文档标识: **DSP0266**
日期: **2015-09-17**
版本: **1.0.1**

Redfish 可扩展平台管理 API 规范

取代: 1.0.0
文档分类: 标准
文档状态: 出版
文档语言: 中文-中国

目 录

前言

Redfish 可扩展平台管理 API("Redfish")是由 DMTF 可扩展平台管理论坛编写。

DMTF 是一个非盈利行业成员协会，致力于推广企业和系统管理以及互操作性。参见 http://www.dmtf.org。

致谢

DMTF 感谢下列人员对本文档所作出的贡献：

- Jeff Autor Hewlett-Packard Company
- David Brockhaus Emerson Network Power
- Richard Brunner VMware Inc.
- Lee Calcote Seagate Technology
- P Chandrasekhar Dell Inc
- Chris Davenport Hewlett-Packard Company
- Gamma Dean Emerson Network Power
- Wassim Fayed Microsoft Corporation
- Mike Garrett Hewlett-Packard Company
- Steve Geffin Emerson Network Power
- Jon Hass Dell Inc
- Jeff Hilland Hewlett-Packard Company
- Chris Hoffman Emerson Network Power
- John Leung Intel Corporation
- Milena Natanov Microsoft Corporation
- Michael Pizzo Microsoft Corporation
- Irina Salvan Microsoft Corporation
- Hemal Shah Broadcom Corporation
- Jim Shelton Emerson Network Power
- Tom Slaight Intel Corporation
- Donnie Sturgeon Emerson Network Power
- Pawel Szymanski Intel Corporation
- Paul Vancil Dell Inc
- Linda Wu Super Micro Computer, Inc.

1 摘要

Redfish 可扩展平台管理 API(The Redfish Scalable Platforms Management API ("Redfish"))是一种新的规范,其使用 RESTful 接口语义来访问定义在模型格式中的数据,用于执行带外系统管理 (out-of-band systems management)。其适用于大规模的服务器,从独立的服务器到机架式和刀片式的服务器环境，而且也同样适用于大规模的云环境。

现在行业中已有几个带外系统管理标准(事实标准和法律标准)。在实现时，他们都有很大的差别。他们是针对嵌入式环境的单一服务器而开发，或基于过时的软件建模结构。当前，没有一种业界标准，既简单易用，符合新兴编程标准，易于嵌入，又能满足大型数据中心和云计算的需求。

2 引用标准

以下引用文档对这个文档的应用程序是不可缺少的。对于有日期或版本的引用,只有引用的版本(包括任何需要改正之处或 DMTF 更新版本)适用。对于没有日期或版本要求的引用,引用的文件的最新出版的版本 (包括任何需要改正之处或 DMTF 更新版本)可以适用。

- IETF RFC 2616, R. Fielding et al., HTTP/1.1, http://www.ietf.org/rfc/rfc2616.txt
- IETF RFC 2617 J. Franks et al., HTTP Authentication: Basic and Digest Access Authentication, http://www.ietf.org/rfc/rfc2617.txt
- IETF RFC 3986 T. Berners-Lee et al, Uniform Resource Identifier (URI): Generic Syntax, http://www.ietf.org/rfc/rfc3986.txt
- IETF RFC 4627, D. Crockford, The application/json Media Type for JavaScript Object Notation (JSON), http://www.ietf.org/rfc/rfc4627.txt
- IETF RFC 4627, L. Dusseault et al, PATCH method for HTTP, http://www.ietf.org/rfc/rfc5789.txt
- IETF RFC 5988, M. Nottingham, Web linking, http://www.ietf.org/rfc/rfc5988.txt
- IETF RFC 6901, P. Bryan, Ed. et al, JavaScript Object Notation (JSON) Pointer, http://www.ietf.org/rfc/rfc6901.txt
- IETF RFC 6906, E. Wilde, The 'profile' Link Relation Type, http://www.ietf.org/rfc/rfc6906.txt
- ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards, http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtypeH
- JSON Schema, Core Definitions and Terminology, Draft 4 http://tools.ietf.org/html/draft-zyp-json-schema-04.txt
- JSON Schema, Interactive and Non-Interactive Validation, Draft 4 http://tools.ietf.org/html/draft-fge-json-schema-validation-00.txt
- OData Version 4.0 Part 1: Protocol. 24 February 2014. http://docs.oasis-open.org/odata/odata/v4.0/os/odata-v4.0-part1-protocol.html

• OData Version 4.0 Part 2: URL Conventions. 24 February 2014. http://docs.oasis-open.org/odata/odata/v4.0/os/odata-v4.0-part2-url-conventions.html

• OData Version 4.0 Part 3: Common Schema Definition Language (CSDL). 24 February 2014. http://docs.oasis-open.org/odata/odata/v4.0/os/odata-v4.0-part3-csdl.html

• OData Version 4.0: Core Vocabulary. 24 February 2014. http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Core.V1.xml

• OData Version 4.0 JSON Format. 24 February 2014. http://docs.oasis-open.org/odata/odata-json-format/v4.0/os/odata-json-format-v4.0-os.html

• OData Version 4.0: Units of Measure Vocabulary. 24 February 2014. http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Measures.V1.xml

3 术语和定义

在本文档中，有些条款有一个特定的意义超出了正常的英语意思。这些术语在这一条款中被定义。

本文档中，术语“应当”（“需要”），“不得”，“应该”（“推荐”）、“不应该”（“不推荐”），“可能”、“不需要”（“不需要”），“能”与“不能”等参看 ISO / IEC 指示第 2 部分附件 H 中描述的解释。当前面引号中的术语在特殊情况下由于语言原因不能使用时，括号中的术语可以替换。请注意，ISO / IEC 指示第 2 部分附件 H 指定额外的替代选择。出现这样的额外替代选择时，应当使用正常的英语意思解释。

在这个文档中，术语“条款”、“分条款”、“段”，“附件”的解释参看 ISO / IEC 指示第 2 部分条款 5 中的描述。

这个文档中的术语“规范”和“信息”的解释参看 ISO / IEC 指令第 2 部分条款 3 中的描述。在本文档中，条款、子条款或标记“(信息)”的附件不包含规范的内容。注意事项和例子总是信息的元素。

以下附加条款在本文档中被使用。

术语	定义
基板管理控制器（Baseboard Management Controller）	嵌入式设备或服务通常是计算机系统内一个带有固件，独立的微处理器或是片上系统，用来执行系统监控和管理相关任务，一般是带外执行的。
集合(Collection)	集合是一种资源,其充当其它资源的一个容器。集合的成员通常有类似的特征。容器处理发送到容器中的消息。容器的成员只处理发送到某个成员的消息,而不影响容器的其他成员。
CRUD	任何接口使用的基本的内在操作:创建、读取、更新和删除。
事件	对应于一个单独的警告的记录。

受管理系统（managed system）	在本规范的背景下,一个受管理系统是一个系统。通过一个 Redfish 定义的接口，其提供信息或状态,或者是可控制的。
消息	一个完整的请求或响应,使用 HTTP / HTTPS 格式。基于 REST 的协议是一个请求/响应协议。其中，每个请求都应该产生一个响应。
操作	HTTP 请求的方法，其映射通用的 CRUD 操作。它们是 POST、GET、PUT / PATCH, HEAD 和 DELETE。
OData	开放数据协议,如 OData-Protocol 中定义。
OData 服务文档	资源的名称,它提供服务根的信息。服务文档提供了一个标准格式，用于列举服务使用的资源。该服务使通用的超媒体驱动(hypermedia-driven) OData 客户能够导航到 Redfish 服务的资源。
Redfish 警告接收机（receiver）	从 Redfish 服务接收警告的功能的名称。这个功能通常是运行在一个远程系统的软件。其中，远程系统可以独立于管理系统。
Redfish 客户（client）	与 Redfish 服务通信，并访问一个或多个资源或服务的功能的名称。
Redfish 协议	协议的集合,用于对 Redfish 服务进行发现、连接和内部通信。
Redfish 模式	Redfish 资源的模式定义。其根据 OData 模式表示进行定义，其可以直接转换为一个 JSON 模式表示。
Redfish 服务	也称为“服务”。功能的集合实现了协议、资源,以及功能。其中，功能用于交付该规范定义的接口，以及针对一个或多个管理系统的该规范的相关行动。
Redfish 服务入口点（entry point）	也称为“服务入口点”。通过该接口，一个特定的 Redfish 服务实例可以被访问。一个 Redfish 服务可以有多个服务接入点。
请求	从客户端到服务器的一个消息。它由一个请求行构成(包括操作),请求头,一个空行和可选的消息体。
资源	资源是通过 URI 寻址，并能够接收和处理消息。资源可以是一个单独的实体,或为几个其他实体充当一个容器的一个集合。
资源树	资源树是一个使用 JSON 编码资源的树结构，其可以通过一个已知的开始 URI 接入。一个客户可能会在树的底部通过资源链接发现 Redfish 服务上可用的资源。 Redfish 客户实现的注意事项:虽然资源是一棵树,资源之间的引用可能会形成图而不是树。客户遍历资源树时，必须包含逻辑来避免无限循环。
响应	从一个服务器到客户端的消息，来响应请求消息。它由一个状态行,响应头,一个空行和可选的消息体构成。
服务根	术语服务根用于指一个特定的资源。通过服务接入点可以直接访问该资源。这个资源作为起点用于定位和访问其它资源。该资源与相关的元数据一起形成一个 Redfish 服务的实例。
订阅	为了能够接收事件而注册一个目的地的行动。

4 符号和缩写形式

本文档中使用以下额外的缩写。

术语	定义
BMC	基板管理控制器(Baseboard Management Controller)
CRUD	创建、替换、更新和删除（Create, Replace, Update and Delete）
CSRF	跨站点请求伪造(Cross-Site Request Forgery)
HTTP	超文本传输协议（Hypertext Transfer Protocol）
HTTPS	基于 TLS 的超文本传输协议（Hypertext Transfer Protocol over TLS）
IP	互联网协议(Internet Protocol)
IPMI	智能平台管理接口(Intelligent Platform Management Interface)
JSON	JavaScript 对象表示法（JavaScript Object Notation）
KVM-IP	基于 IP 的键盘、视频、鼠标重定向(Keyboard, Video, Mouse redirection over IP)
NIC	网络接口卡（Network Interface Card）
PCI	外围组件互连(Peripheral Component Interconnect)
PCIe	串行总线 （PCI Express）
TCP	传输控制协议(Transmission Control Protocol)
XSS	跨站点脚本(Cross-Site Scripting)

