用于表达资源集合内的成员的属性的名称应该为“Members”。成员属性不得为空值。对于空的资源集合而言，应该以 JSON 形式返回一个空数组。

6.5.5.4 下一链接属性和部分结果

在响应中还可以包括完整资源集合中成员的一个子集。对于只返回部分资源集合的响应而言，其中应该包括一个下一链接属性，此属性的名称为“Members@odata.nextLink”。下一链接属性的取值应该为资源的不透明 URL，具有相同的@data.type 属性。其中应该含有部分成员的下一个组合。只要在资源集合中的成员数量多于返回的成员数量时，才需要设置下一链接属性。

计数属性的数值表示的是在客户端枚举了资源集合的所有页面之后，获得的可用资源的总数量。

6.5.5.5 其他注释

JSON 中的资源集合表达中也可以包括其他一些额外注释，使用名称为下列形式的属性进行表达：

@NameSpace.TermName

其中：

- NameSpace 是指对注释项做出定义的命名空间的名称。此命名空间必须被请求的环境 URL 中规定的元数据文件引用；
- TermName 是指适用于资源集合的注释项的名称。

客户端可以从服务元数据文件中调阅注释内容，也可以完全忽略注释内容。但客户端不能因为存在无法识别的注释，就导致无法读取资源，包括在 Redfish 命名空间内定义的新注释。

6.5.6 错误响应

只使用 HTTP 响应状态代码通常并不能提供充分的信息，不能确保用户获得确定性的错误语义。例如，如果客户端执行 PATCH 操作，但其中有部分属性不匹配、其他属性不支持，则仅仅通过返回 HTTP 400 错误代码是无法向客户端说明具体哪些数值存在错误的。通过错误响应可以向客户端提供更有意义的确定性错误语义。

Redfish 服务在 HTTP 响应中提供多种错误响应，以便尽可能向客户端提供更多关于错误情况的信息。除此之外，服务器还可以提供 Redfish 标准错误代码、原始设备制造商规定的错误代码，或者同时提供上述两种错误代码，具体取决于实施过程中是否具有传递关于潜在错误的最有用信息的能力。

错误响应使用扩展错误资源进行定义，使用单一的 JSON 对象进行表达，其属性名称为“error”，具有下列属性：

属性	描述
code	用于表示来自消息寄存器的具体消息标识的字符串。只有在没有更好错误提示信息时，才应该使用“Base.1.0 基本错误”信息。
message	人类可读的错误消息，与消息寄存器中的错误消息一一对应。
@Message.ExtendedInfo	一个消息对象数组，用于描述一项或多项错误消息。

```
{}{
    "error": {
        "code": "Base.1.0.GeneralError",
        "message": "A general error has occurred. See ExtendedInfo for more
information.",
        "@Message.ExtendedInfo": [
            {}{
                "@odata.type" : "#Message.v1_0_0.Message",
                "MessageId": "Base.1.0.PropertyValueNotInList",
                "RelatedProperties": [
                    "#/IndicatorLED"
                ],
                "Message": "The value Red for the property IndicatorLED is not
in the list of acceptable values",
                "MessageArgs": [
                    "RED",
                    "IndicatorLED"
                ],
                "Severity": "Warning",
                "Resolution": "Remove the property from the request body and
resubmit the request if the operation failed"
            },
            {}{
                "@odata.type" : "#Message.v1_0_0.Message",
                "MessageId": "Base.1.0.PropertyNotWriteable",
                "RelatedProperties": [
                    "#/SKU"
                ],
                "Message": "The property SKU is a read only property and cannot
be assigned a value",
                "MessageArgs": [
                    "SKU"
                ],
                "Severity": "Warning",
                "Resolution": "Remove the property from the request body and
resubmit the request if the operation failed"
            }
        ]
    }
}
```

6.5.6.1 消息对象

消息对象中能够提供关于对象、属性或者错误响应的信息。

消息使用具有下列属性的 JSON 对象进行表达：

属性	描述
MessageId	用于表示具体错误或者消息的字符串（不能与 HTTP 错误代码相混淆）。此代码可以用于访问消息寄存器中的详细信息。
Message	人类可读的错误消息，表示与错误相关的语义。可以为完整的消息，与替换变量无关。
RelatedProperties	一个 JSON 指针的可选数组，用于定义消息描述的 JSON 负载内的具体属性。
MessageArgs	一个可选字符串数组，表示消息的替换参数值。对于参数化的消息而言，如果规定了 MessageId，则响应中必须包括此属性。

Severity	一个可选字符串，表示错误的严重程度。
Resolution	一个可选字符串，表示可以解决错误的建议动作。

每个消息对象实例中都应该包括至少一个 MessageId 属性、以及任何适用的 MessageArgs 属性，或者能够规定完整的人类可读的错误消息的其他消息属性。

MessageId 属性用于识别消息寄存器中规定的具体消息。

MessageId 属性的取值应该为下列形式：

RegistryName.MajorVersion.MinorVersion.MessageKey

其中：

- RegistryName 是指寄存器的名称，寄存器的名称必须使用 Pascal-case 命名；
- MajorVersion 是指代表寄存器主要版本的正整数；
- MinorVersion 是指代表寄存器次要版本的正整数；
- MessageKey 是寄存器内人类可读的密钥，消息密钥应该使用 Pascal-case 编码，其中不得包含空格、句号、或者特殊符号。

客户端可以通过 MessageId 来搜索消息寄存器，以便获得相应的消息。

消息寄存器方法具有国际标准化（因为寄存器可以便捷地转译）和容易实施（因为在实施过程中不需要纳入大型的字符串）等优点。

7. 数据模型和 Schema

Redfish 界面的一个核心设计理念就是让协议与数据模型分离。在本条中，将对通用数据模型、资源、以及 Redfish schema 等方面的要求做出规定。

- 每种资源都必须根据资源类型定义设置较强的类型性。具体类型应该在 Redfish schema 文件中进行定义，并使用独特的类型识别符进行识别。

7.1 Schema 库

SPMF 编制、审批和公布的所有 Redfish schemas 都将在 DMTF 网站上提供下载，具体网址为：http://redfish.dmtf.org/schemas。schema 库中的每个文件夹中都包含有 CSDL 和 JSON schema 格式。网站中的 schema 文件按照下表中列出的方式进行分类：

URL	文件夹内容
redfish.dmtf.org/schemas	每种 schema 文件的当前版本（包括最新的次要版本和勘误版本）。
redfish.dmtf.org/schemas/v1	所有 v1.xx schema 文件，每个 schema 文件的

	每种 v1.xx 次要版本或勘误版本补丁。
redfish.dmtf.org/schemas/archive	包括有每个具体版本的 schema 文件的子文件夹。

7.1.1 Schema 文件命名公约

Schema 库中公布的，以及其他人编制并重新发布的标准 Redfish schema 文件必须遵守一组命名公约。这些命名公约的目的在于确保命名的一致性，消除命名冲突的情况。

7.1.1.1 CSDL（XML）schema 文件命名

Redfish CSDL schema 文件应该使用类型名称（TypeName）属性的值进行命名，后面接“_v”和 schema 文件的主要版本号。对于包含了所有次要版本的单一 CSDL schema 文件而言，在文件名中只需要列出主要版本号。因此，schema 文件的名称应该为下列形式：

TypeName_vMajorVersion.xml

例如，1.3.0 版的 Chasis schema 文件应该命名为“Chasis_v1.xml”。

7.1.1.2 JSON schema 文件命名

Redfish JSON schema 文件应该使用类型识别符（Type indentifiers）进行命名，采用下列形式：

ResourceTypeName.vMajorVersion_MinorVersion_Errata.json

例如，1.3.0 版的 Chasis schema 文件应该命名为“Chasis_v1_3_0.json”。

7.1.1.3 OEM schema 文件命名

为了避免当前或未来的标准 Redfish schema 文件命名出现冲突，第三方编制的 Redfish schema 文件命名过程中，应该在命名空间内预先加上第三方组织机构的名称，例如，“ContosoDisk_v1/xml”或者“ContosoDisk.v1.0.4.json”。

7.1.2 schema 文件的程序访问

可以通过永久不变的 redfish.dmtf.org/schemas/v1 URL，使用程序来访问 schema 库，此文件夹中包括有每种 schema 的每种已发行版本的文件。对于需要使用 schema 文件的程序而言，应该采用本地 schema 缓存方法，以降低延迟、减少程序的互联网连接要求、以及减少 DMTF 网站的流量负担。

7.2 类型识别符

类型通过类型 URI 进行识别，类型 URI 的形式如下所述：

#Namespace.TypeName

其中：

- Namespace 是指对类型做出规定的命名空间的名称；
- TypeName 是指类型的名称。

本规范规定的类型的命名空间应该采用下列形式：

ResourceTypeName.vMajorVersion_MinorVersion_Errata

其中：

- ResourceTypeName 是指资源类型的名称。对于结构化（复式）类型、枚举和动作而言，此属性通常是指所含有的资源类型的名称；
- MajorVersion 是一个整数，表示以向后兼容的方式，对分级中的部分内容进行了变更；
- MinorVersion 是一个整数，表示可以加入新功能，但不能删除任何内容。应该保留与之前次要版本之间的兼容性；
- Errata 是一个整数，表示之前版本中的部分内容损坏，需要进行勘误和修改。

有效的类型命名空间示例为“ComputerSystem.v1_0_0”。

7.2.1 JSON 中的类型识别符

在 JSON 负载中使用的类型应该通过、或者引用服务元数据进行定义。

在 JSON 文件中，应该使用完整（带有版本号）的命名空间名称来引用本规范中规定的资源类型。

备注：数据模型和 schema 安全性方面的信息，请参阅本规范“安全性”一条相关内容。

7.3 通用命名公约

Redfish 界面应该具有易读性和直观性。因此，一致性能够帮助不太熟悉界面的用户了解新发现的属性及其用途。尽管下文这些规则不能替代 Redfish 规范和 Redfish schema 中的强制性要求，但这些规则有助于提高界面的易读性和用户友好性。

资源名称、属性名称、以及诸如枚举等常量都应该使用 Pascal case 编码。

- 每个词中的第一个字母应该大写，词之间的空格应该删除（例如：PowerState、SerialNumber）；
- 不得使用下划线；

- 对于两个字符的缩写词而言，两个字符都应该大写（例如：IPAddress、RemoteIP）；
- 对于包含了三个或以上字符的缩写词而言，只有第一个字母需要大写，除非第一个词是 Pascal case 编码的识别符（例如：Wwn、VirtualWwn）。

允许出现的例外情况包括：

- 著名的技术名称，例如“iSCSI”；
- 产品名称，例如“iLO”；
- 著名的缩写或缩略语。

对于带有单位、或者具有其他特殊含义的属性而言，在名称的后面应该加上单位识别符。当前可用的列表包括：

- 带宽（Mbps），（例如：端口速度 Mbps）；
- CPU 速度（Mhz），（例如：处理器速度 Mhz）；
- 存储容量（兆字节，MB），（例如：内存 MB）；
- 项目计数（计数值），（例如：处理器计数、风扇计数）；
- 资源状态（状态），（例如：电源状态）；
- 正在“工作中”的状态值应该以（ing）结尾（例如：应用中（Applying）、清洁中（Clearing））。