5 概述

Redfish 可扩展的平台管理 API(The Redfish Scalable Platform Management API ("Redfish"))是一个管理标准，使用数据模型表示并且包含超媒体 RESTful 接口。因为它是基于 REST, REST 比其他解决方案更容易使用和实施。因为它是面向模型的,它能够表达现代系统组件之间的关系，以及服务和组件的语义。它也可以很容易地扩展。通过使用 REST 超媒体方法, Redfish 可以表达来自多个供应商的各种各样的系统。通过要求 JSON 表示,各种各样的资源可以被规范化的方式创建，不仅是为了提高可扩展性,而且在大多数编程环境中负载（payload）可以很容易地被解释，以及人检查数据时有较好的相对直观性。模型是采用一个可互操作的 Redfish 模式展现,并且采用 OData 模式表示并可以翻译为一个 JSON 模式表示。并且消息的负载采用符合 OData JSON 约定的 JSON 表示。将资源的 Redfish 模式定义放入一个机器可读格式的能力，允许数据与元数据之间建立关联，并且没有阻碍 Redfish 服务和元数据,从而实现更高级的客户场景（例如许多数据中心和云环境）。

5.1 范围

本规范的范围是定义协议,数据模型,和行动,以及内部可操作、跨厂商、远程和带外能力接口等其他架构方面的组件。其中，远程和带外能力接口满足云计算和基于 web 的 IT 专业人员对可扩展平台管理的期望。而大规模系统是主要的焦点,此规范也能够被用于更多传统的系统平台管理实现。

规范定义了对于所有 Redfish 实现的必选元素，以及被系统供应商或制造商可选的元素。规范也定义了一些要点，其中，特定 OEM(系统供应商) 扩展可以被一个给定实现提供。

规范设置了 Redfish 服务和相关材料的规范要求,如 Redfish 模式文件。一般来说,规范没有设置 Redfish 客户要求,但会显示 Redfish 客户应该做什么来成功的和有效的访问和使用 Redfish 服务。

对于实现 Redfish 接口和功能必须使用的特定硬件或固件，规范没有设定要求。

5.2 目标

作为一个架构、一种数据表示、以及用于访问与 Redfish 服务交互协议的定义，Redfish 有多个目标和目的。Redfish 旨在提供规范实现以下目标:

- 可扩展(scalable)——支持云服务环境中独立机器和机架设备。
- 灵活(flexible)——支持当今服务中的各种各样的系统。

可延展(extensible)—— 在数据模型框架内支持新的和特定厂商提供的能力

- 向后兼容(backward compatible)——使新功能可以被添加,同时保留规范的早期版本。
- 互操作(interoperable)——提供一个有用的需求基线，从而确保跨多个供应商的常见功能和实现的一致性。
- 专注系统(system-focused)——有效支持所需的最常见的平台硬件管理功能，从而被应用在可扩展的环境,同时也能够管理当前的服务器环境。
- 基于标准(standards based)——利用被广泛接受和在今天的环境中使用的协议和标准,特别是, 今天被广泛采用的基于 web 客户端的编程环境。
- 简单（simple）——可以被软件开发人员直接使用，不需要高度专业化的编程技能或系统知识。
- 轻量级（lightweight）-减少在管理系统上实现和验证 Redfish 服务的复杂性和成本。

5.3 设计原则

以下设计原则和技术用来帮助交付之前所述的目标和特点:

- 提供一个使用 JSON 负载和实体数据模型的 RESTful 接口
- 从数据模型分离协议，使他们能够被独立修改
- 指定协议和模式的版本规则。
- 利用互联网协议标准优势，满足构建要求,比如 JSON,HTTP,OData,以及本文档所引用的 RFC。
- 集中在带外访问,可在现有 BMC 和固件产品上实现
- 组织模式以呈现增值特性与标准化的项目
- 使数据尽可能与上下文中的定义一样明白
- 保持实现的灵活性。不将接口与任何特定的底层实现架构进行绑定。“规范接口而不是实现。”
- 专注于最广泛使用的“共同特性（common denominator）”功能。避免增加复杂性到只有一小部分用户可以使用的地址功能，。
- 避免放置复杂性到管理控制器上，从而可以更好地支持在客户端上的操作。

5.4 限制

Redfish 无法保证客户端软件永远不需要更新。可能需要更新的例子包括容纳新类型系统、组件、数据模型更新等等。为应用程序的系统优化总是需要构建监督。然而,通过使用模式,严格的版本，向前兼容性规则，并通过从数据模型中分离协议, Redfish 试图最小化客户端需要升级的实例。

内部可操作并不意味着完全相同。Redfish 客户可能需要适应不同供应商提供的可选的元素。实现和配置一个给定的供应商的一个特定的产品也可以有所不同。

例如, Redfish 不能使一个客户端读资源树和写资源到另一个 Redfish 服务。这是不可能的,因为它是一个超媒体 API。只有根对象有一个众所周知的 URI。资源拓扑反映了系统的拓扑结构和它代表的设备。因此,不同的服务器或设备类型将导致不同的资源树,甚至对于同一制造商的相同系统。

此外,并不是所有的 Redfish 资源是简单的读/写资源。实现可能遵循稍后讨论的其他交互模式。作为一个例子,用户凭证或证书不能简单地从一个服务读取并移植到另一个服务。另一个例子是使用设置数据而不是写入读取的相同资源。

最后,资源和其他元素之间的联系值可以在多个实现中变化。客户不应该假设：链接可以在 Redfish 服务的不同实例化之间重用。

5.5 额外的设计背景和基本原理

5.5.1 基于 REST

本文档定义了一个 RESTful 接口。许多服务应用程序呈现了 RESTful 接口。

有几个理由来定义一个 RESTful 接口:

- 它是一个轻量级实现,经济是必要的(数据传输比 SOAP 少,协议层比 WS-Man 少)。
- 这是业界流行的访问方法。
- 很容易学习并且用文档记录。
- 有许多可被用于 REST 的工具包和开发环境。
- 它支持数据模型语义和容易映射到常见的 CRUD 操作。
- 它符合我们简单的设计原理。
- 同样适用于软件应用程序空间,因为它是为嵌入式环境,从而实现融合以及在管理生态系统内共享组件代码。
- 它是模式无关，所以适应于任何建模语言。
- 通过使用它,Redfish 可以利用行业中现有的安全与发现机制。

5.5.2 遵循 OData 约定

随着 RESTful api 的流行,有与应用程序同样多的 RESTful 接口。遵循 REST 模式有助于促进良好实践,由于许多 RESTful api 设计之间的差异，它们之间并没有互操作性。

OData 定义了一组常见的 RESTful 约定和标记。如果采用,它提供 api 之间的互操作性。

采用 OData 约定描述 Redfish 模式,URL 约定,以及在一个 JSON 负载的通用属性的命名和结构,不仅对 RESTful api 封装最佳实践,而且进一步使 Redfish 服务被越来越多的通用客户端库、应用程序和工具的生态系统所采用。

5.5.3 面向模型

Redfish 模型为管理系统而建立。所有资源被定义在 OData 模式表示，并且转换成 JSON 模式表示。OData 是一种工业标准，为 RESTful 服务封装最佳实践,并提供跨越不同类型服务的互操作性。当采用这些方法时，JSON 是被广泛应用于多个学科，并且有大量的工具和快速发展的编程语言。

5.5.4 从数据模型中分离协议

指定的协议操作是独立于数据模型。版本化的协议也是独立的数据模型。如果协议版本变化非常频繁,数据模型版本可以根据需要改变。这意味着创新应主要发生在数据模型中,而不是协议中。它允许数据模型根据需要被扩展和改变，不需要改变协议或 API 版本。相反,从数据模型中分离协议允许更改协议，不需要引起数据模型的重大变化。

5.5.5 超媒体 API 服务端点

类似其它超媒体 API, Redfish 采用单一服务端点 URI。所有其它资源都通过从根引用的不透明的 URI 进行访问。

通过访问根服务或任何服务创建的链接发现的任何资源，或从根服务引用的资源，将符合被根服务支持的相同协议版本。

5.6 服务元素

5.6.1 同步和异步操作支持

虽然在此体系结构中大部分的操作在本质上是同步的, 一些操作需要花很长时间来执行,比客户通常期待的等待时间更长。出于这个原因, 在服务判定时一些操作可以是异步的。一个异步操作的请求部分与同步操作的请求部分是不同的。

使用 HTTP 响应代码让客户确定操纵是同步还是异步。更多的信息请参见“任务”部分。

5.6.2 事件机制

在某些情况下，一个服务提供给客户信息是有用的。这些信息不属于正常的请求/响应模式。这些消息（称为事件）被服务用于异步通知客户一些重要的状态变化或错误条件,或者通常的一个时间关键性质。

目前只有一个事件的类型在这个规范中被定义——推样式事件(push style eventing)。推样式事件中,当服务器检测到需要发送一个事件,它使用一个 HTTP POST 将事件消息推给客户。通过在事件服务中创建一个事件终点（EventDestination）订阅订阅条目,或管理员可以创建订阅作为 Redfish 服务配置的一部分，客户可以订阅事件服务从而接收事件。所有订阅都是持久的配置设置。

事件源于一个特定的资源。并不是所有的资源都能够生成事件。这些可能会生成事件的资源不能产生任何事件,除非被创建的订阅可以侦听事件。通过发送"subscribe"消息到事件服务,管理员或客户端创建一个订阅。使用 HTTP POST 发送订阅消息到事件订阅集合。