7.4 本地化问题

本地化、数据或元数据的翻译不在本 Redfish 规范 1.0 版本的范围之内。尽管如此，属性的名称绝对不允许本地化。

7.5 schema 定义

具体的资源、以及其相关的类型和动作将在 Redfish schema 文件中进行定义。

7.5.1 通用注释

所有 Redfish 类型和属性中都应该包括“描述”和“长描述”注释内容。

7.5.1.1 描述

描述注释可以适用于任何类型、属性、动作或参数，主要功能是对 Redfish schema 要素提供人类可读的描述内容。

描述注释将在下列文件中进行定义：http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Core.V1.xml。

7.5.1.2 长描述

长描述注释项可以应用于任何类型、属性、动作或参数，主要目的在于为相关 schema 要素提供正式、强制性的规范要求。在 Redfish schema 文件的长描述中提到“必须”字样时，说明服务器必须遵守相关的强制性要求。

长描述注释项将在下列文件中进行定义 http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Core.V1.xml：。

7.5.2 schema 文件

在 Redfish schema 根据 OData schema 进行的 OData schema 表达中，具体的资源将被定义为实体类型。表达中可以包括注释内容，以便自动生成能够验证 JSON 负载的 Redfish schema 的 JSON schema 表达。

7.5.2.1 schema 修改规则

在实施过程中引用的 schema，不论是引用自 OData 服务文件、还是引用自 JSON schema 文件表达，都可以与 Redfish schema 或其他实体规定的这些 schema 的典型定义有所不同，但是必须遵守如下所述的各项规则。客户端在视图操作通过 schema 定义的资源时，也必须考虑下列规则。

- 可以对 schema 进行修改，将读/写属性修改为只读属性；
- 可以对 schema 进行修改，删除部分属性；
- 可以对 schema 进行修改，将任何“引用 URI”修改为指向遵守修改规则的 schema；
- 不允许对 schema 进行其他任何修改。

7.5.2.2 schema 版本要求

OData schema 表达文件中的外部元素应该为 Edmx 要素，应该具有取值为“4.0”的 Version 属性。

```
<edmx:Edmx xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx" Version="4.0">
  <!-- edmx:Reference and edmx:DataService elements go here -->
</edmx:Edmx>
```

7.5.2.3 引用其他 schema

Redfish schema 可以引用其他 schema 文件中定义的类型。在 OData schema 表达中，可以通过纳入 Reference 要素来实现。在 JSON schema 表达中，可以使用“$ref”属性来实现。

在 Reference 要素中应该规定 OData schema 表达文件的 URI，以便描述被引用的类型；同时还应该设置一个或多个 Include 子要素，以便规定含有被引用类型的 Namespace 属性，以及相关命名空间的可选 Alias 属性。

对于普通类型注释项而言，类型定义通常引用 OData 和 Redfish 命名空间，对于基本类型而

言，资源类型定义通常引用 Redfish Resource.v1_0_0 命名空间。包括诸如温度、速度、或者尺寸等检测数据的 Redfish OData schema 表达中，通常需要包括“OData 检测数值命名空间”。

```
  <edmx:Reference Uri="http://docs.oasis-
open.org/odata/odata/v4.0/cs01/vocabularies/Org.OData.Core.V1.xml">
    <edmx:Include Namespace="Org.OData.Core.V1" Alias="OData"/>
  </edmx:Reference>
  <edmx:Reference
    Uri="http://docs.oasis-
open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Measures.V1.xml">
    <edmx:Include Namespace="Org.OData.Measures.V1" Alias="Measures"/>
  </edmx:Reference>
  <edmx:Reference
Uri="http://redfish.dmtf.org/schemas/v1/RedfishExtensions_v1.xml">
    <edmx:Include Namespace="RedfishExtensions.v1_0_0" Alias="Redfish"/>
  </edmx:Reference>
  <edmx:Reference Uri="http://redfish.dmtf.org/schemas/v1/Resource_v1.xml">
    <edmx:Include Namespace="Resource"/>
    <edmx:Include Namespace="Resource.v1_0_0"/>
  </edmx:Reference>
```

7.5.2.4 命名空间定义

资源类型在 OData schema 表达中的命名空间内定义。通过含有用于说明 schema 的 Namespace 和本地 Alias 的属性的 Schema 要素来定义命名空间。

OData schema 元素是 DataServices（数据服务）元素的子元素，该元素又是 Edmx 元素的子元素。

```
  <edmx:DataServices>
    <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm"
Namespace="MyTypes.v1_0_0">

      <!-- Type definitions go here -->

    </Schema>
  </edmx:DataServices>
```

7.5.3 资源类型定义

资源类型使用 EntityType 要素中的命名空间进行定义。其中的 Name 属性定义资源的名称，BaseType 属性定义资源的基本类型（如果有）。

在 Resource.v1_0_0 的命名空间中，需要通过通用资源基本类型获得的 Redfish 资源命名为“Resource”。

EntityType 中含有属性和引用属性要素，用于定义资源、以及用于描述资源的注释。

```
  <EntityType Name="TypeA" BaseType="Resource.v1_0_0.Resource">
    <Annotation Term="OData.Description" String="This is the description of
TypeA."/>
    <Annotation Term="OData.LongDescription" String="This is the specification
of TypeA."/>

    <!-- Property and Reference Property definitions go here -->

  </EntityType>
```

所有资源都必须包括描述和长描述注释项。

7.5.4 资源属性

资源的结构属性将使用 Property 要素进行定义。其中 Name 属性定义属性的名称，Type 属性定义属性的类型。

请求和响应 JSON 负载中的属性名称必须 Name 属性的取值相一致。

属性必须具有一个不可为空的取值，可为空属性的取值应该为“错误”。

```
  <Property Name="Property1" Type="Edm.String" Nullable="false">
    <Annotation Term="OData.Description" String="This is a property of TypeA."/>
    <Annotation Term="OData.LongDescription" String="This is the specification
of Property1."/>
    <Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
    <Annotation Term="Redfish.Required"/>
    <Annotation Term="Measures.Unit" String="Watts"/>
  </Property>
```

所有属性都必须包括描述和长描述注释项。

只读的属性应该使用数值为 OdataPermission/Read 的允许注释项进行注释。

需要所有服务实施的属性应该使用要求注释项进行注释。

带有单位的属性可以使用单位注释项进行注释。

7.5.4.1 属性类型

属性的类型使用 Type 属性进行定义。Type 属性的取值可以为原生类型，结构化类型，枚举类型，以及原生、结构化或枚举类型的集合。

7.5.4.1.1 原生类型

原生类型属性需要在前面加上“Edm”命名空间前缀。

Redfish 服务可以使用下列任何一种原生类型：

类型	含义
Edm.Boolean	正确或错误。
Edm.DateTimeOffset	带有时区的日期和时间数据。
Edm.Decimal	具有确定精度和范围和数值。
Edm.Double	IEEE 754 binary64 浮点数值（15-17 位小数位数）。

Edm.Guid	全球独特的识别符。
Edm.Int64	带正负号的 64 位整数。
Edm.String	一串 UTF-8 字符串。

7.5.4.1.2 结构化类型

结构化类型将在 ComplexType 要素的命名空间内进行定义。此要素的 Name 属性中将对结构化类型的名称做出规定。此要素中还可以包括一个 BaseType 属性，用于规定基本类型（如果有）。

结构化类型可以在不同资源类型的不同属性之间重复使用。

```
<ComplexType Name="PropertyTypeA">
  <Annotation Term="OData.Description" String="This is type used to describe a
structured property."/>
  <Annotation Term="OData.LongDescription" String="This is the specification
of the type."/>

  <!-- Property and Reference Property definitions go here -->

</ComplexType>
```

结构化类型中可以含有属性、引用属性和注释项。

结构化类型中必须包括描述和长描述注释项。

7.5.4.1.3 枚举类型

枚举类型将在 EnumType 要素的命名空间内进行定义。此要素的 Name 属性中将对枚举类型的名称做出规定。

枚举类型可以在不同资源类型的不同属性之间重复使用。

EnumType 要素中可以包含一个 Member 要素，用于规定枚举的成员。Member 要素中含有一个 Name 属性，用于规定成员名称的字符串取值。

```
<EnumType Name="EnumTypeA">
  <Annotation Term="OData.Description" String="This is the EnumTypeA
enumeration."/>
  <Annotation Term="OData.LongDescription" String="This is used to describe
the EnumTypeA enumeration."/>
  <Member Name="MemberA">
    <Annotation Term="OData.Description" String="Description of MemberA"/>
  </Member>
  <Member Name="MemberB">
    <Annotation Term="OData.Description" String="Description of MemberB"/>
  </Member>
</EnumType>
```

枚举类型中必须包括描述和长描述注释项。

枚举成员中必须包括描述注释项。

7.5.4.1.4 集合

type 属性中可以规定一个原生、结构化或枚举类型的集合。

取值为集合的 type 属性的取值应该采用下列形式：

Collection（*NamespaceQualifiedTypeName*）

其中 *NamespaceQualifiedTypeName* 是指原生、结构化或枚举类型的命名空间限定名称。

7.5.4.2 其他属性

可以使用 AdditionalProperties 注释项来说明某个类型中是否含有本规范规定范围之外的其他属性。使用 AdditionalProperties 注释项注释的类型如果取值为“False”，则其中绝对不允许含有其他属性。

```
<Annotation Term="OData.AdditionalProperties"/>
```

AdditionalProperties 注释项的定义参阅：

https://tools.oasis-open.org/version-control/browse/wsvn/odata/trunk/spec/vocabularies/Org.OData.Core.V1.xml

7.5.4.3 不可为空属性

属性中还可以设置一个 Nullable 属性，如果其取值为“false”，则说明该属性中不能含有空值。Nullable 属性取值为“true”、或者不含有 Nullable 属性的属性可以接受空值。

```
<Property Name="Property1" Type="Edm.String" Nullable="false">
```

7.5.4.4 只读属性

对于取值为 OData.Permission/Read 的属性而言，可以使用 Permissions 注释项，以便将其规定为只读属性。

```
<Annotation Term="OData.Permissions" EnumMember="OData.Permission/Read"/>
```

Permissions 注释项的定义如下所述：

http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Core.V1.xml.

7.5.4.5 要求属性

可以使用 Required 注释项来规定要求服务器支持相关属性。要求属性应该使用 Required 注释项进行注释。所有其他属性都是可选项。

如果实施过程中支持相关属性，则应该随时为该属性提供取值。如果属性的取值未知，则在绝大多数情况下可以接受空值。如果在执行 GET 操作之后，相关属性没有返回，则说明当前的实施不支持该属性。

```
<Annotation Term="Redfish.Required"/>
```

Required 注释项的定义如下所述：

http://redfish.dmtf.org/schemas/v1/RedfishExtensions_v1.xml.

7.5.4.6 创建要求属性

RequiredOnCreate 注释项用于规定在创建资源时，要求对其做出规定的属性。没有使用 RequiredOnCreate 注释项注释的属性、或者使用取值为“false”的 Boolean 属性注释的属性，则在创建时没有要求。

```
<Annotation Term="Redfish.RequiredOnCreate"/>
```

RequiredOnCreat 注释项的定义如下所述：

http://redfish.dmtf.org/schemas/v1/RedfishExtensions_v1.xml.

7.5.4.7 检测单位

除了遵守命名公约之外，表示检测单位的属性应该使用 Unit 注释项进行注释，以便对属性的检测单位做出规定。

注释的取值应该为一个字符串，其中含有在“检测单位统一规范（UCUM）”中列出的区分大小写（c/s）的符号，除非相关符号表达并不是常用的表达形式（例如：通常使用 RPM（转每分）来表达风扇的运行速度，但此符号并不是 UCUM 的单位表达符号）。对于带有前缀的单位（例如：Mebibyte（1024^2 字节）其中带有 UCUM 前缀“Mi”和符号“By” ）而言，应该在单位符号中加入 UCUM 中列出的区分大小写的前缀符号。对于包括比例信息的数值（例如：兆位每秒）而言，应该加入比例单位符号，并插入一根斜杠“/”作为分隔符（例如：Mbit/s）。

```
<Annotation Term="Measures.Unit" String="MiBy"/>
```

Unit 注释项的定义如下所述：

http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Measures.V1.xml.

7.5.5 引用属性

引用其他资源的属性可以使用 NavigationProperty 要素，以引用属性的方式进行表达。在 NavigationProperty 要素中将规定相关资源的 Name 和限定类型命名空间。

如果属性引用的是单一的类型，则 type 属性的取值应该为相关资源类型的命名空间限定名称。

```
<NavigationProperty Name="RelatedType" Type="MyTypes.TypeB">
  <Annotation Term="OData.Description" String="This property references a
related resource."/>
  <Annotation Term="OData.LongDescription" String="This is the specification
of the related property."/>
  <Annotation Term="OData.AutoExpandReferences"/>
</NavigationProperty>
```

如果属性引用的是资源集合，则 type 属性的取值应该采用下列形式：

```
Collection(NamespaceQualifiedTypeName)
```

其中 *NamespaceQualifiedTypeName* 是指相关资源类型的命名空间限定名称。

```
<NavigationProperty Name="RelatedTypes" Type="Collection(MyTypes.TypeB)">
  <Annotation Term="OData.Description" String="This property represents a
collection of related resources."/>
  <Annotation Term="OData.LongDescription" String="This is the specification
of the related property."/>
  <Annotation Term="OData.AutoExpandReferences"/>
</NavigationProperty>
```

所有引用属性都必须包括描述和长描述注释项。

7.5.5.1 包含的资源

如果引用属性中的成员包含在引用资源之内，则应该使用取值为“true”的 ContainsTarget 属性进行规定。

例如，如果需要规定 Chasis 资源中含有 Power 资源，则你需要在 Chasis 类型定义中表达 Power 资源的资源属性中规定：ContainsTarget=true。

```
<NavigationProperty Name="Power" Type="Power.Power" ContainsTarget="true">
  <Annotation Term="OData.Description" String="A reference to the power
properties (power supplies, power policies, sensors) for this chassis."/>
  <Annotation Term="OData.LongDescription" String="The value of this property
shall be a reference to the resource that represents the power characteristics
of this chassis and shall be of type Power."/>
  <Annotation Term="OData.AutoExpandReferences"/>
</NavigationProperty>
```

7.5.5.2 扩展引用

对 Redfish JSON 负载的引用可以进一步扩展，纳入到相关资源的 id 或者相关资源集合的 id。此行为可以通过使用 AutoExpandReferences 注释项进行表达。

```
<Annotation Term="OData.AutoExpandReferences"/>
```

AutoExpandReferences 注释项的定义如下所述：
https://tools.oasis-open.org/version-control/browse/wsvn/odata/trunk/spec/vocabularies/Org.OData.Core.V1.xml.

7.5.5.3 扩展资源

此项可以用于引用属性，主要功能是规定服务器在响应过程中，对相关资源或者资源集合进行扩展的默认行为。

```
<Annotation Term="OData.AutoExpand"/>
```

AutoExpand 注释项的定义如下所述：
https://tools.oasis-open.org/version-control/browse/wsvn/odata/trunk/spec/vocabularies/Org.OData.Core.V1.xml.

7.5.6 资源动作

可以在一个名为“Actions”的属性下，对动作进行分类：

```
<Property Name="Actions" Type="MyType.Actions">
```

Actions 属性的类型为结构化类型，其中包括一个单一的 OEM 属性，其类型为结构化类型、没有规定的具体属性。

```
<ComplexType Name="Actions">
  <Property Name="OEM" Type="MyType.OEMActions"/>
</ComplexType>


<ComplexType Name="OEMActions"/>
```

每种动作都可以使用 Actions 要素，在命名空间内进行定义。动作的 Name 属性中规定了动作的名称，IsBound 属性规定相关动作是否与资源或者结构化类型绑定（即是否是其成员）。Action 要素中可以包括一个或多个 Parameter 要素，对每个参数的名称和类型做出规定。第一个参数称为“绑定参数”，主要用于规定相关动作是其中成员的资源或结构化类型（资源中动作属性的类型）。其余参数要素用于描述与动作相关的其他参数。

```
<Action Name="MyAction" IsBound="true">
  <Parameter Name="Thing" Type="MyType.Actions"/>
  <Parameter Name="Parameter1" Type="Edm.Boolean"/>
</Action>
```

7.5.7 资源可扩展性

公司、原始设备制造商、以及其他机构可以使用资源的 OEM 属性、Links 属性、以及动作属性，在通用 Redfish 资源中定义其他的属性、超链接、以及动作。

尽管这些扩展的信息和语义不在本标准范围之内，但用于编导数据的 schema、资源本身、以及协议相关的语义都必须遵守本规范文件的相关要求。

7.5.7.1 OEM 属性

在本条中，原始设备制造商（OEM）是指对 DMTF 公布的 schema、以及 Redfish 的功能性提供或者规定扩展的任何公司、生产商或机构。Redfish 自定义资源的基本 schema 包括一个空的复数型属性，名称为“OEM”，其取值可以用于纳入一项或多项 OEM 自定义复数型属性。因此，在标准 Redfish schema 中，OEM 属性是一个提前规定的占位符，可以用于规定 OEM 自定义属性。

正确使用 OEM 属性需要定义 OEM 自定义复数型属性的元数据，也可以在 OEM 属性内部进行引用。下列碎片是用于在复数类型“AnvilType1”下定义一组 OEM 自定义属性的 XML schema 示例（通常也可以使用其他 schema 要素，例如 XML 和 OData schema 描述识别符。在本示例中，为了便于说明，没有列出这些信息）。

```
<Schema Name="Contoso.v1_2_0">
  ...
  <ComplexType Name="AnvilType1">
    <Property Name="slogan" Type="Edm.String"/>
    <Property Name="disclaimer" Type="Edm.String"/>
  </ComplexType>


  ...
</Schema>
```

下面这个碎片显示的是：在对资源使用 GET 指令之后，之前的 schema 和“AnvilType1”属性类型是如何出现了 OEM 属性的例示过程中的。此示例中列出了在使用 OEM 属性时必须的两个要素：一个对象的名称、以及一个对象的类型属性。对这些要素的具体要求将在后文中进一步说明。

```
{}{
    "Oem": {
        "Contoso": {
            "@odata.type": "#Contoso.v1_2_0.AnvilType1",
            "slogan": "Contoso anvils never fail",
            "disclaimer": "* Most of the time"
        }
    },
    ...
}
```

7.5.7.2 OEM 属性形式和内容

OEM 属性中所包括的每个属性都应该是一个 JSON 对象。对象（属性）的名称通常用于识别对对象中所包含的属性进行定义的原始设备制造商或机构。这方面的内容将在后文中详细说明。OEM 自定义对象中应该纳入一个类型属性，主要目的是规定 schema 的具体位置、以及该 schema 内包括的属性的类型定义。OEM 属性中可以同时容纳多个 OEM 自定义对象，包括多家公司或多个机构的对象。

OEM 自定义复数类型中所包含的其他任何属性的定义，以及其功能规范、验证方式、以及其他内容要求都不在本标准文件的范围之内。尽管 Redfish 对于包含在 OEM 自定义 JSON 对象中的 OEM 自定义要素的尺寸或复杂度没有做出限制，但 OEM 属性通常只能用于规定少量的简单属性，主要用于 Redfish 资源的认证。如果需要支持大量对象或者大量数据（与 Redfish 资源的尺寸相比），则 OEM 应该考虑将 OEM 自定义对象指向一个用于扩展的单独资源。

7.5.7.3 OEM 属性命名

OEM 属性内的 OEM 自定义对象应该使用独特的 OEM 识别符进行命名，应该在命名空间的顶部命名，在其下方加上所定义的属性。识别符可以采用两种具体的形式。识别符要么采用可以被 ICANN 识别的域名（包括顶层域名后缀），之后所有点分隔符“.”都使用逗号“,”代替；要么采用 IANA 分配的企业编号，并在前面加上“EID”。已废除内容：识别符要么采用可以被 ICANN 识别的域名（包括顶层域名后缀）；要么采用 IANA 分配的企业编号，

并在前面加上“EID”。

使用“.com”域名的机构可以忽略“.com”后缀（例如：Contoso.com 可以使用“Contoso”作为属性名称，但“Contoso.org”必须使用“Contoso_org”作为其 OEM 属性名称）。OEM 识别符中的域名部分不应该区分大小写。也就是说“Contoso-biz”、“contoso_BIZ”、“conTOso_biZ”等等，都表示相同的原始设备制造商和顶层命名空间。

属性名称中的 OEM 识别符部分可以加上一根下划线，再接上任何其他字符串，以便根据原始设备制造商的要求设置命名空间。例如“Contoso_xxxx”或者“EID_412_xxxx”。在下划线后的文本的形式和内容完全由原始设备制造商自行确定。原始设备制造商自定义的文件类型后缀的扩展名可能需要根据具体情况确定。即通用客户端软件必须将上述扩展名（如果有）视为不透明，确保其不会影响或者解释文件内容。

有很多种使用此后缀名的方式，具体由原始设备制造商确定。例如，Contoso 公司可能有一个名为“Research”的下辖机构，则此 OEM 自定义属性名称可以扩展为“Contoso_Research”。另外还可用于识别职能部门、地理区域、子公司的命名空间。