这个规范的“事件”部分进一步讨论事件机制的细节。

5.6.3 行动

操作可以被分为两组:内在和外在。内在的操作通常被称为 CRUD,可以被映射到 HTTP 方法。协议也有支持外部操作的能力——那些不容易映射到 CRUD 的操作。外在的例子是作为一个集合被更好的执行的项目(为了可扩展性、容易的接口,服务器端语义保存或类似的原因)或没有自然映射到 CRUD 的操作。一个例子是系统复位（reset)。将多个操作组合到一个单一的行动是可能的。系统复位可以建模为一个状态更新,但是语义上客户端实际上请求状态改变,而不是简单地改变状态的值。

在 Redfish 中,这些外在的操作被称为行动，本规范的不同部分进行了详细讨论。

Redfish 模式定义了与常见 Redfish 资源关联的某些标准操作。对这些标准的行动,Redfish 模式包含规范性语言行动。OEM 扩展也允许 Redfish 模式,包括为现有资源定义的行动。

5.6.4 服务入口点的发现

服务本身有一个众所周知的 URI,服务主机必须能够被发现。Redfish（像 UpnP）使用 SSDP 发现。SSDP 是支持各种各样的设备,例如打印机。它是简单的、轻量级的,IPv6 能力的,并且适合在嵌入式环境中实现。Redfish 正在调查额外服务入口点发现方法(例如 DHCP-based)。

有关更多信息,请参见“发现”章节。

5.6.5 远程访问支持

各种各样的远程访问和重定向服务在此体系结构中被支持。带外环境的关键是支持串行控制台访问、键盘视频和鼠标重定向 (KVM-IP), 命令 Shell(即命令行接口)和远程虚拟介质等机制。支持串行控制台、命令 Shell,KVM-IP 和虚拟介质都包含在此标准中，并且采用 Redfish 模式表达。这个标准中没有定义协议或访问机制，用于访问这些设备和服务。Redfish 模式提供了这些服务的表示和配置,建立连接使这些服务可用、以及获取这些服务的运行状况。然而,协议本身的规范是本规范以外的范围。

5.7 安全

远程的可编程接口的安全挑战是确保用于与 Redfish 互动的接口和交换的数据是安全的。这意味着设计接口相关的适当安全控制机制，以及确保用来交换数据的通道的安全。作为安全的一部分,特定的行动都要到位，包括定义和使用最低级别加密通信通道等。

6 协议细节

Redfish 可扩展的平台管理 API 是基于 REST 和遵循互操作性（例如被定义在 OData 协议中）、JSON 负载(定义在 OData-JSON 中)、以及一个机器可读的模式表示（定义在 OData-Schema 中）的 OData 约定。OData 模式表示包括为了验证直接翻译为 JSON 模式表示的注释,以及由支持 JSON 模式工具的使用。以下这些常见的标准和约定提高了互操作性，并使利用现有的工具链可行。

为客户消费最小的元数据，Redfish 遵循 OData 最小的一致性水平。

在这个文件中,我们称 Redfish 为映射到一个数据模型的协议。更准确地说,HTTP 是应用程序协议，用于传输消息。TCP / IP 传输协议。RESTful 接口是到消息协议的一个映射。为简单起见,我们将参考（refer to）RESTful 映射到 HTTP、TCP / IP 等协议，传输和消息层方面作为 Redfish 协议。

Redfish 遵循 OData 最小的一致性水平方便客户端使用最小的元数据，。

在这个文档中,我们称 Redfish 为映射到一个数据模型的协议。更准确地说,HTTP 是应用程序协议，用于传输消息。TCP / IP 传输协议。RESTful 接口是到消息协议的一个映射。为简单起见,我们将参考（refer to）RESTful 映射到 HTTP、TCP / IP 等协议，传输和消息层方面作为 Redfish 协议。

Redfish 协议是围绕基于接口模型的 web 服务而设计，并且在用户界面(UI)和自动化的使用方面为提高网络和交互的效率而设计。接口是专门设计为 REST 模式语义。

对于常见的 CRUD 操作和检索头信息，HTTP 方法被 Redfish 协议所使用。

行动用于扩大的操作,行动是用于扩大操作超越 CRUD 操作类型的，但应该限制使用。

媒体类型是用来协商数据类型，它作为消息的内容被发送。

HTTP 状态代码是用来指示服务器处理请求的尝试。扩展的错误处理是用来返回比 HTTP 错误代码提供的信息更多的信息。

发送安全信息的能力是非常重要的;本文档的安全部分描述特定的 TLS 需求。

一些操作可能需要比同步返回语义需求更长的时间。因此,确定异步语义包括在架构中。

6.1 使用 HTTP

HTTP 特别适合于一个 RESTful 接口。本节描述如何在 Redfish 接口中使用 HTTP，以及什么约束被添加在 HTTP 之上,以确保 Redfish 兼容实现的互操作性。

- Redfish 接口应当通过 web 服务端点公开，并使用超文本传输协议(Hypertext Transfer Protocols)（ 1.1 版）(RFC2616)实现。

6.1.1 URI

使用一个 URI 来识别一个资源,包括基本服务和所有 Redfish 资源。

- 资源的每个独特的实例都应该被 URI 惟一的标识。因此，虽然一个 URI 可以引用一个单一集合资源，但是它不能引用多个资源。
- 一个 URI 应该被客户端不透明的处理,并且不应该被应用标准引用解析规则之外的客户端试图去理解或解构，在 RFC3986 的第 5 章节引用解析中所定义。

开始操作时，客户端必须知道一个资源的 URI。

- 执行一个 GET 操作产生资源的一种表示，其包括相关资源的属性和链接。

基础资源 URI 是众所周知的，并且基于协议版本。发现额外资源的 URI 是通过观察以前响应返回的相关资源的链接。被服务返回的导航 URI 使用的 API 类型被称为超媒体 API。

Redfish 考虑在 RFC3986 中描述的 URI 的三个部分。

第一部分包括 URI 的模式和授权。第二部分包括根服务和版本。第三部分是一个独特的资源标识符。

例如,下面的 URL:

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

- 第一部分是模式和授权部分(https://mgmr.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 的模式和授权部分不应该被认为是资源的惟一标识符的一部分。这是由于重定向功能和本地操作可能会导致连接部分的可变性。剩下的 URI(服务和资源路径)惟一地标识资源,这就是所有的 Redfish 负载返回的内容。

- URI 的惟一标识符部分在实现中应当是独一无二的。

例如, 在响应的头位置 (指示 POST 创建的新资源), 一个 POST 可能会返回以下 URI:
例如: /redfish/v1/Systems/2

假设客户端是通过一个名为“mgmt.vendor.com”的设备连接, 访问这个新资源所需的完整的 URI 是 https://mgmt.vendor.com/redfish/v1/Systems/2。

URI(如 RFC3986 所述)也可能包含一个查询(?query)和碎片(#frag)组件。可参阅查询参数部分。当提交操作使用 URI 时, 碎片(#frag)应当被服务器忽略。

6.1.2 HTTP 方法

RESTful 接口的一个有吸引力的功能是非常有限的被支持的操作。下表描述了 HTTP 方法操作的一般映射。如果列中的值标题“要求”的值是“是”,那么,HTTP 方法应当被 Redfish 接口支持。

HTTP 方法	接口的语义	要求
POST	对象的创建、对象行动, 事件	是
GET	对象或集合检索	是
PUT	对象替换	否
PATCH	对象更新	是
DELETE	对象删除	是
HEAD	对象或集合标题检索	否

其他 HTTP 方法不被允许, 并应当收到一个 405 响应。

6.1.3 HTTP 重定向（Redirect）

HTTP 重定向允许服务将请求重定向到另一个 URL。除此之外,这使得 Redfish 资源对应于数据模型的别名区域。

- 所有 Redfish 客户端应当正确处理 HTTP 重定向。

注意: 关于 HTTP 重定向的安全影响, 请参考“安全”部分。

6.1.4 媒体类型

一些资源可以采用多个类型表示。表示的类型由媒体类型显示。

在 HTTP 消息中，媒体类型在 Content-Type 头中被指定。通过设置 HTTP Accept 头到一个可接受的媒体类型列表，客户端可以告诉一个服务“它希望得到使用某些媒体类型的响应”。

- 所有可用资源应当使用 JSON 媒体类型"application/json"。
- Redfish 服务应当使每个资源基于 JSON 表示而可获得(如 RFC4627 中规定)。因为它是用 JSON 编码,接收方不得拒绝一个消息,并提供至少一个基于 JSON 响应的表示。使用 non-JSON 媒体类型，一个实现可以提供额外表示。

客户端通过在请求中通过指定一个 Accept-Encoding 头来要求压缩。

- 如果被客户端请求，GET 请求的响应只能被压缩。
- 当被客户端请求时，服务应该支持 gzip 压缩。

6.1.5 ETags

为了减少不必要的 RESTful 资源访问的情况, Redfish 服务应该支持将一个单独的 ETag 与每个资源进行关联。

- 实现应该支持为每个资源返回 ETag 属性。
- 实现应该支持为每个响应返回 ETag 头，用于代表单个资源。实现应支持为“安全”部分列出的某些请求和响应返回 ETag 头。

因为服务是在最佳的位置被知道的。所以，如果对象的新版本足以被认为实质上是不同的， ETag 作为资源负载的一部分被生成和提供。有两种类型的 ETags:弱和强。

- 弱模型 -- 只有对象的“重要”部分被包含在 ETag 的制定中。例如,元数据(比如，最近的修改时间)不应包括在 ETag 生成中。决定 ETag 改变的“重要”属性包括可写的设置和可改变的属性,如 UUID,FRU 数据,序列号等。
- 强模型——对象的所有部分都包含在 ETag 的制定中。

本规范并不强制要求使用某个特定的算法用于创建 ETag,但 ETag 应该高度无冲突。ETag 可以是一个散列(hash),一个生成 ID,一个时间戳，或当底层对象变化时可以发生变化的值。

如果一个客户端 PUTs 或 PATCHes 一个资源,它应该在之前 GET 的 HTTP If-Match/If-None-Match 头中包括一个 ETag。

除了在每个资源返回 ETag 属性。

- 在客户端的 PUT/POST/PATCH 上，一个 Redfish 服务应该返回 ETag 头。
- 在个别资源的 GET 上，一个 Redfish 服务应该返回 ETag 头。