名称中的 OEM 识别符部分通常用于识别创建和维持相关属性的 schema 的公司或机构。尽管如此，这并不是强制性要求。识别符只需要具有独特性，能够识别命名空间的顶层管理员，防止不同供货商或组织机构的 OEM 属性定义发生冲突即可。因此，命名空间顶层的机构可以与对 OEM 自定义属性做出定义的机构不相同。例如，Contoso 公司可以允许其客户，例如“客户 A”将特定的 Contoso 产品扩展为客户 A 的专有属性。在这种情况下，尽管 Consoto 公司为属性分配的名称为“Contoso_customers_CustomerA”，但客户 A 可以在该命名空间内对属性的内容和功能做出定义。在所有情况下，除非获得被识别的公司或者机构的许可，否则不得使用 OEM 识别符。

7.5.8 OEM 属性示例

下面的代码表示的是在访问资源时可能出现的 OEM 属性的命名和使用示例。此示例说明了 OEM 识别符可以使用不同的形式，OEM 自定义内容可以简单或者复杂，OEM 识别符的格式和扩展使用也是由 OEM 自定义的。

```
{}{
    "Oem": {
        "Contoso": {
            "@odata.type": "#Contoso.v1_2_1.AnvilTypes1",
            "slogan": "Contoso anvils never fail",
            "disclaimer": "* Most of the time"
        },
        "Contoso_biz": {
            "@odata.type": "#ContosoBiz.v1_1.RelatedSpeed",
            "speed" : "ludicrous"
        },
        "EID_412_ASB_123": {
            "@odata.type": "#OtherSchema.v1_0_1.powerInfoExt",
            "readingInfo": {
                "readingAccuracy": "5",
                "readingInterval": "20"
            }


        },
        "Contoso_customers_customerA": {
            "@odata.type" : "#ContosoCustomer.v2015.slingPower",
            "AvailableTargets" : [ "rabbit", "duck", "runner" ],
            "launchPowerOptions" : [ "low", "medium", "eliminate" ],
            "powerSetting" : "eliminate",
            "targetSetting" : "rabbit"
        }
    },
    ...
}
```

7.5.8.1 自定义动作

OEM 自定义动作可以通过定义绑定在资源动作属性类型中的 OEM 属性中的动作进行定义。

```
<Action Name="Ping" IsBound="true">
  <Parameter Name="ContosoType" Type="MyType.OEMActions"/>
</Action>
```

上述绑定动作将出现在 JSON 负载中，作为 OEM 类型属性，位于动作属性之下。

```
{}{
    "Actions": {
        "OEM": {
            "Contoso.vx_x_x#Contoso.Ping": {
                "target":"/redfish/v1/Systems/1/Actions/OEM/Contoso.Ping"
            }
        }
    },
    ...
}
```

7.5.8.2 自定义注释

本规范中将对一组常用的注释做出规定，以便 Redfish 能够利用其来对资源的定义进行扩展。除此之外，服务器也可以规定自定义注释。

服务器可以对资源进行注释，以便提供与服务相关的具体类型信息，例如相关服务是否支持对特定属性进行修改。

服务器可以对现有资源进行注释，如果这些资源尚没有明确规定的注释属性取值。对于作为资源定义部分的注释而言，服务器不能修改其取值。
由于服务器注释可以应用到现有资源定义中，因此这些注释通常在被服务元数据引用的服务器具体的元数据文件中做出规定。

7.6 Redfish 通用资源属性

在本条中，将规定一组适用于所有 Redfish 的通用属性。本条中规定的属性不能用于其他任何用途，即使其没有在任何具体资源中得到实施。
通用属性将在基础“资源”Redfish schema 中进行定义。对于 OData schema 表达而言，在 Resource_v1.xml 中定义，对于 JSON schema 表达而言，在 Resource.v1_0_0.json 中进行定义。

7.6.1 ID

资源的 ID 属性是一个独特的识别属性，用于在资源集合内识别资源。在整个资源集合范围内，ID 属性的取值必须具有独特性。

7.6.2 名称

Name（名称）属性用于为资源提供人类可读的名称。Name 属性的类型应该为字符串。在整个资源集合范围内，Name 属性的取值不要求具有独特性。

7.6.3 描述

Description（描述）属性用于为资源提供人类客户的描述内容。Name 属性的类型应该为字符串。

7.6.4 状态

Status（状态）属性用于表达资源的属性。
Status 属性的取值是本规范文件中规定的通用状态对象类型。通过为状态设置通用的表达形式，客户端可以使用相同的语义来进行调用。状态对象能够显示当前的预计状态、要求资源改变到的状态、当前的实际状态、以及可能会对资源当前状态造成影响的任何问题。

7.6.5 链接

Links（链接）属性表示的是与资源相关的超链接，具体将在该资源的 schema 定义中做出规定。该资源所有规定的相关引用属性都应该设置在 Links 属性之下。该资源所有规定的直接（从属性）引用属性都应该位于资源的根目录下。

7.6.6 成员

资源集合的 Members（成员）属性的主要功能用于识别集合中的成员。

7.6.7 相关项

RelatedItem（相关项）属性表示的是在资源 schema 定义中规定的资源（或者部分资源）的超链接。与其他引用方法不同，此方法不是一种很强的关联方法。相反，此属性的主要功能是显示服务中互相分离的部分中的要素和子要素之间的相关关系。例如，在实施过程中，可能风扇处于一个区域、而处理器处于另一个区域，此时即可以使用 RelatedItem 属性来向客户端说明两者之间的相关关系（在本例中，风扇用于冷却处理器）。

7.6.8 动作

Actions（动作）属性用于表示资源所支持的动作。

7.6.9 原始设备制造商

OEM（原始设备制造商）属性主要用于进行“schema 可扩展性”一条中所述的原始设备制造商扩展。

7.7 Redfish 资源

通常也统称为 Redfish schema，表示一组资源描述，其中包括有根据本规范要求实施过程中需要遵守的强制性要求。

Redfish 资源可以分为如下所述的各类：

- 服务根目录资源：
  - ✧ 包括有具体服务实例与对向相关资源的应用之间的映射关系；

- 包括有服务实例的 UUID，此 UUID 应该与通过 SSDP 发现指令返回的 UUID 相一致。

- 当前配置资源，其中包括：
  - 详细目录（静态和只读）；
  - 健康遥测数据（动态和只读）；
  - 当前配置设置（动态和读/写）；
  - 当前计量数值。
- 设置资源：
  - 动态、读/写待处理配置设置。
- 服务：
  - 诸如事件、任务、会话等通用服务。
- 寄存器资源：
  - 关于事件和信息寄存器的静态、只读 JSON 编码信息。

7.7.1 当前配置

当前配置资源表示的是服务器对资源的当前状态和配置的了解和信息。这些资源可以使用 PATCH 指令进行更新，也可以由客户端将其设置为只读，在这种情况下，客户端必须使用 PATCH 以及/或者 PUT 指令对单独的设置资源进行操作。对于可以直接修改的资源而言，在 GET 响应中的 Allow 数据头中，必须包含 PATCH 以及/或者 PUT 指令。对于只读资源而言，在 GET 响应中的 Allow 数据头中，不得包含 PATCH 以及/或者 PUT 指令。

7.7.2 设置

设置资源表示的是资源的未来状态和配置。对于支持未来状态和配置的资源而言，响应中应该保护一个带有“@Redfish.Settings”注释项的属性。尽管资源本身表达的是其当前状态，但设置资源表达的是资源的未来预计状态。

下面是一个支持设置资源的资源消息体示例。客户端可以使用“SettingsObject”属性来定位设置资源的 URI。

```
{}{
    "@Redfish.Settings": {
        "@odata.type": "#Settings.v1_0_0.Settings",
        "SettingsObject": {
            "@odata.id": "/redfish/v1/Managers/1/EthernetInterfaces/1/SD"
        },
        "Time": "2017-05-03T23:12:37-05:00",
        "ETag": "someetag",
        "Messages": []
    },
    ...
}
```

可以直接将设置资源的取值赋予资源，例如 POST 一个动作（例如：重置）或者通过 PUT/PATCH 请求；也可以进行间接赋值，例如用户重启 Redfish 服务外部的设备。在对设置资源进行配置时，客户端可以在请求消息体的“@Redfish.SettingsApplyTime”注释项中进行注释，说明在未来实施这些配置的具体时间。如果在实施这些未来设置时，服务支持这些配置，则表达设置资源的响应消息体中应该包括一个带有“@Redfish.SettingsApplyTime”注释项的属性。更多详细信息，请参阅 Redfish schema“设置”一节中定义的属性。
下面是一个请求消息体示例，其中说明了客户端如何配置向设置资源赋值的时间：

```
{}{
    "@Redfish.SettingsApplyTime": {
        "ApplyTime": "OnReset",
        "MaintenanceWindowStartTime": "2017-05-03T23:12:37-05:00",
        "MaintenanceWindowDurationInSeconds": 600
    },
    ...
}
```

7.7.3 服务

服务资源表示的是 Redfish 服务本身的组件以及独立资源。完整的清单只有通过浏览 Redfish 服务树才能发现，上述清单中包括了诸如事件服务、任务管理、以及会话管理等服务。

7.7.4 寄存器

寄存器资源是指用于协助客户端解释在 Redfish schema 定义范围之外的 Redfish 资源的资源。寄存器资源的实例包括消息寄存器、时间寄存器、以及枚举寄存器，例如在 BIOS 中使用的各种寄存器。在寄存器中，可以通过一个识别符来检索关于特定资源、事件、消息或者其他项目的更多信息。其中可以包括其他属性、属性限制、以及类似信息。寄存器本身就是一种资源。

7.8 特殊资源情况

在特定类型的资源需要使用通用语义进行表达时，可能会出现一些特殊情况。

7.8.1 不存在的资源

在客户端请求获得相关资源的信息时，相关资源可能不存在、或者其状态为止。对于已经删除。但预计其 URI 将保持不变的资源而言（例如：在风扇被拆除的情况下），资源应该将状态对象的状态属性表达为“不存在（Absent）”。在这种情况下，任何没有已知取值的要求的、或者支持的属性都应该取值为空值。

7.8.2 schema 变更

在有的情况下，需要对公布的 Redfish schema 进行必要的修改。其中一个具体示例就是 BIOS，不同的服务器可以对可用的配置设置进行少量的修改和变更。提供商可以构建一个单独的脚骨，此 schema 应该为各种实施情况的超集。为了支持这些变更，Redfish 支持忽略当前配置对象中的分级 schema 中定义的参数。具体规则如下：

- 所有 Redfish 服务都必须支持在设定数据中设定不支持配置要素的尝试，具体做法为在设置数据应用状态结构中将其标记为例外情况，但必须确保不会导致整套配置无法运行；
- 通过在当前配置对象中纳入相关属性，表示在资源中支持该属性。如果当前配置中有要素缺失，则客户端必须假设认为该要素不被资源支持。
- 对于允许取值可能会发生变动的 ENUM 配置项目而言，可以在当前配置中加入一个特殊的只读能力，对要素的限值做出规定。这种设置将会取代 schema，因此只能在必要的情况下使用。

提供商还可以将 schema 资源分为单独的文件，例如 schema+字符串寄存器，每个文件都具有单独的 URI 和不同的 Content-Encoding。

- 在适用的情况下，资源可以通过当前配置对象，将当前公布的 schema 中的疏漏告知客户端。

8. 服务细节

8.1 事件

在本条中，将对订阅和接收事件消息的 REST 基机制做出规定。

Redfish 服务要求客户端或者管理员创建接收事件的订阅。在管理员向订阅资源的 URI 发送 HTTP POST 指令之后，即可以创建订阅。此请求中应该包括事件接收者客户端希望事件被发送到的目的地 URI，以及需要发送的事件的类型。之后，在服务中触发了事件之后，Redfish 服务将会把事件发送到上述 URI 处。

- 对于所有能够发送事件的资源而言，服务器都必须能够支持“推送式”事件；
- 除非已经创建了事件订阅，否则服务器不得支持“推送式”事件（使用 HTTP POST 指令）。客户端或者服务器都可以在任何时间，通过删除订阅来中止事件流。如果传输错误的次数超过了预先规定的阈值，则服务器也可以删除订阅；
- 在成功订阅之后，服务器应该使用 HTTP 状态代码 201 进行响应，同时将 HTTP 中的 Location 数据头设定为新订阅资源的地址。订阅具有持续性，在相关事件服务器重启之后依然有效；
- 客户端可以通过向订阅资源的 URI 发送 HTTP DELETE 指令来中止订阅；
- 服务器可以在最后一条消息中发送特殊的“订阅中止”消息，进而中止订阅。之后，未来对相关订阅资源的请求将会以 HTTP 状态代码 404 进行响应。

Redfish 服务会生成两种类型的事件，分别是寿命周期事件和报警事件。

寿命周期事件在资源被创建、修改或删除时发生。并不是对资源进行的所有修改都会触发事件，这有点类似于在 ETag 被修改时，实施设备也不会针对每次资源变更发送事件。例如，如果在接收到每一个以太网数据包之后、或者在传感器每改变 1 度后就发送事件，则可能导致事件的数量过多，超过可扩展界面的承受能力。此类型事件通常用于表示资源发生了变化，同时也可以选择用于表示资源属性发生了变化。

在资源需要说明某些具有重要性的事件时，将会触发报警事件。报警事件可以直接或间接与资源相关。这种类型的事件通常采用消息寄存器方法，类似于扩展错误处理，其中必须包括一个 MessageId。此类型事件的示例包括机壳被打开、按钮被按下、电缆没有插好、或者阈值被超过的情况。这些事件通常与寿命周期事件无关，因此具有自己独特的分类。

备注：事件的安全性方面的信息，请参阅“安全性”一条相关内容。

8.1.1 事件消息订阅

客户端可以通过浏览 Redfish 服务器界面来定位事件服务。在找到事件服务之后，客户端可

以向相关资源集合的 URL 发送 HTTP POST 指令，“订阅”相关事件服务。事件服务可以通过服务器根目录进行查找，具体方法将在 Redfish schema 文件中进行说明。

订阅请求消息体中的具体语法，将在 Redfish schema 文件“EventDestination”一节中进行说明。

在订阅成功之后，事件服务将会返回 HTTP 状态代码 201（已创建），同时在响应消息的数据头中将会包括一个 URI，说明新创建订阅资源的具体位置。响应消息的消息体（如果有）中，必须按照“EventDestination”schema 要求，包含一个订阅资源的表达。向订阅资源发送 HTTP GET 指令将会返回订阅的配置信息。

在相关订阅在服务器内创建完成之后，客户端即可以开始接受事件，但不能接收订阅之前已经触发的事件。服务器不会保留历史事件。

8.1.2 事件消息对象

POST 到具体客户终端处的事件消息中，必须包括 Redfish 事件 schema 中所规定的属性。

事件消息的结构必须支持消息寄存器。在消息寄存器方法中，需要使用已知的规定格式列出一个 MessageId 清单或者数组。这些 MessageId 具有简洁的性质，因此尺寸比实际的消息要小，因此使其非常适合于嵌入式环境。在寄存器中也保存有消息。消息本身可以带有提要，同时其 Severity（严重程度）和 Recommended Action（建议动作）属性将具有默认值。

MessageId 属性内容必须采用下列形式：

RegistryName.MajorVersion.MinorVersion.MessageKey

其中：

- RegistryName 是指寄存器的名称，寄存器的名称必须使用 Pascal-case 命名；
- MajorVersion 是指代表寄存器主要版本的正整数；
- MinorVersion 是指代表寄存器次要版本的正整数；
- MessageKey 是寄存器内人类可读的密钥，消息密钥应该使用 Pascal-case 编码，其中不得包含空格、句号、或者特殊符号。

8.1.3 订阅删除

如果需要取消对相关资源的订阅，则客户端或者管理员只需要向订阅资源的 URI 发送 HTTP DELETE 指令即可。

在全局设置中有一些可配置的属性，用于规定所有事件订阅的具体行为。更多关于可用于配置服务器订阅行为的参数的信息，请参阅 Redfish schema“EventService”一节中规定的属性。

8.2 非同步操作

支持非同步操作的服务器可以实施任务（Task）服务和任务（Task）资源。

任务服务是用于描述处理任务的服务。其中含有一个包括零个或以上“任务（Task）”资源的资源集合。任务资源用于描述在提出请求之后，需要等待数秒或以上时间才能执行完毕的长时间运行任务。客户端可以查询任务资源的 URI，以确定相关操作将在何时完成、以及是否成功完成。

Redfish schema 中的任务（Task）结构中，含有任务的具体结构。其中所包含的信息包括：开始时间、结束时间、任务方式、任务状态、以及零条或以上的与任务相关的消息。

每个任务都有多种可能的实现方式，具体实现方式及其语义将在 Redfish schema 的任务资源中做出规定。

在客户端提出长时间运行的任务请求时，服务器将会返回 202 状态代码（已接受）。

在任何 202 状态代码（已接受）响应中，都必须包括一个 Location 数据头，在其中说明任务监控器（Task Monitor）的 URL，同时还可以包括一个 Retry-After 数据头，在其中说明客户端在可以查询任务状态之前需要等待的时间。

任务监控器（Task Monitor）是一个由服务器生成的不透明 URL，供提出相关请求的客户端使用。客户端可以对任务监控器执行 GET 请求，查询相关操作的状态。

在对任务监控器执行 GET 请求时，客户端不应该在 Accept 数据头中加入 MIME 类型的 application/http。

在 202 状态代码（已接受）响应中，应该包括一个任务资源实例，对任务方式进行描述。

在操作处理过程中，如果客户端查询在 Location 数据头中返回的任务监控器（Task Monitor），则服务器应该持续返回 202 状态代码（已接受）。

客户端可以通过对任务监控器（Task Monitor）URL 执行 DELETE 操作，取消相关任务的运行。具体何时删除相关的任务资源对象将由服务器决定。

客户端还可以通过对 Task（任务）资源执行 DELETE 操作，取消任务运行。删除 Task（任务）资源目标可能会导致相关的 Task Monitor 失效，进而导致在后续对 Task Monitor URL 执行 GET 操作时，返回 410（已删除）或者 404（未发现）错误代码。

在操作完成之后，任务监控器（Task Monitor）应该返回适宜的状态代码（绝大多数操作为正常（200），创建资源的操作位已创建（201）），同时必须包括数据头和相关操作的响应消息体，就像相关操作是同步完成的一样。如果初始操作过程中发生错误，则响应消息体中应该包括错误响应。

如果操作已经完成，且服务器已经删除了相关任务，则服务器可以返回 410（消失）或者 404（未发现）状态代码。在客户端等待了过长时间才读取任务监控器（Task Monitor）时可能会出现这种情况。

客户端可以使用在 202 状态代码（已接受）响应的消息体中返回的资源识别符，直接查询任务资源，以便获得相关信息。

- 支持非同步操作的服务器必须实施任务资源；
- 对非同步操作的响应中应该返回 202 状态代码（已接受），同时将 HTTP 响应中的“Location”数据头设置为与活动相关的任务监控器（Task Monitor）的 URI。响应中可以包括一个 Retry-After 数据头，在其中说明客户端在可以查询任务状态之前需要等待的时间。响应消息体中应该包括一个 JSON 格式的任务资源表达；
- 向任务监控器（Task Monitor）或任务资源提出 GET 请求时，应该立即返回相关操作的当前状态；
- 使用 HTTP GET、PUT、PATCH 指令发起的操作应该随时都是同步操作；
- 客户端必须准备好，在提起 HTTP PUT、PATCH、POST 和 DELETE 请求之后，能够同时处理返回的同步和非同步响应。

8.3 资源树稳定性

资源树（Resource Tree）是指在实施设备内部规定的一组 URI 和数组元素。在同一个服务中，在设备重启和交流电源循环的过程中，上述资源树必须能够保持恒定，同时必须能够承受一定范围的合理配置变更（例如：为服务器添加适配器）。在同一个服务中，不同设备实例之间的资源树可以不一致。客户端必须浏览数据模型，发现资源并与其互动。在不同系统中，部分资源可能能够非常稳定（例如：BMC 网络设置），但从 schema 的层面来看，这种稳定性没有保证。

- 在服务重启和进行微小的配置变更时，资源树应该保持稳定，因此该组 URI 和数组元素指数应该保持恒定；
- 客户端不能假设资源树能够在不同的服务实例间保持一致。

8.4 发现

可以使用简单服务发现协议（SSDP）来自动发现支持 Redfish 可扩展平台管理功能 API 的受管理设备。此协议允许通过网络进行高效的查找和发现，而不需要凭借 ping 扫描、路由器表单搜索、或者限制 DNS 命名机制等手段。使用 SSDP 是一种可选方案，在实施之后，必须允许用户通过“管理员网络服务（Manager Network Service）”资源禁用该协议。

对于客户端软件而言，发现的目的在于定位满足 Redfish 要求的受管理设备，因此包含的基本 SSDP 功能即为 M-SEARCH 查询功能。在必要的情况下，Redfish 还将遵守 SSDP 扩展和 UPnP 使用的命名规则，确保满足 Redfish 要求的系统可以在不冲突的情况下实施 UPnP。

8.4.1 UPnP 兼容性

为了保证通用型 SSDP 客户端软件（主要是 UPnP）的兼容性，所有 SSDP 流量都应该使用 UDP 1900 端口。除此之外，SSDP 多路广播消息的生存时间（TTL）跳次计数应该设置为默认值 2。

8.4.2 USN 格式

在服务的 USN 栏中提供的 UUID 应该等于服务器根目录的 UUDI 属性。如果有多个/冗余管理员，则不论是否执行的冗余故障转移操作，服务的 UUID 都应该保持不变。UUID 应该使用典型的 UUID 格式，后面接上“::dmtf-org”。