ETag 头的格式是:

ETag W/"<string>"

6.2 协议版本

协议的版本号是独立于资源的版本或它们支持的 Redfish 模式版本。

Redfish 协议的每个版本是强类型。这是通过使用 Redfish 服务 URI 和 URI 中获得的资源来完成,被称为根服务（ServiceRoot）。

这个 Redfish 协议版本的根 URI 应当是“/redfish / v1 /”。

当协议的主要版本在 URI 中表示时， 协议的主要版本、小版本和勘误表版本等在根服务资源的版本属性中表示。例如在那个资源的 Redfish 模式中的定义。该协议版本是一个字符串形式:

MajorVersion.MinorVersion.Errata

其中:
主要版本（Major Version）=整数: 类中的一些内容发生了一种向后不兼容的改变。
小版本（Minor Version）=整数:一个小更新。新功能可以被添加但不能被删除。与之前的小版本是兼容的。
勘误表（Errata）=整数:之前的版本好像有什么东西损坏了,需要被修复。

通过使用来自根服务的引用引用的任何服务或资源和通过使用访问根服务的链接发现的任何资源应当符合根服务支持的协议的相同版本。

在资源“/Redfish ”上的一个 GET 应当返回下列内容:

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

6.3 Redfish 定义 URI 和 相对 URI 规则

Redfish 是一个超媒体 API 以及一些定义 URI 的小集合。其他资源都是通过从根服务引用不透明的 URI 进行访问。下列 Redfish 定义 URI 应该被一个 Redfish 服务支持:

URI	描述
/Redfish	用于返回版本的 URI
/redfish/v1/	Redfish 根服务的 URI
Redfish / v1 / odata	Redfish OData 服务文档的 URI
/Redfish / v1 / $metadata	Redfish 元数据文档的 URI

此外,下列末尾没有斜杠的URI应该被重定向到在下表中所示的关联Redfish 定义URI,

或者是被服务视为相当于关联 Redfish 定义 URI 的服务:

URI	关联的 Redfish 定义 URI
/redfish/v1	/redfish/v1/

被服务使用的所有相关 URI 应该以双斜杠“//”开始，并且包括授权（例如，//mgmt.vendor.com/redfish/v1/Systems）。或者以一个单一的斜杠“/”开始，并且包括绝对路径（例如，/redfish/v1/Systems）。

6.4 要求

本节描述的请求可以被发送到 Redfish 服务。

6.4.1 请求头

HTTP 定义了头可用于请求消息。对于 Redfish 服务，下表定义了那些头和他们的需求。

- 如果需求列中的值被设置为“是”，像 HTTP 1.1 规范中定义的一样，Redfish 服务应当理解并能够处理在下表中的头。
- 如果需求列中的值被设置为描述中条件下的“条件（Conditional）”，像 HTTP 1.1 规范中定义的一样，Redfish 服务应当理解并能够处理在下表中的头。
- 如果需求列中的值被设置为“否”，像 HTTP 1.1 规范中定义的一样，Redfish 服务应该理解和能够处理在以下表中的头。

头	需求	支持值	描述
Accept	是	RFC 2616,14.1 节	向服务器指出这个客户端准备接受什么媒体类型。application / json 应该支持请求的资源。application/xml 应当支持请求的元数据。
Accept-Encoding	否	RFC 2616,14.4 节	指出 gzip 编码可以由客户端处理。如果 Accept-Enconding 头出现在请求中，并且根据 Accept-Enconding 服务不能发出“哪一个是可接收的”响应，那么服务应当以状态码 406 响应。如果 Accept-Enconding 头不在请求中，服务不能返回被 gzip 编码的响应。
Accept-Language	否	RFC 2616,14.4 节	这个头是用来指示响应中的语言请求。如果这个头没有被指定,设备默认位置将被使用。
Content-Ty	条件	RFC	描述了用于消息正文表示的类

pe		2616,14.17 节	型。charset=utf-8 应当支持有一个正文的请求。如果有正文请求，则需要该项。
Content-Length	否	RFC 2616,14.3 节	描述了消息体的大小。表示消息体的大小的一个可选的方法是使用 Transfer-Encoding:分块,其不使用 Content-Length 头。如果一个服务不支持 Transfer-Encoding，并且需要 Content-Length 代替,服务将采用状态码 411 响应。
OData-MaxVersion	否	4.0	显示 odata-aware 客户机能理解的 OData 的最高版本
OData-Version	是	4.0	服务应该驳回一个不支持 OData 版本的请求。
Authorization	条件	RFC 2617,第 2 节	所需基本授权
User-Agent	是	RFC 2616,14.43 节	跟踪产品令牌和他们版本的需求。多个产品令牌可以被列出。
Host	是	RFC 2616,14.23 节	要求在一个单一的 IP 地址允许支持多个来源主机。
Origin	是	W3C CORS,5.7 节	当阻止 CSRF 攻击时，允许 web 应用程序使用 Redfish 。
Via	否	RFC 2616,14.45 节	指出网络层次和识别消息循环。每一次通过需要插入自己的 VIA。
Max-Forwards	否	RFC 2616,14.31 节	限制了网关和代理跳转。防止消息在网络中被无限期的保存。
If-Match	条件	RFC 2616,14.31 节	If-Match 应当支持 AccountService 对象上的原子请求。服务返回 ETags 时，If-Match 应当支持对资源的 PUT 和 PATCH 请求。
If-None-Match	否	RFC 2616,14.31 节	如果这个 HTTP 头出现，当资源的 ETag 不匹配这个头中发送的 ETag 时，服务只会返回所请求的资源。如果指定的 ETag 头匹配资源的当前 ETag,GET 返回的状态码是 304。

如果需求列中的值设置为“是”，Redfish 服务应当理解并能够处理下表中这个规范中定义的头。

Header	需求	支持的值	描述
X-Auth-Token	是	不透明编码八	用于用户会话

		字节字符串	的身份验证。令牌值应是随机的。

6.4.2 读请求(GET)

GET 方法用于检索资源的表示。该表示可以是一个资源或集合。服务将返回使用在 Accept 头中指定的一个媒体类型的表示,从而符合媒体类型部分媒体类型的需求。如果 Accept 头不存在,服务返回的资源表示为 application / json。

- HTTP GET方法应当用于检索资源，并不会引起任何副作用。
- 服务应当忽略GET的正文内容。
- 在缺乏外部资源的变化时，GET操作应当是幂等的

6.4.2.1 服务根请求

Redfish 版本1服务的根URL应当是“/redfish / v1 /”。
服务的根URL返回此规范中定义的ServiceRoot资源。

6.4.2.2 元数据文档请求

Redfish 服务应当公开在"/redfish/v1/$metadata"资源中描述的服务的元数据文档。这个元数据文档描述了在根部可用的资源和集合,并且引用服务显示的全套资源类型描述的其他元数据文档。

为了检索元数据的文档，服务不要求身份验证。

6.4.2.3 OData 服务文档的请求

在"/redfish/v1/odata"资源中，Redfish 服务应当公开OData服务文档。这个服务文档提供了一个标准格式，用于列举服务所暴露的资源,使通用超媒体驱动的 OData客户端能够导航到服务的资源。

为了检索服务的文档，服务不要求身份验证。

6.4.2.4 资源检索请求

对于个人资源或资源集合，客户端通过发出GET请求到URI请求资源。资源或资源集合的URI可以从前一个请求返回的一个资源标识符属性中获得(例如,之前返回资源的链接部分)。通过添加一个包含属性名的片段到资源的URI中，服务可能（但不需要）支持检索一个资源的单个属性的公约。

6.4.2.4.1 查询参数

当处理的资源是一个集合时,客户端可以使用下面的分页查询选项指定被返回的成员的一个子集。这些分页查询选项适用于集合资源的成员属性。

属性	描述	例子
$skip	整数表示在检索第一个资源之前集合中跳过检索的资源的数量。	http://collection?$skip=5
$top	整数表示响应中集合的数量。这个参数的最小值是1。默认行动是返回所有成员。	http://collection?$top=30

- 服务应该支持$top 和 $skip的查询参数。
- 对于以不被支持的“$”开始的任何查询参数,实现应当返回501（没有实现未被执行)状态代码,而且应该返回一个扩展错误用于指示请求的查询参数(s)不支持这个资源。
- 实现应当忽略未知或不支持“不以’$’开始”的查询参数。

6.4.2.4.2 检索集合

检索一个集合是通过发送HTTP GET方法到该集合的URI。响应将是资源的集合表示,包括集合的属性以及集合的成员列表。成员的一个子集可以使用客户分页查询参数返回。

当一系列使用分页查询参数的请求是随着时间的推移获得成员的整个组时,没有需求被要求实现返回一组不变的成员。如果多个GET被用于使用分页检索一个集合，这可能导致遗漏或重复的元素。

- 对于集合的资源成员，客户端不得做出假设的URI。
- 检索集合总是包括计数属性来指定集合中的成员的总数。
- 如果由于客户指定分页查询参数或返回部分结果的服务,返回集合的一部分，那么，跨页的资源总数应该返回数量属性。

6.4.3 头

HEAD方法与GET方法的不同之处在于：它不能返回消息正文的信息。然而, HTTP头中所有相同的元信息和状态代码将被返回,就像一个GET方法的处理,包括授权检查。

- 服务可能支持HEAD方法，以返回HTTP响应头形式的元信息。
- 服务可能支持头方法,以验证链接的有效性。
- 服务可能支持头方法,以验证资源可访问性
- 服务不支持任何其他头方法的使用。
- 在缺乏资源的外部变化时，头方法应该是应幂等的。

6.4.4 数据修改请求

通过发出相应的创建、更新、替换或删除操作,或者通过在资源上调用一个Action，客户端可以创建、修改和删除资源。如果指定的资源存在但不支持请求的操作，服务返回一个状态码405。如果一个客户(4 xx)或服务(5 xx)状态码被返回,资源不得修改为操作的结果。