8.4.3 M-SEARCH 响应

Redfish 服务搜索目标（ST）的定义为：urn:dmtf-org:service:redfish-rest:1。
受管理装置应该对针对 Redfish 服务搜索目标（ST）的 M-SEARCH 查询搜索和“ssdp:all”做出响应。对于 UPnP 兼容性而言，受管理装置应该对搜索目标的“upnp:rootdevice”的 M-SEARCH 查询搜索做出响应。
回复中的 ST 数据头中所提供的 URN 应该使用“redfish-rest”服务名称，之后接上 Redfish 规范的主要版本号。如果服务所遵守的 Redfish 规范的次要版本号是非零值、且该次要版本与之前的次要版本之间具有向后兼容性，则名称中应该加上次要版本号，并在其前面加上一个冒号。例如，遵守“1.4”版 Redfish 规范的服务应该使用“redfish-rest:1:4”进行回应。
受管理装置应该向客户端提供 AL 数据头，在其中指向 Redfish 服务器根目录的 URL。
对于 UPnP 兼容性而言，受管理装置应该向客户端提供 LOCATION 数据头，在其中指向 UPnP XML 描述文件。
对 M-SEARCH 多通道或单通道查询的响应应该采用如下所述的形式。括号内的空格是装置具体数值的占位符：

```
HTTP/1.1 200 OK
CACHE-CONTROL:max-age=<seconds, at least 1800>
ST:urn:dmtf-org:service:redfish-rest:1
USN:uuid:<UUID of Manager>::urn:dmtf-org:service:redfish-rest:1
AL:<URL of Redfish service root>
EXT:
```

8.4.4 通知、激活和关闭消息

Redfish 服务还可以实施其他额外的通过 UPnP 定义的 SSDP 消息，以便宣告其软件的可用性。此功能在实施之后，必须允许最终用户单独禁用 M-SEARCH 响应功能的流量。通过这种方式可以允许用户在使用最少网络流量的情况下，利用发现功能。

9. 安全性

9.1 协议

9.1.1 TLS

在实施过程中必须支持 TLS V1.1 或更新版本。
在实施过程中应该支持最新版本的 TLS v1.x 规范。
在实施过程中应该支持“存储系统 SNIA TLS 规范”。

9.1.2 密码组

在实施过程中必须遵守通过 TLS 协议获得的 AES-256 密码组。
Raedfish 实施设备应该考虑支持与如下所述的密码组类似的密码组，以便能够在不使用受信任证书的情况下进行认证和识别：

```
      TLS_PSK_WITH_AES_256_GCM_SHA384
  TLS_DHE_PSK_WITH_AES_256_GCM_SHA384
  TLS_RSA_PSK_WITH_AES_256_GCM_SHA384
```

使用上述建议密码组的其他优点还包括：
“AES-GM 不仅有效、安全，而且可以在较低成本和较低延迟的情况下，让硬件实施达到较快速度，因为这种模式可以流水线作业。”
Redfish 实施设备还应该支持下列额外密码组：

```
        TLS_RSA_WITH_AES_128_CBC_SHA
```

对 RFC 的引用：

```
  http://tools.ietf.org/html/rfc5487
  http://tools.ietf.org/html/rfc5288
```

9.1.3 证书

在提供了替换证书的情况下，Redfish 实施设备必须能够支持对默认证书进行替换。
Redfish 实施设备必须能够支持使用遵守 X.509 v3 证书格式的证书，具体定义参阅 RFC 5280 相关内容。

9.2 认证

- 认证方法：

  服务应该同时支持“基本认证”和“Redfish 会话登录认证”（具体定义将在后文“会话管理”一节中进行介绍）。在使用基本认证的情况下，服务器不得要求客户端发起会话。服务器可以实施其他的认证机制。

9.2.1 HTTP 数据头安全性

- 所有发送给 Redfish 对象的书面请求都必须视为已认证状态，即 POST、PUT/PATCH、以及 DELETE，例外情况包括：
  - ✧ 发送给会话服务/对象的 POST 操作需要进行认证：
    - ■ 在认证失败时，在扩展错误消息中不得包含任何保密信息。
- 除了下列情况之外，在未通过认证之前，不得向客户端提供 Redfish 对象：
  - ✧ 在识别设备和服务位置时所需要的根目录对象；
  - ✧ 在检索资源类型时所需要的$metadata 对象；
  - ✧ 在确认 OData 客户端兼容性时所需要的 OData 服务文件；
  - ✧ 位于/redfish 位置的版本对象。
- 通过外部引用链接的外部服务不在本规范范围之内，这些服务可以遵守其他不同的安全性要求。

9.2.1.1 HTTP 重新定向

- 在出现 HTTP 重新定向时，必须强制执行目标资源的特权要求；
- 在通常情况下，如果相关位置可以在不需要认证的条件下访问，但只能以 https 的形式访问，则服务器应该执行重新定向，将其重新定向到资源的 https 版本。如果相关资源只有在认证之后才能访问，则应该返回 404 状态代码。

9.2.2 扩展错误处理

- 在认证失败时，在扩展错误消息中不得包含任何保密信息。

9.2.3 HTTP 数据头认证

- 用于认证的 HTTP 数据头必须在可能影响响应的其他数据头之前进行处理，例如：etag、If-Modified 等；
- 绝对不能使用 HTTP Cookies 来认证任何请求，例如：GET、POST、PUT/PATCH、以及 DELETE。

9.2.3.1 基本认证

必须能够支持 RFC 7235 中规定的 HTTP 基本认证，只能使用合格的 TLS 连接在任何第三方认证服务和客户端之间进行数据传输。

9.2.3.2 请求/消息级别认证

应该为每个请求建立一条安全通道，同时伴随有一个认证数据头。

9.2.4 会话管理

9.2.4.1 会话寿命周期管理

会话管理留待 Redfish 服务实施。其中包括会话超时和同时发起的会话数量。

- **Redfish 服务必须根据本规范要求提供登录会话。**

9.2.4.2 Redfish 登录会话

对于需要执行多次 Redfish 操作的功能而言、或者出于安全性方面的原因，客户端可以通过会话管理界面创建一个 Redfish 登录会话。用于会话管理的 URI 将在会话（Session）服务中做出规定。建立会话的 URI 可以在会话服务的 Session（会话）属性中找到，或者在服务根目录中，Session 属性之下的 Link 属性中找到。

```
{}{
    "SessionService": {
        "@odata.id": "/redfish/v1/SessionService"
    },
    "Links": {
        "Sessions": {
            "@odata.id": "/redfish/v1/SessionService/Sessions"
        }
    },
    ...
}
```

9.2.4.3 会话登录

在不需要认证数据头的情况下，通过向会话服务的会话（Sessions）资源集合发送 HTTP POST 指令即可以创建 Redfish 会话，其中应该包括下列 POST 消息体：

```
POST /redfish/v1/SessionService/Sessions HTTP/1.1
Host: <host-path>
Content-Type: application/json;charset=utf-8
Content-Length: <computed-length>
Accept: application/json;charset=utf-8
OData-Version: 4.0

{}{
    "UserName": "<username>",
    "Password": "<password>"
}
```

初始的数据头应该保存在本次会话创建的引用资源中，并与利用此会话与后续的请求进行比较，以便验证相关请求是否是由具有授权的客户端提出的。

在对创建会话的 POST 请求的响应中，应该包括下列信息：

- 一个 X-Auth-Token 数据头，其中带有客户端在后续提出请求时可以使用的“会话授权令牌”；
- 一个 Location 数据头，其中带有新创建的会话资源的超链接；
- JSON 响应消息体，其中带有新创建的会话对象的完整表达（参阅下文中的示例）。

```
Location: /redfish/v1/SessionService/Sessions/1
X-Auth-Token: <session-auth-token>

{}{
    "@odata.context": "/redfish/v1/$metadata#Session.Session",
    "@odata.id": "/redfish/v1/SessionService/Sessions/1",
    "@odata.type": "#Session.v1_0_0.Session",
    "Id": "1",
    "Name": "User Session",
    "Description": "User Session",
    "UserName": "<username>"
}
```

发送会话登录请求的客户端应该保存“会话授权令牌”和 Location 数据头中返回的超链接。其中“会话授权令牌”用于后续请求的认证，具体方法为将请求的“X-Auth-Token”数据头设置为从登录 POST 处接收到的“会话授权令牌”。在之后，客户端可以使用 POST Location 数据头中返回的超链接来注销登录或者中止会话。

需要注意的一点是，“会话 ID”和“会话授权令牌”是不同的。会话 ID 具有独特性，用于识别会话资源，会在响应数据、以及 Location 数据头返回的超链接中的最后一段中返回。具有充分权限的管理员可以使用相关的会话 ID 来查看和中止激活的会话。只有成功登录的客户端才会获得会话授权令牌。

9.2.4.4 X-Auth-Token HTTP 数据头

在实施过程中，只能使用合格的 TLS 连接在任何第三方认证服务和客户端之间进行数据传输。因此，用于创建新会话的 POST 指令只能使用 HTTPS 进行支持，基本认证中使用的所有请求都需要采用 HTTPS 形式。

9.2.4.5 会话寿命周期

需要注意的一点是，Redfish 会话“超时”机制与某些具有令牌到期时间的令牌基认证方法不同。对于 Redfish 会话而言，只要客户端在会话过程中发送请求的时间间隔短于会话的超时时间，则会话将会持续进行，令牌也会一直生效。如果会话超时，则会话将会自动中止。

9.2.4.6 会话中止或注销

在客户端注销之后，Redfish 会话将被中止。在会话中止时，会执行一个 DELETE 指令，删除在创建会话时响应中的 Location 数据头中返回的超链接、或者删除响应数据中返回的会话 ID。

通过规定会话资源 ID 来删除会话的能力，确保管理员具有通过不同的会话、中止其他用户会话的充分权限。

9.2.5 账号服务

- 应该使用单向加密技术来储存用户密码；
- 在实施过程中，可以支持将带有密码的用户账号导出。但在导出的过程中，必须使用加密方法来对其进行保护；
- 用户账号必须支持 ETag，同时必须支持原子操作：
  - 在实施过程中，可以拒绝未带有 ETag 的请求。
- 用户管理活动必须为原子操作；
- 在认证失败时，在扩展错误消息中不得包含任何特权信息。

9.2.6 非同步任务

- 不论使用了哪种用户/特权环境来启动非同步任务，都必须使用状态对象中的信息来强制执行访问对象所需要的权限认证。

9.2.7 事件订阅

- 在向目的地推送事件数据对象之前，Redfish 装置可以对目的地进行验证，以便识别确定事件订阅的具体目的。

9.2.8 特权模式/授权

授权子系统可以使用角色和特权来控制有权限访问资源的用户。

- 角色：
  - ✧ 角色是一组规定的特权，因此，两个具有相同特权的角色应该具有相同的行为模式；
  - ✧ 所有用户都将被分配一个角色；
  - ✧ 在本规范文件中将规定一组预先规定的角色，在用户创建时，应该向用户分配其中一个角色；
  - ✧ 预定角色的创建流程如下所述（其中“角色名称”是指角色（role）资源中 ID 属性的取值）：
    - ■ 角色名称为“管理员（Administrator）”：
      - ■ 分配的特权：登录、配置管理员、配置用户、配置部件、配置自身。
    - ■ 角色名称为“操作员（Operator）”：
      - ■ 分配的特权：登录、配置部件、配置自身。
    - ■ 角色名称为“只读（ReadOnly）”：
      - ■ 分配的特权：登录、配置自身。
  - ✧ 在实施过程中，必须能够支持所有上述预定角色；
  - ✧ 预定角色中可以包括原始设备制造商自定义的特权；
  - ✧ 针对预定角色规定的特权数组不可以修改；
  - ✧ 服务还可以选择规定其他额外的“自定义”角色；
  - ✧ 服务可以允许用户通过向“角色（Role）”资源集合发送 POST 指令，创建自定义角色。
- 特权：
  - ✧ 特权是在规定的管理域内（例如：用户配置），允许执行某项操作的权限（例如：读出、写入）；
  - ✧ 在 Redfish 规范中，在角色（Role）资源的 AssignedPrivileges 数组中规定了一组“已分配特权”；
  - ✧ 在实施过程中，可以对“原始设备制造商特权做出规定”，具体做法为在角色（Role）资源的 OemPrivileges 数组中进行定义；
  - ✧ 使用 Redfish schema“特权”文件中的特权映射（privilege mapping）注释项可以将特权映射到资源之上；
  - ✧ 多个特权映射构成特权 OR。
- 用户管理：
  - ✧ 在创建用户账号时，应该为用户分配一个角色；
  - ✧ 该用户享有的特权将通过其角色进行定义。
- ETag 处理：

- ✧ 在实施过程中，对于 ETag 相关的工作而言，必须强制执行与 ETag 所代表的数据相同的特权模型；
- ✧ 例如，如果某项工作在读取 ETag 所代表的数据时需要访问特权，则在读取 ETag 时也必须需要同样的访问特权。

9.2.9 Redfish 服务操作与特权的映射

对于 Redfish 客户端向 Redfish 服务器提出的每次请求而言，Redfish 服务器都应该确定提出请求的客户端的认证身份，以确定请求者是否具有对请求中所述的资源执行相关操作的权限。通过使用角色和特权授权模型，可以将身份认证的语义分配给角色、将角色定义为一组特权，这样服务器就能够检查 HTTP 请求与经过认证的请求角色/特权之间的映射关系，从而确定相关用户是否具有直形请求中所述的操作的充分权限。

9.2.9.1 将操作映射到特权上的原因

在初始版本的 Redfish 规范中，针对标准化的角色规定了多组角色与特权的映射关系，并强制性地规定了多个特权标签，但并没有强制性规定这些特权的详细含义、也没有强制性规定特权与操作之间的映射关系。由于缺乏确定针对请求中所述的 URI 执行请求中所述的操作所需要的具体特权的方法，这就导致了 Redfish 服务实施的可互通性存在风险，因为客户端在不同的实施过程中，可能会遇到不同的特权要求。另外，缺乏规定和表达操作与特权的映射关系的方法，也导致 SPMF 或者其他机构难以强制性地规定服务的特权要求。

9.2.9.2 操作与特权映射关系表达

Redfish 服务器应该在服务寄存器集合中提供一个特权寄存器（Privilege Registry）文件。特权寄存器文件代表的是针对向服务器提出的 HTTP 请求中规定的 URI 执行某项操作所需要的特权。特权寄存器文件是一个单独的 JSON 文件，其中含有 PrivilegeMapping（特权映射）的实体要素，其中针对服务器支持的每种 schema 实体提供一个具体要素。操作与特权之间的映射关系通过实体 schema 进行定义，适用于在适用 schema 下实施的服务中的所有资源。另外，存在一些特殊情况，在其中特定资源或资源要素可能具有与实体映射中规定的不同的特权映射关系。在这些情况下，实体层面的影射关系将会被取代。规定实体层面操作与特权映射关系和相关取代关系的方法将在“特权寄存器（Privilege Registry）”schema 中进行规定。如果 Redfish 服务器提供了一个特权寄存器（Privilege Registry）文件，则对于服务器支持的操作而言，服务器应该使用 SPMF Redfish 特权映射寄存器定义作为基本的操作与特权映射定义，以便促进 Redfish 客户端之间的可互通性。

9.2.9.3 操作映射语法

操作映射定义了一组在对实体、实体要素、或者资源执行特定操作时需要的特权。映射的操作包括 GET、PUT、PATCH、POST、DELETE 和 HEAD。不论相关服务器或者 API 数据模

型是否支持对实体、实体要素、或者资源执行的特定操作，都应该针对每种操作建立特权映射。使用的特权标签可以为在 Privilege. PrivilegeTyep 枚举中定义的标准 Redfish 标签，也可以为原始设备制造商自定义的特权标签。执行某项操作需要的特权可以使用逻辑 AND 和 OR 语法进行规定（更多信息参阅“特权 AND 和 OR 语法”一节相关内容）。在下面的示例中，规定了对 Manager 实体执行各项操作所需要的特权。除非映射关系被定义的 OperationMap 数组取代（具体语法将在下一节中进行介绍），否则在实施过程中，规定的操作与特权映射关系将适用于所有 Manager 资源。

```
{}{
    "Entity": "Manager",
    "OperationMap": {
        "GET": [
            {}{
                "Privilege": [ "Login" ]
            }
        ],
        "HEAD": [
            {}{
                "Privilege": [ "Login" ]
            }
        ],
        "PATCH": [
            {}{
                "Privilege": [ "ConfigureManager" ]
            }
        ],
        "POST": [
            {}{
                "Privilege": [ "ConfigureManager" ]
            }
        ],
        "PUT": [
            {}{
                "Privilege": [ "ConfigureManager" ]
            }
        ],
        "DELETE": [
            {}{
                "Privilege": [ "ConfigureManager" ]
            }


    ]
  }
}
```

9.2.9.4 映射取代语法

可能会出现多种情况，导致操作与特权的映射关系与实体 schema 层面规定的映射关系有所不同。这些情况包括：

- 属性取代：是指资源（文件）中的属性具有不同的特权要求。例如，在 ManagerAccount 资源中的 Password（密码）属性中要求“配置自身”或者“配置用户”特权才能进行修改，但 ManagerAccount 资源中的其余属性中只要求“配置用户”特权即可以进行修改；
- 从属取代：是指某个实体在另一个实体的语境中被使用，同时根据语境，相关特权

会受到管辖。例如，对 EhternetInterface 资源的 PATCH 操作所需要的特权取决于资源从属于 Manager 资源（需要“配置管理员”特权）还是 ComputerSystem 资源（需要“配置部件”特权）；

- 资源 URI 取代：是指某个具体的资源实例具有与实体 schema 规定不同的操作特权要求。这种取代应该在实体的操作与特权映射关系的语境中进行定义。

9.2.9.5 属性取代示例

在下面这个示例中，在 ManagerAccount 资源中的 Password（密码）属性中要求“配置自身”或者“配置用户”特权才能进行修改，但 ManagerAccount 资源中的其余属性中只要求“配置用户”特权即可以进行修改。

```
{}{
    "Entity": "ManagerAccount",
    "OperationMap": {
        "GET": [
            {}{
                "Privilege": [ "ConfigureManager" ]
            },
            {}{
                "Privilege": [ "ConfigureUsers" ]
            },
            {}{
                "Privilege": [ "ConfigureSelf" ]
            }
        ],
        "HEAD": [
            {}{
                "Privilege": [ "Login" ]
            }
        ],
        "PATCH": [
            {}{
                "Privilege": [ "ConfigureUsers" ]
            }
```

```
            ],
            "POST": [
                {}{
                    "Privilege": [ "ConfigureUsers" ]
                }
            ],
            "PUT": [
                {}{
                    "Privilege": [ "ConfigureUsers" ]
                }
            ],
            "DELETE": [
                {}{
                    "Privilege": [ "ConfigureUsers" ]
                }
            ]
        },
        "PropertyOverrides": [
            {}{
                "Targets": [ "Password" ],
                "OperationMap": {
                    "GET": [
                        {}{
                            "Privilege": [ "ConfigureManager" ]
                        }
                    ],
                    "PATCH": [
                        {}{
                            "Privilege": [ "ConfigureManager" ]
                        },
                        {}{
                            "Privilege": [ "ConfigureSelf" ]
                        }
                    ]
                }
            }
        ]
    }
```

9.2.9.6 从属取代

在从属取代的 Targets（目标）属性中，需要列出一个层级关系表达，说明何时需要执行此种取代。在下面这个示例中，在其从属于 EthernetInterfaceCollection 实体（其本身从属于 Manager 实体）时，需要执行 EthernetInterface 的从属取代。如果客户端对 EthernetInterface 提出的 PATCH 请求满足此取代条件，则需要提供“配置管理员特权”；否则的话，客户端需要提供“配置部件”特权。

```
{}{
    "Entity": "EthernetInterface",
    "OperationMap": {
        "GET": [
            {}{
                "Privilege": [ "Login" ]
```

```
            }
        ],
        "HEAD": [
            {{
                "Privilege": [ "Login" ]
            }
        ],
        "PATCH": [
            {{
                "Privilege": [ "ConfigureComponent" ]
            }
        ],
        "POST": [
            {{
                "Privilege": [ "ConfigureComponent" ]
            }
        ],
        "PUT": [
            {{
                "Privilege": [ "ConfigureComponent" ]
            }
        ],
        "DELETE": [
            {{
                "Privilege": [ "ConfigureComponent" ]
            }
        ]
    },
    "SubordinateOverrides": [
        {{
            "Targets": [
                "Manager",
                "EthernetInterfaceCollection"
            ],
            "OperationMap": {
                "GET": [
                    {{
                        "Privilege": [ "Login" ]
                    }
                ],
                "PATCH": [
                    {{
                        "Privilege": [ "ConfigureManager" ]
                    }
                ]
            }
        }
    ]
}
```

9.2.9.7 资源 URI 取代

在下面这个示例中使用了资源 URI 取代语法，用于说明具体资源 URI 的操作特权变化。在此示例中，在对作为目标列出的 2 个资源 URI 执行 PATCH 操作时，需要同时具备“配置部件”特权和“OEM 管理员”特权。

```
{{
    "Entity": "ComputerSystem",
    "OperationMap": {
        "GET": [
            {{
                "Privilege": [ "Login" ]
            }
        ],
        "HEAD": [
            { {
                "Privilege": [ "Login" ]
            }
        ],
        "PATCH": [
            {{
                "Privilege": [ "ConfigureComponent" ]
            }
        ],
        "POST": [
            {{
                "Privilege": [ "ConfigureComponent" ]
            }
        ],
        "PUT": [
            {{
                "Privilege": [ "ConfigureComponent" ]
            }
        ],
        "DELETE": [
            {{
                "Privilege": [ "ConfigureComponent" ]
            }
        ]
    },
    "ResourceURIOverrides": [
        {{
            "Targets": [
                "/redfish/v1/Systems/VM6",
                "/redfish/v1/Systems/Sys1"
            ],
            "OperationMap": {
                "GET": [
                    {{
                        "Privilege": [ "Login" ]
                    }
                ],
                "PATCH": [
                    {{
                        "Privilege": [ "ConfigureComponents","OEMSysAdminPriv" ]
                    }
                ]
            }


        }
    ]
}
```