6.4.4.1 Update (PATCH)

PATCH方法是用于执行更新已有资源的首选方法。修改资源请求在请求正文中被发送。请求中没有指定的属性不可以被PATCH请求直接改变。在更新完成后，响应或者是空，或是一个资源的表示。对某些字段实现可能基于它自己的策略而拒绝更新操作，如果是这样,实现不适用任何更新请求。

- 服务应当支持更新资源的PATCH方法。如果资源不能被更新,应当返回状态码405。
- 在任何服务器端转换后，服务可以在响应体中返回资源的表示。
- 如果请求中的属性不能被更新（例如当一个属性是只读时）,应当返回状态码200，以及包含指定不可更新属性声明的资源表示。在这个成功的案例中,资源中的其它属性可能被更新。
- 如果客户端对一个集合指定一个PATCH请求，服务应该返回状态代码405。
- 尽管最初的ETag值可能不再匹配，在资源没有外部更改时, PATCH操作应该是等幂的。

在一个PATCH请求中,在一个JSON数组的未改变的成员可能指定为空的JSON对象。
OData标记(资源标识符、类型，etag和链接)被忽略更新。

6.4.4.2 Replace (PUT)

使用PUT方法完全替代资源。省略了从请求体的属性被重置为它们的默认值。

- 服务可能支持PUT方法来完全地替代一个资源。如果一个服务没有实现这个方法,状态码405应当被返回。
- 在任何服务器端转换后，服务可能在响应体中返回一个资源的表示，。
- 如果对于一个集合，客户端指定一个PUT请求，服务应该返回状态代码405。
- 在资源没有外部更改时, PUT操作应该幂等的。可能的例外是ETAG值可能改变并作为这个操作的结果。

6.4.4.3 Create (POST)

POST方法被用于创建一个新的资源。POST请求被提交到资源的集合，其中，新资源属于该集合。

提交一个POST请求到代表一个集合的一个资源，相当于提交相同的请求到该资源的成员属性。支持将成员添加到集合的服务应当支持两种形式。

- 服务应当支持POST方法用于创建资源。如果资源不提供被创建任何事物,应当返回状态码405。
- POST操作不应该是幂等的。

创建请求的主体包含一个被创建对象的表示。服务可以忽略任何服务控制属性(例如id),强制那些属性被服务覆盖。服务应当设置位置头（location）为新创建资源的URI。成功创建请求的响应应该是201(被创建),可能包括一个新创建资源表示的响应主体。

6.4.4.4 Delete (删除)

DELETE方法被用于删除一个资源。

- 对于可以删除的资源，服务应当支持Delete方法。如果资源不能被删除,应当返回状态码405。
- 在响应体中，服务可能返回一个刚刚删除的资源的表示。
- 对一个集合，如果客户指定一个DELETE请求，服务应该返回状态代码405。

如果资源已被删除，服务可能返回状态代码404或一个成功代码。

6.4.4.5 Actions (POST)

POST方法用于启动对象(如行动)的操作。

- 服务应当支持POST方法用于发送行动。
- POST操作可能不是幂等。

通过发送HTTP POST方法到行动的URI，一个资源可以请求自定义的行动。如果一个资源的行动属性不指定一个目标属性,那么行动的URI应当是下面的形式:

ResourceUri/Actions/QualifiedActionName

其中：

- ResourceUri是资源的URL，该资源支持调用操作。
- “Actions”是属性的名称，包含一个资源的行动，如此规范中的定义所示。
- QualifiedActionName是行动的命名空间或别名的限定名称。

一个绑定函数的第一个参数是资源，在该资源上行动被调用。其余的参数表示为请求主体的名称/值对。

客户端可以直接查询资源以确定可用的行动以及这些行动的有效参数值。一些参数信息可能需要客户端检查对应资源的Redfish 模式。

例如，一个Redfish模式文档：
http://redfish.dmtf.org/schemas/v1/ComputerSystem.xml ，在 ComputerSystem 命名空间定义了一个Reset行动，并绑定到 ComputerSystem.1.0.0.Actions 类型，例如：

```
<Schema Name="ComputerSystem">
...
<Action Name="Reset" IsBound="true">
<Parameter Name="Resource" Type="ComputerSystem.1.0.0.Actions"/>
<Parameter Name="ResetType" Type="Resource.ResetType"/>
</Action>
...
</Schema>
```

并且，一个计算机系统资源包含一个Actions属性，例如：

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

那么，对于这个行动，下面就代表一个可能的要求：

```
POST /redfish/v1/Systems/1/Actions/ComputerSystem.Reset HTTP/1.1
Content-Type: application/json
Content-Length: <computed length>
OData-Version: 4.0
{
"ResetType": "On"
}
```

6.5 响应

Redfish 定义了四种类型的响应：

- 元数据响应（Metadata Responses）：描述服务展示给一般客户的资源和类型。
- 资源响应（Resource Responses）-单独资源的 JSON表示。
- 资源集合响应（Resource Collection Responses）- 资源集合的JSON表示。
- 错误响应（Error Responses）-在一个HTTP错误例子中提供额外信息的顶级JSON响应。

6.5.1 响应标头

HTTP定义的可用于响应消息的头。下表定义了Redfish服务的那些头和他们的需求。

- 如果需求列中的值设置为“是”，Redfish 服务应能返回HTTP 1.1规范定义的下表中的头。
- 如果需求列中的值设置为“否”，Redfish 服务应能返回HTTP 1.1规范定义的下表中的头。
- Redfish 客户端应当能够理解和能够处理HTTP 1.1规范定义的下表中的所有头。

头	要求	支持的值	描述
OData-Version	是	4.0	描述了响应符合的负载的OData版本
Content-Type	是	RFC 2616,14.17节	描述了用于消息体类型的表示。application / json应当被支持。charset = utf - 8应当被支持。
Content-Encoding	否	RFC 2616, 14.17节	描述了媒体类型使用的编码
Content-Length	否	RFC 2616, 14.3节	描述了消息体的大小。一个可选的显示消息体大小的方式是使用传输编码(Transfer-Encoding):分块,其不使用内容长度（Content-Length ）头。如果一个服务不支持传输编码但需要内容长度,服务将以状态码411响应。
ETag	条件	RFC 2616, 14.19节	一个资源特定版本的一个标识符通常是一个消息摘要。帐户对象应当包括Etags。
Server	是	RFC 2616, 14.38节	需要描述一个产品令牌及其版本。多个产品令牌可能被罗列。
Link	是	见链接头（Link Header）	正如链接头章节所描述的，链接头应当被返回。
Location	条件	RFC 2616, 14.30节	表示可用于请求一个资源表示的URI。如果创建一个新的资源应当被返回。位置和X-Auth-Token应当包含在创建用户会话的响应中。
Cache - Control	是	RFC	这个头应该被支持，并用

		2616, 14.9节	于表明响应是否可以被缓存。
Via	否	RFC 2616, 14.45节	表明网络层次结构和识别消息循环。每一次通过需要插入它自己的VIA。
Max-Forwards	否	RFC 2616, 14.31节	限制网关和代理跳转。防止信息无限制的留在网络中。
Access-Control-Allow-Origin	是	W3C CORS, 5.1节	阻止或允许基于原始域的请求。用于阻止CSRF攻击。
allow	是	POST, PUT, PATCH, DELETE	GET 或 HEAD操作的返回，用于指示这个资源的其他允许操作。指定的Request URI的有效方法需要返回405(不被允许的方法)响应。
WWW-Authenticate	是	RFC 2617	需要的基本和其他可选的身份验证机制。详见[安全][#安全]部分。
X-Auth-Token	是	不透明的八位字节编码字符串	用于用户会话的身份验证。令牌值应是随机的。

6.5.1.1 链接头（Link header）

链接头提供了相应HEAD 或 GET操作中访问资源的元数据信息。除了链接资源, 资源的JSON模式的URL将返回一个rel = describedby。注释的JSON模式的URL将不返回一个rel = describedby。

HEAD应当返回连接头，并且GET和HEAD应当返回满足rel = describedby 的连接头。

6.5.2 状态码

HTTP定义状态代码,可以作为响应消息返回。

HTTP状态代码表示失败时,响应体包含一个扩展错误（extended error）资源，以提供客户更有意义和确定性的错误语义。

- 如本规范描述，当状态代码400或大于400被返回时，服务应该在响应体中返回扩展错误资源。
- 当身份验证失败，扩展错误消息不能提供特权信息

注意:扩展错误对于安全的影响，请参考安全（Security）部分。

下表列出了一些常见的HTTP状态代码。其他代码可以被服务返回（具体视情况而定）。参见状态码描述的描述列（the Description column）以及此规范要求的额外需求。

- 客户端应当理解并能够处理由HTTP 1.1规范定义的下表中的状态码，以及此规范要

求的额外需求所考虑的内容。

- 服务端应该以适当的状态码进行响应。
- 操作的异常应该被映射到 HTTP 状态代码。
- Redfish 服务不应该返回状态码 100。除了非常大的数据的上传，应该避免使用 HTTP 协议进行多路数据传输。
-

HTTP 状态代码	描述
200 OK	请求已成功完成,并且包括在它主体中的表示。
201 Created（被创建）	创建一个新资源的请求已成功完成。对于新创建的资源，位置头应设置为规范化的 URI。新创建的资源的表示可能会包含在响应体内。
202 Accepted（接受）	处理的请求已被接受,但处理还未完成。位置头应当被设置为任务资源的 URI,该任务资源稍后可以通过查询来确定操作的状态。任务资源的一个表示可能会包含在响应主体中。
204 No Content （无内容）	请求已成功,但是返回响应的主体没有内容。
301 Moved Permanently（永久的移除）	所请求的资源驻留在一个不同的 URI 之下
302 Found（发现）	所请求的资源暂时驻留在一个不同的 URI 之下。
304 Not Modified（未修改）	允许访问时，服务执行了一个有条件的GET请求。但资源内容没有被改变。使用If-Modified-Since 和/或 If-None-Match （见 HTTP 1.1, 14.25 和 14.26 节)发起条件请求。如果没有变化时，大大节省了网络带宽。
400 Bad Request（坏的请求）	无法处理请求,因为它包含缺失或无效的信息(如一个输入字段的验证错误,缺少必需的值,等等)。一个扩展的错误应当在响应的主体中被返回,如扩展错误处理（Extended Error Handling）小节中的定义。
401 Unauthorized（未授权）	这个请求包括的身份验证凭证缺失或无效。
403 Forbidden（禁用）	服务器已经认可请求中的凭证,但这些凭证不具备授权来执行这个请求。
404 Not Found（未发现）	请求指定的资源 URI 不存在。
405 Method Not Allowed（不	请求中指定的 HTTP 动词(如,DELETE, GET, HEAD, POST, PUT, PATCH)不支持这个请求 URI。响应应包括一个 Allow 头，其提供了请求 URI 识