9.2.9.8 特权 AND 和 OR 语法

对实体、实体要素或者资源执行某项操作所需要的特权的逻辑组合，可以通过操作映射中 GET、HEAD、PATCH、POST、PUT、DELETE 操作要素数组中的特权标签数组排列来进行定义。对于 OR 逻辑而言，操作要素数组中的特权标签按照单独要素的方式排列。在下面这个示例中，具备“登录”或者“OEM 特权 1”即可以执行 GET 操作。

```
{}{
    "GET": [
        {}{
            "Privilege": [ "Login" ]
        },
        {}{
            "Privilege": [ "OEMPrivilege1" ]
        }
    ]
}
```

对于逻辑 AND 组合而言，特权标签在操作要素内部按照特权属性数组的方式排列。在下面这个示例中，只有同时具备“配置部件”和“OEM 系统管理员”特权，才能够执行 PATCH 操作。

```
{}{
    "PATCH": [
        {}{
            "Privilege": [ "ConfigureComponents","OEMSysAdminPriv" ]
        }
    ]
}
```

10. Redfish 主机界面

在 Redfish 主机界面规范中，将对在主机计算机系统上运行的软件与管理主机的 Redfish 服务之间的界面和互动做出规定。更多详细信息参阅 DSP 0270 相关内容。

11. Redfish 可组合性

服务器可以在服务根目录下实施 CompositionService（组合服务）资源，以便支持将各种资源绑定在一起。其中一个示例就是分散化硬件，通过组合可以将独立的部件（例如：处理器、内存、输入/输出控制器、以及驱动器）绑定在一起，创建能够共同运行的逻辑实体。通过这种方式可以让客户端能够在特定引用中动态分配资源。

11.1 组合请求

实施 CompositionService（在 CompositionService schema 中进行定义）的服务器必须能够支持下列一种或多种类型的组合请求：

- 自定义组合。

支持删除组合资源的服务器，必须支持使用 DELETE 方法将组合资源删除。

11.1.1 自定义组合

自定义组合是指客户端识别确定了一组具体的资源，要求将其组合成一个逻辑实体。支持自定义组合请求的服务器必须实施 CompositionService 的 ResourceBlock（资源块）资源（ResourceBlock schema）和 ResourceZone（资源区）资源（Zone schema）。在 ResourceBlock 中将提供客户可以用于构建组合实体的部件清单。在 ResourceZone 中将对受服务器管理的 ResourceBlock 进行组合时需要遵守的限制条件。

CompositionService 中的 ResourceZone 资源应该在响应中包括一个 CollectionCapabilities（集合能力）注释项。通过 CollectionCapabilities 注释项，客户端可以发现服务器中哪些资源集合支持组合请求、对相关资源集合提出的 POST 请求需要采用何种形式、以及需要哪些属性。支持自定义组合的服务器必须支持含有 ResourceBlock 超链接数组的 POST 请求。ResourceBlock 数组的自定义嵌套将在被组合资源的 schema 中进行定义。

支持更新组合资源的服务器还必须能够支持通过修改后的 ResourceBlock 清单，对组合资源使用 PUT 以及/或者 PATCH 方法。

ComputerSystem 自定义组合的示例如下所述：

```
POST /redfish/v1/Systems HTTP/1.1
Content-Type: application/json;charset=utf-8
Content-Length: <computed length>
OData-Version: 4.0

{}{
    "Name": "Sample Composed System",
    "Links": {
        "ResourceBlocks": [
            { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/ComputeBlock0" },
            { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/DriveBlock2" } ,
            { "@odata.id":
"/redfish/v1/CompositionService/ResourceBlocks/NetBlock4" }
        ]
    }
}
```

12. 附件 A（供参考）

12.1 修订记录

版本	日期	描述
1.3.0	2017 年 8 月 11 日	新增了下列规定：“如果在返回 HTTP 428 响应代码时要求加入 If-Match 或者 If-Match-None 数据头，则支持服务可以拒绝 PATCH

		或 PUT 操作”。
		新增了下列规定：“在通过@Redfish.SettingsApplyTime 注释项为资源的设置对象赋值时，支持服务进行描述”。
1.2.1	2017 年 8 月 10 日	修改了“OEM”对象定义的措辞。
		修改了“部分资源结果”一条中的相关措辞。
		说明了在接收到带有空的 JSON 对象时服务器的具体行为。
		新增了关于 HTTP 503 响应代码的其他用途的说明。
		说明了满足 RFC 6901 要求的 URI 碎片的具体格式。
		说明了绝对和相对 URI 的用途。
		说明了来自于 OData 的“目标”属性的定义。
		说明了“超链接”和“Links 属性”之间的区别。
		修改了特权映射中的 JSON 示例。
		说明了“@odata.context”属性的格式。
		加入了说明 schema 文件命名公约的条款。
		说明了在接收到带有缺失属性的 PUT 请求时服务器的行为。
		说明了“Accept”数据头的有效值，根据 RFC 7231 规定加入了通配符。
		修正了“配置用户”特权的拼写（从“ConfigureUser”修改为“ConfigureUsers”）。
		修正了“会话登录”一条相关内容，加入了强制性语言方面的规定。
1.2.0	2017 年 4 月 14 日	新增了对 Redfish 组合服务的支持。
		说明了服务器对请求中的 Accept-Enconding 数据头的处理方式。
		增加了整个文件内请求和响应示例的一致性，规范了格式。
		纠正了在响应示例中使用的“@odata.type”属性。
		说明了“Required”schema 注释项的用途。
		说明了特权寄存器中从属取代的用途。
1.1.0	2016 年 12 月 9 日	加入了“Redfish 服务操作与特权映射”一条。此功能允许服务器在资源甚至属性层面上的 HTTP 操作与账号的角色和特权映射起来。
		新增了对“Redfish 主机界面规范”（DSP 0270）的引用。
1.0.5	2016 年 12 月 9 日	发布勘误版本，修订了各种印刷错误。

		规范了整个文件中“集合”、“资源集合”和“成员”等术语的使用。
		增加了“资源集合”和“成员”术语定义条目。
		修正了 RFC 5280 中的证书要求和引用定义和要求，同时将 RFC 5280 新增为本规范参考标准。
		说明了 HTTP POST 和 PATCH 操作的用途。
		说明了 HTTP 状态代码和错误响应的用途。
1.0.4	2016 年 8 月 28 日	发布勘误版本，修订了各种印刷错误。
		新增了 HTTP Link 数据头的示例，说明了其用途和内容。
		新增了“schema 变更”一条，用于描述 schema 文件的允许用途。
		新增了使用 TLS 1.2 或更新版本、以及遵守 SNIA TLS 规范的建议。新增了对 SNIA TLS 规范的引用。新增了使用 TLS_RSA_WITH_AES_128_CBC_SHA 密码组的建议。
		规定了“角色（Role）”资源的“ID”属性必须与角色名称相一致。
1.0.3	2016 年 6 月 17 日	发布勘误版本。修改了目录中缺失的内容和条款编号。修正了对外部规范文件的引用 URL。新增了缺失的参考标准。修改了 Etag 示例中的印刷错误。
		提供了使用 ExtendedInfo 来显示消息数组的示例。
		规定了对会话（Session）服务执行 POST 操作创建新会话时，不需要授权数据头。
1.0.2	2016 年 3 月 31 日	发布勘误版本，修订了各种印刷错误。
		修改了 M-SEARCH 查询和响应中使用的各种强制性语言。
		修改了 M-SEARCH 响应中的缓存控制和 USN 格式。
		修改了 schema 命名空间规则，使其能够满足 OData 命名空间要求（将.n.n.n 修改为.vn_n_n），同时更新了整个规范文件内的示例，使其满足此格式要求。JSON schema 和 CSDL（XML）schema 的文件命名规则也继续了修改，使其与此格式匹配，以便能够与未来主要版本（v2）兼容。
		加入了缺失的条款内容，详细说明了 schema 库的位置，以及 schema 库的永久性 URL。
		加入了对 Unit 注释项的取值的定义，使用 UCUM 规范中的定义。对整个文件的示例进行了更新，使其满足此标准形式要求。
		修改了 OEM 属性命名规则，以避免在未来在属性名称中使用冒号

		“:”和句号“.”，因为在部分编程语言或者环境中，这两个符号会导致变量名称无效或者出现问题。这两个分隔符都使用下划线“_”进行提单，废除了使用冒号和句号的规定（但依然有效）。
		从“安全性”一节中删除了重复的、或者不在本规范文件范围之内的内容。这些内容对 Redfish 服务实施过程提出了一些不在本文件范围内的要求。
		补充了之前缺失的要求，即要求资源响应中的属性名称的大小写必须与 schema 文件中的规定相一致。
		将参考标准更新为最新的 HTTP RFC，在整个规范文件中加入了对上述参考标准的引用。
		说明了 ETag 数据头的具体要求。
		规定了在访问服务器根目录资源时，不需要进行认证。
		加入了对检索集合的描述。
		说明了“charset=utf-8”在 HTTP Accpet 和 Content-Type 数据头中的用途。
		说明了“Allow”HTTP 响应数据头的用途，加入了“Retry-After”数据头用途的缺失表格内容。
		说明了类型属性和环境属性的强制性用途，解释了使用两种形式 URL 的能力，在整个文件中修正了“@odata.context”URL 示例。
		修改了“资源集合响应”一条中术语不统一的错误。
		修改了强制要求的资源成员属性（“成员”而不是“取值”）。
		规定了错误响应中可以包括关于多个错误条件的信息。
		在示例中，修改了 Measures.Unit 注释项的名称。
		在注释项的示例中，修正了对过期的核心 OData 规范的引用。
		在“通用 Redfish 资源属性”一条中，补充了之前缺失的“Members”属性。
		在“非同步操作”一条中，说明了任务监视器和相关操作的术语和用途。
		规定了 SSDP 协议的实施具有可选性。
		修正了 SSDP USN 一栏字符串定义中的印刷错误（修改为’::dmtf-org）。
		在允许 HTTP 方法列表中，补充了之前缺失的 OPTIONS 方法。
		在示例中规定了可为空性。
1.0.1	2015 年 9 月 17 日	发布勘误版本，修订了各种印刷错误。

		强制要求在 schema 文件中使用长描述注释项。
		说明了 link 数据头中“rel-describedby”的用途。
		在 OData 环境属性中的“选择清单”中，修改了文本错误。
		修改了 Accept-Encoding 请求数据头的处理方式。
		删除了“返回扩展错误资源”中重复和相冲突的内容。
		说明了相对 URI 相关规则。
		说明了 USN 格式。
1.0.0	2015 年 8 月 4 日	首次发布。