允许的方法）	别资源所支持的方法列表。
406 Not Acceptable（未接受）	Accept 头在请求中被指定，并且被这个请求识别的资源不能生成与 Accept 头中媒体类型相一致的一个表示。
409 Conflict（冲突）	创建或更新请求无法被完成,因为在平台支持资源的当前状态下它会导致冲突(例如,试图设置以使用不相容值的链接的方式工作的多个属性)
410 Gone(用完）	所请求的资源在服务器中不再可用,也没有转发地址是已知的。这个条件是永久性的。在用户的批准之后，有链接编辑功能的客户应该删除对 Request-URI 的引用。如果服务器不知道,或者没有设备来确定“条件是否是永久性的”,应该使用状态码 404(未找到)代替。除非另有指示，这个响应是缓存的。
411 Length Required（要求长度）	请求没有使用Content-Length头(也许Transfer-Encoding:Chunked被使用)指定内容的长度。处理的资源要求Content-Length头。
412 Precondition Failed（前提条件识别）	前提条件(如果匹配或者没有被修改)检查失败。
415 Unsupported Media Type（不支持媒体类型）	请求指定一个为不被支持的主体 Content-Type。
500 Internal Server Error （内部服务错误）	服务器遇到了一个意想不到的阻止它满足请求的条件。一个扩展的错误应当在响应主体中被返回,类似于扩展错误处理（Extended Error Handling）中的定义。
501 Not Implemented(未实现）	服务器(目前)不支持实现请求所需的功能。当服务器不识别请求方法,或者服务器不能支持对于任何资源的方法时，这是适当的响应。
503 Service Unavailable （服务不可用）	由于服务器的临时超载或维护，服务器目前无法处理请求。

6.5.3 元数据响应

元数据描述资源、集合、功能和一般消费者的服务依赖行为,包括 OData 客户机工具、

对该规范没有具体理解的应用程序。如果它们已经足够了理解目标服务，客户不需要请求元数据;例如,请求和解释在本规范中定义的一个资源的 JSON 表示。

6.5.3.1 服务元数据

根据 OData-Schema，服务元数据描述了服务的顶级资源和资源类型。 Redfish 服务元数据表示为一个包含命名为“ Edmx ” 根元素的 XML 文档（在 http://docs.oasis-open.org/odata/ns/edmx” 命名空间中定义）,以及一个等于“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 引用其他模式

服务元数据应包括每个 Redfish 的资源类型的命名空间，以及“RedfishExtensions.1.0.0”命名空间。对于托管的 Redfish 模式定义，这些引用可以使用标准的 Uri (即，在 http://redfish.dmtf.org/schema 上)或与托管版本相同的 Redfish 模式的本地版本的 Url。

```
<edmx:Reference
Uri="http://redfish.dmtf.org/schemas/v1/AccountService.xml">
<edmx:Include Namespace="AccountService"/>
<edmx:Include Namespace="AccountService.1.0.0"/>
</edmx:Reference>
<edmx:Reference Uri="http://redfish.dmtf.org/schemas/v1/ServiceRoot.xml">
<edmx:Include Namespace="ServiceRoot"/>
<edmx:Include Namespace="ServiceRoot.1.0.0"/>
</edmx:Reference>
...
<edmx:Reference Uri="http://redfish.dmtf.org/schemas/v1/VirtualMedia.xml">
<edmx:Include Namespace="VirtualMedia"/>
<edmx:Include Namespace="VirtualMedia.1.0.0"/>
</edmx:Reference>
<edmx:Reference
Uri="http://redfish.dmtf.org/schemas/v1/RedfishExtensions.xml">
<edmx:Include Namespace="RedfishExtensions.1.0.0" Alias="Redfish"/>
```

</edmx:Reference>

服务元数据应包括一个定义了顶级资源和集合的实体容器。这个实体容器应当扩展在 ServiceRoot.1.0.0 模式中定义的服务容器（ServiceContainer），可能包括额外的资源或集合。

```
<edmx:DataServices>
<Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="Service">
<EntityContainer Name="Service"
Extends="ServiceRoot.1.0.0.ServiceContainer"/>
</Schema>
</edmx:DataServices>
```

6.5.3.1.2 引用 OEM 扩展

元数据文档可能参考服务使用的描述特定 OEM 扩展的额外模式文档，例如，额外集合的自定义类型。

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

6.5.3.1.3 注释

服务可以用 Redfish 定义或自定义的注释术语解释集、类型、操作和参数。这些注释通常在一个单独的注释文件中，其中，注释文件引用使用 IncludeAnnotations 指令的服务元数据文档。

```
<edmx:Reference Uri="http://service/metadata/Service.Annotations">
<edmx:IncludeAnnotations TermNamespace="Annotations.1.0.0"/>
</edmx:Reference>
```

注释文件本身指定带注释的目标 Redfish 模式元素（Target Redfish Schema element）、被应用的术语、以及术语的值:

```
<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 服务文档

对于通用 OData 客户，OData 服务文档作为一个顶级入口点。

```
{
    "@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"
},
        ...
    ]
}
```

通过使用 MIME 类型 application / JSON，OData 服务文档应当返回一个 JSON 对象。

JSON 对象将包含一个名为“@odata.context”并且有值"/redfish/v1/$metadata"的上

下文属性。该上下文告诉一个通用的 OData 客户“如何找到描述服务展示类型的服务元数据”。

JSON 对象应包括名为“value”的属性，以及每一个资源。其中，“value”属性的值是一个对于服务根包含一个入口的 JSON 数组。每一个资源是服务根的直接孩子。

每个条目表示为 JSON 对象,应包括“name”属性，“kind”属性，以及“url”属性。其中，“name”属性的值是一个资源的用户友好的名称，“kind”属性的值对于单个资源是“Singleton”(包括集合资源)或对于顶层资源集合是“Entity Set”。对于顶层资源，“url”属性的值是相对 url。

6.5.4 资源响应

通过使用 MIME 类型的 application /json，资源作为 JSON 有效负载被返回。

6.5.4.1 上下文属性

代表单个资源的响应，应当包含一个名为“@odata.context”的上下文属性。其中，“@odata.context”描述有效负载的来源。上下文属性的值应该是上下文 URL，其根据 OData-Protocol 描述资源。

在一个集合中的资源的上下文的 URL 形式:

Metadata Url#Collection[(Selectlist)]/$entity

其中:

- Metadata Url (元数据 Url) = 服务的元数据 URl (/redfish/v1/$metadata)
- Collection(集合) = 集合资源. 对于包含的资源，其包括从根集合或单个资源到限制属性的路径。
- Selectlist（选择列表） =包括在响应中的用逗号分离的属性列表，并且响应包括为表示资源定义的属性的子集。

一个顶级单资源的上下文 URL 具有的形式:

Metadata Url#SingletonName[(Selectlist)]

其中:

- Metadata Url(元数据 Url) = 服务的元数据 Url (/redfish/v1/$metadata)

- SingletonName(单独名字) =顶层单资源的名字
- Selectlist(选择列表) =如果响应包括为表示资源定义的属性的子集，那么，它包括在响应中的用逗号分离的属性列表。

6.5.4.1.1 选择列表

如果一个响应包含为一个类型在 Redfish 模式中定义的属性的一个子集，那么，上下文的 URL 应该指定包含的属性的子集。对于一个给定的资源，一个星号(*)可用于指定“所有结构属性”。

如果结果为了扩展资源而定义的属性的一个子集资源,扩大引用属性应当包含在选择列表中。

例如,下面的上下文的 URL 说明：结果包含系统资源成员集合中的单个资源。

…

"@odata.context":"/redfish/v1/$metadata#Systems/Members/$entity",

…

6.5.4.2 资源标识符属性

响应中的资源应包括一个惟一的名为“@odata.id”的标识符属性。标识符属性的值对于资源应当是独特的标识符。

资源标识符应在 JSON 的有效负载中被表示为符合 URI 路径规则(在 RFC3986 的路径,3.3 节中定义)的字符串。类似于请求 URI 的相同权限内的资源，应该根据那个规范定义的绝对路径规则进行表示。即，它们应该总是以单斜杠“/”开始。类似于请求 URI 的不同权限内的资源，应该以双斜杠“//”开始，然后是资源的授权和路径。

对于资源来说，资源标识符是规范化的 URL。根据具体情况，资源标识符可以被用来检索或编辑资源。

6.5.4.3 类型属性

在响应中，所有资源应包括命名为“@odata.type”的类型属性。类型属性的值应当是用于指定资源类型的绝对 URL，形式为：

#Namespace.TypeName

其中:

- Namespace（命名空间）=定义类型的 Redfish 模式的完整命名空间名称。对于 Redfish 资源，这将是版本化的命名空间的名字。
- 类型名称（TypeName）=资源的类型的名称。

为了检索包含资源定义的一个文档,客户端可能会使用 application/xml 内容类型发出到这个 URL 的一个 GET 请求。

6.5.4.4 ETag 属性

ETags 提供有条件地检索或更新资源的能力。资源应该包括一个名为“@odata.etag”的 ETag 属性。ETag 属性的值是一个资源的 ETag。

6.5.4.5 原始属性

根据下表，原始属性应作为 JSON 值返回。

类型	JSON 表示
Edm.Boolean	布尔
Edm.DateTimeOffset	字符串，格式化为日期时间值
Edm.Decimal	数字,可以包含小数点
Edm.Double	数字,可以包含一个小数点和一个指数
Edm.Guid	字符串，匹配模式（[0-9a-f]{ 8 } -{ 4 }[0-9a-f][0-9a-f]{ 4 } -{ 4 }[0-9a-f][0-9a-f]{ 12 }）
Edm.Int64	数量，没有小数点
Edm.String	字符串

当从客户端接收值，服务端应该在指定的 JSON 类型中支持数据的其他有效表示。在特别情况下，服务端应该支持有效的整数和十进制值，其中，十进制值采用指数符号和整数值描写，整数值包括小数点和非零的尾数。

6.5.4.5.1 日期时间（DateTime）值

根据 ISO 8601“扩展的”格式,日期时间值应作为 JSON 字符串返回，包括时间偏移量或 UTC 后缀。格式为：

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

其中，

SSS =一个或多个数字，代表秒的十进制小数,数字的数量意味着精度。
“T”分离器和“Z”后缀应当是大写。

6.5.4.6 结构化属性

定义为复杂类型或扩大资源类型的结构化属性,被作为 JSON 对象返回。JSON 对象的类型在属性的 Redfish 模式定义中被指定，其包含结构化的值。

6.5.4.7 集合属性

值集合(Collection-valued)属性作为 JSON 数组被返回,其中，数组的每个元素是一个 JSON 对象，它的类型在 Redfish 模式文档中被指定，并且描述包含的类型。

值集合属性可能包含完整集合成员的一个子集。在这种情况下，值集合属性应该用下一个连接属性进行注释。代表下一个链接的属性应当与值集合属性是对等的,并且有以“"@odata.next Link”属性作为后缀的值集合属性的名字。下一个链接属性的值应该是一个不透明的 URL，客户可以使用它去检索集合成员的下一个集合。如果请求资源的数量大于返回的资源的数量，仅下一个链接属性存在。

值集合属性应当带计数的注释。表示计数的属性是值集合属性的一个对等，并且有以“"@odata.next Link”属性作为后缀的值集合属性的名字。数的值是集合中可用成员的总数。
值集合属性不得空。在 JSON 中，空集合中返回一个空数组。

6.5.4.88 Actions（行动）属性

资源的可用行动表示为个体属性，其被嵌套在一个命名为“Actions(行动)”的资源的结构化属性之下。

6.5.4.8.1 行动表示

行动由一个属性表示，其被嵌套在“Actions”下。其中，“Actions”的名字是惟一的 URI,并用于标识该行动。这个 URI 的形式为:

#Namespace.ActionName

其中:

Namespace（命名空间）=命名空间适用于对定义行动的 Redfish 模式的引用。对于 Redfish 资源，它是版本独立的命名空间。

ActionName（动作名称）=动作的名称

在引用的与指定命名空间相关的 Redfish 模式文档中，客户端可以使用这个片段识别动作定义。

属性的值是一个 JSON 对象，包含名为“target(目标)”的属性，其值是一个相对或绝对 URL，用于调用行动。

为了给一个特定的参数指定许可值的列表，代表可用性行动的属性可能用许可值（Allowable Values）注释解释。

例如,下面的属性代表了复位动作,其被定义在计算机系统命名空间中:

```
"#Computer System.Reset": {
   "target":"/redfish/v1/Systems/1/Actions/Computer System.Reset",
   "Reset Type@Redfish.Allowable Values": [
"On",
"Force Off",
"Graceful Restart",
"Graceful Shutdown",
"Force Restart",
"Nmi",
"Force On",
"Push Power Button"
      ]
}
```

在这种情况下，客户可以用下面的正文调用一个 /redfish/v1/Systems/1/Actions/ComputerSystem.Reset 的 POST 请求:

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

6.5.4.8.2 许可值（Allowable Values）

为了给一个特定的参数指定许可值的列表，代表行动的属性可能用许可值（Allowable Values）注释解释。

许可值的集合通过包括一个属性进行指定。其中，属性的名字是"@Redfish.AllowableValues"后面参数的名字。属性的值是一个 JSON 字符串数组，代表参数的允许值。

6.5.4.9 链接属性

其他资源的引用由资源上的链接属性表示。

链接属性应当命名为“Links”,并且应当为那个类型在 Redfish 模式中定义的每个未包含（non-contained）引用属性包含一个属性。对于单值引用属性,属性值应当是单个相关的资源 id。对于值集合引用属性，属性的值应当是相关资源 id 的数组。

链接属性也包括一个 Oem 属性，用于导航特定供应商的链接。

6.5.4.9.1 一个单相关资源的引用

一个单一资源的引用作为一个 JSON 对象被返回，其包含单个 resource-identifier-property。它的名字是关系的名字,其值是引用资源的 URI。

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

6.5.4.9.2 相关资源的引用数组

零个或多个相关资源的集合的引用作为一个 JSON 对象的数组被返回。其名字是关系的名字。数组的每一个成员是一个 JSON 对象,它包含一个 resource-identifier-property，其中 resource-identifier-property 的值是引用资源的 URI。

```
{
```

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

6.5.4.10 OEM 属性

特定的 OEM 属性被嵌套在一个 OEM 属性下面。

6.5.4.11 扩展信息

响应对象可能包括扩展信息,例如，不能被更新的属性信息。这个信息被表示为一个注释，被应用于 JSON 响应的特定属性或一个完整的 JSON 对象。

6.5.4.11.1 扩展对象信息

一个 JSON 对象可以用"@Message.Extended Info"注释，用于指定对象层的状态信息。

```
{
    "@odata.context": "/redfish/v1/$metadata#Managers/Members/1/SerialInterfaces/Members/$entity",
    "@odata.id": "/redfish/v1/Managers/1/SerialInterfaces/1",
    "@odata.type": "#SerialInterface.1.0.0.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."
    }
}
```

属性的值是一个信息对象数组。

6.5.4.11.2 扩展属性信息

在一个 JSON 对象内的个别属性，可以使用"@Message.ExtendedInfo"的扩展信息去解释，并且在前面加上属性的名称。

```
{
    "@odata.context": "/redfish/v1/$metadata#Managers/Members/1/SerialInterfaces/Members/$entity",
    "@odata.id": "/redfish/v1/Managers/1/SerialInterfaces/1",
    "@odata.type": "#SerialInterface.1.0.0.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."
        }
    ],
    "Oem": {}
}
```

属性的值是一个消息对象。

6.5.4.12 额外的注释（Additional Annotations）

在 JSON 中的资源表示可以包括以属性的形式表示的额外注释，它的名字的形式是：

[Property Name]@Namespace.TermName

其中：

- PropertyName（属性名）=一个已注释的属性的名称。如果省略,注释适用于整个资源。
- Namespace（命名空间）=注释术语被定义的命名空间名字。这个命名空间必须被在请求的上下文的 url 中指定的元数据文档引用。
- TermName（术语名称）=注释术语的名称被应用于资源或资源的属性。

客户可以从服务元数据中得到注释的定义,或者完全可以忽略注释。但是，由于未被识别的注释，读资源不应该失败，包括在 Redfish 命名空间中定义的新的注释。

客户可以从服务元数据中得到注释的定义,或者完全可以忽略注释。但是，由于未被识别的注释，读资源不应该失败，包括在 Redfish 名称空间中定义的新的注释。

6.5.5 资源集合

资源集合返回一个 JSON 对象。JSON 对象应包括一个上下文、资源统计,和值数组,并且可能包括部分结果的下一个链接。

6.5.5.1 上下文属性

代表一组资源集合的响应，应当包含一个名为"@odata.context"的上下文属性，用于描述有效负载的来源。上下文属性的值应是上下文 URL，其中上下文 URL 根据 OData-Protocol 描述资源。

资源集合的上下文 URL 的形式:

MetadataUrl.#Collection[(SelectList)]

其中:

- MetadataUrl（元数据 Url） =服务的元数据 url(/redfish/v1/$metadata)
- Collection（集合）= 集合资源。对于包含的资源，这个包括从根集合或单个资源

到包容属性的路径。

- SelectList（选择列表）=如果响应包括表示资源定义的属性子集，那么响应中包含用逗号分割的属性列表。

6.5.5.2 资源计数属性

集合中可用资源的总数是通过计数属性表示。计数属性应该被命名为命名“"@odata.count",其值应是一个整数，用于代表结果中的记录总数。这个数不受$top 或$skip 查询参数的影响。

6.5.5.3 资源成员属性

资源集合的成员作为 JSON 数组被返回。代表集合成员属性的名称应该是 “value”。

6.5.5.4 部分结果

代表单个资源的响应不得分解成多个结果。

资源的集合（或资源 id）可能在多个部分响应中被返回。对于部分集合，服务包括名为“@odata.nextLink”的下一个链接属性。下一个链接属性的值应是一个不透明的 URL,客户端可以用它来检索下一组的资源。只有当被请求的资源数量大于被返回的资源数量，下一个链接才被返回。

如果客户列举了集合的所有页面，计数属性的值代表了可用资源的总数量。

6.5.5.5 额外的注释

代表资源的一个集合的 JSON 对象，可能包括被表示为属性的额外注释。其中属性名字的形式为:@Namespace.TermName

其中，

- Namespace（命名空间）=命名空间的名字，其中，注释术语被定义。这个命名空间应该被请求的上下文 URL 中指定的元数据文档进行引用。
- TermName（术语名字） =被应用于资源集合的注释术语的名称。

客户可以从服务元数据得到注释的定义,或者完全可以忽略注释。但是，由于未被识别

的注释,客户不应该舍弃读响应，包括定义在 Redfish 命名空间中的新的注释。

6.5.6 错误响应

对于确定的错误语义，HTTP 响应状态代码通常不能单独提供足够的信息。例如,如果一个客户端执行 PATCH 并且一些属性不匹配而其它则不支持时，只简单的返回一个 HTTP 状态代码 400 并不能告诉客户端哪个值是错误的。错误的响应提供客户更有意义的和确定的错误语义。

错误响应是由一个扩展的错误资源进行定义,表示为一个单独 JSON 对象并有一个名为“error”的属性，以及以下属性。

属性	描述
代码（Code）	一个字符串，从消息注册表中显示特定的消息 id（MessageId）。只有当没有更好的消息，才使用“Base.1.0.GeneralError”
信息（Message）	一个人类可读的错误消息，其与消息注册表中的消息一致。
@Message.ExtendedInfo	一个消息对象的数组，用于描述一个或多个错误消息。

```
{
    "error": {
        "code": "Base.1.0.GeneralError",
        "message": "A general error has occurred. See ExtendedInfo for more information.",
        "@Message.ExtendedInfo": [
            {
                "@odata.type" : "/redfish/v1/$metadata#Message.1.0.0.Message",
                "MessageId": "Base.1.0.PropertyValueNotInList",
                "RelatedProperties": [
                    "#/IndicatorLED"
                ],
                "Message": "The value Red for the property IndicatorLED is not in the list of acceptable
values",
                "MessageArgs": [
                    "RED",
                    "IndicatorLED"
                ],
                "Severity": "Warning",
                "Resolution": "Remove the property from the request body and resubmit the request if the
operation failed"
            },
            {
                "@odata.type" : "/redfish/v1/$metadata#Message.1.0.0.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

消息对象提供对象、属性、或错误响应的额外信息。

消息表示为 JSON 对象，有以下属性:

属性	描述
消息 id（MessageId）	字符串，表示一个特定的错误或消息(不与 HTTP 状态代码混淆)。这段代码可以用于访问消息注册表中的详细消息。
消息(Message)	人类可读的错误消息，指示相关错误的语义。这应当是完整的信息,并且不依靠替代变量。
RelatedProperties	一个可选的 JSON 指针数组，其定义消息描述的 JSON 负载中的特定属性。
MessageArgs	一个可选的字符串数组,表示消息的替换参数值。如果为一个参数化的消息而指定一个消息 id（ MessageId)，这应包括在响应中。
严重程度	一个可选的字符串，代表错误的严重性。
解决方案（Resolution）	一个可选的字符串，描述解决错误而推荐的动作

消息对象的每个实例应包含至少一个消息 id（MessageId）,连同任何适用的 MessageArgs,或一个消息属性，用于指定完整的人类可读的错误消息。

消息 id 识别在信息注册表中定义的特定消息。

消息 id 属性的值的形式

RegistryName.MajorVersion.MinorVersion.MessageKey

其中：

- RegistryName 是注册表的名称。注册表名称应当是 Pascal 大小写格式。
- MajorVersion 是一个正整数，表示注册表的主要版本
- MinorVersion 是一个正整数，代表注册表的小版本
- MessageKey 是一个到注册表的人类可读的键。消息键应是 Pascal 大小写格式,不得包括空格、时间或特殊字符。

对于相应的消息，客户端可以使用消息 id 搜索消息注册表。

信息注册表方法有国际化的优势(因为注册表可以被轻松翻译)，并且轻量级的实现(因为在实现中不需要包含大型字符串)。

7 数据模型与模式

Redfish 接口的一个关键租户是协议和数据模型的分离。本节描述常见的数据模型,资源,和 Redfish 模式需求。

- 根据资源类型定义，每个资源应被严格类型化。在一个 Redfish 模式文档中类型被定义，并且被一个独特的类型标识符所标识。

7.1 类型标识符

类型是由由一个 Type URI 识别。类型的 URI 的形式:

#Namespace.TypeName

其中，

- Namespace =定义类型的命名空间的名称
- TypeName =类型的名称

此规范定义的类型的命名空间的形式:

ResourceTypeName.MajorVersion.MinorVersion.Errata

其中：

- ResourceTypeName =资源类型的名称。对于结构化(复杂)的类型、枚举和动作,这是一般的包含资源类型的名称。
- MajorVersion =整数:在类中的一些东西，其变更是向后不兼容的。
- MinorVersion =整数:一个小的更新。新的属性可能被添加但没有被删除。对于先前的 minorversions，兼容性将被保留下来。
- Errata =整数:之前版本的一些东西被破坏了，并且需要被修复。

一个有效类型名称空间的一个例子可能是“ComputerSystem.1.0.0”。

7.1.1 在 JSON 中的类型标识符

在 JSON 负载中使用的类型应当被服务元数据定义,或被引用。

通过使用完整的(版本化)命名空间的名字，此规范中定义的资源类型应当在 JSON 文档中被引用。

注意:对于数据模型和模式的安全影响，请参考“安全”部分。

7.2 常见的命名约定

Redfish 接口是简单易读的和直观的。因此,一致性有助于对不熟悉的新发现属性的消费者理解它的用法。虽然这不能代替 Redfish 规范和 Redfish 模式中规范化的信息,但是，下面的规则提高了可读性和客户使用体验。

资源名称、属性名、枚举等常量应当是 Pascal 大小写格式

- 每个单词的第一个字母应大写，单词之间的空格应被删除(例如 PowerState，SerialNumber) 。
- 没有使用下划线。
- 两个字符的首字母缩写都应该大写(如 IPAddress， RemoteIP)
- 除了 Pascal 大小写格式标识符的第一个单词(如 Wwn,VirtualWwn)，三个或更多字符是首字母缩略词时只有第一个字符是大写。

下列情形的例外是被允许的：

- 众所周知的技术名称如“iSCSI”。
- 产品名称如“iLO”。
- 众所周知的缩写词或首字母缩写。

对于有单位或其他特殊含义的属性,单位标识符应附加到这个名字后。当前列表包括:

- 带宽(Mbps),(如 PortSpeedMbps)
- CPU 速度(Mhz),(如 ProcessorSpeedMhz)
- 内存大小(MegaBytes,MB),(如 MemoryMB)
- 条目数(Count),(如 ProcessorCount、FanCount)
- 一个资源的状态(State)(如 PowerState)。
- 正在执行并以（ing）结尾的状态值,（如 Applying, Clearing)

7.3 本地化考虑事项

数据或元数据的本地化和翻译超出了 Redfish 规范的版本 1.0 的范围。属性名不能被

本地化。

7.4 模式定义

个别资源及其从属类型和行为在 Redfish 模式文档中被定义。

7.4.1 常见的注释

所有 Redfish 类型和属性应包括描述和长描述注释。

7.4.1.1 描述

描述注释可以应用于任何类型、属性、操作或参数，用于提供一个人类可读的 Redfish 模式元素的描述。

描述注释被定义在：
http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Core.V1.xml。

7.4.1.2 长描述（LongDescription）

为了提供模式元素的一个正式和标准的规范，LongDescription 注释术语可以应用于任何类型、属性、 动作或参数。其中，Redfish 模式文件中的 LongDescription 包括“应该”的引用，服务应该与陈述一致。

LongDescription 注释术语被定义在：
http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Core.V1.xml。

7.4.2 模式文档

根据 OData-Schema，在 Redfish 模式的一个 OData 的模式表示中，个别资源被定义为实体类型。表示可能包括注释以便自动生成有能力验证 JSON 负载的 Redfish 模式的 JSON 模式表示。

OData 模式表示文档的外部元素应当是 Edmx 元素,并且有一个“4.0”值的版本属性。

```
<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>
```

7.4.2.1 引用其他模式

Redfish 模式可以引用在其他模式文档中定义的类型。在 OData 模式表示中,这是通过包含一个引用元素实现。在 JSON 模式表示中，这是通过包含一个$ ref 属性实现。

引用元素指定 OData 模式表示文档的 Uri，用于描述引用类型，并且有一个或多个孩子“包括”元素，其指定包含引用类型的命名空间的属性，以及对于那个命名空间的一个可选的“别名”属性。

对于常见类型注释术语,类型定义通常引用 OData 和 Redfish 命名空间。对于基本类型，资源类型定义引用 Redfish Resource.1.0.0 的命名空间。Redfish OData 模式表示包括测量(如，温度、速度、或维度)，其通常包括 OData 测量命名空间(OData Measures namespace)。

```
<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="OData.Measures"/>
</edmx:Reference>
<edmx:Reference Uri="http://redfish.dmtf.org/schemas/v1/RedfishExtensions.xml">
  <edmx:Include Namespace="RedfishExtensions.1.0.0" Alias="Redfish"/>
</edmx:Reference>
<edmx:Reference Uri="http://redfish.dmtf.org/schemas/v1/Resource.xml">
  <edmx:Include Namespace="Resource"/>
  <edmx:Include Namespace="Resource.1.0.0"/>
</edmx:Reference>
```

7.4.2.2 命名空间定义

资源类型被定义在 OData 模式表示的一个命名空间中。命名空间是通过一个模式元素进行定义的。该模式元素包含属性，用于声明命名空间和模式的本地别名。

OData 模式元素是 DataServices 元素的一个孩子，它是 Edmx 元素的一个孩子。

```
<edmx:DataServices>
  <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="MyTypes.1.0.0">

    <!-- Type definitions go here -->

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

7.4.3 资源类型定义

在使用 EntityType 元素的命名空间内定义资源类型。Name 属性指定资源的名称，并且如果有的话 BaseType 指定基础类型。

Redfish 资源来自于 Resource.1.0.0 命名空间的一个名为“Resource”的通用资源基础类型。

EntityType 包含属性和引用属性元素，它定义资源,以及描述资源的注释。

```
<EntityType Name="TypeA" BaseType="Resource.1.0.0.Resource">
  <Annotation Term="Core.Description" String="This is the description of TypeA."/>
  <Annotation Term="Core.LongDescription" String="This is the specification of TypeA."/>

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

</EntityType>
```

所有的资源都应包括 Description 和 LongDescription 注释。

7.4.4 资源属性

通过使用 Property 元素,资源的结构属性被定义。Name 属性指定属性的名称,以及 Type 的类型。

必须有一个非空值的 Properties 包括 nullable 属性，并且值为“false”。

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

所有的属性都应包括 Description 和 LongDescription 注释。
只读的 Properties 使用有 ODataPermissions/Read 值的 Permissions 注释进行解释。

所有服务要求实现的 Properties 使用必须的注释进行解释。
有单位与其关联的 Properties 可以使用单位注释进行解释。

7.4.4.1 属性类型

属性类型由 Type 属性指定。类型属性的值可能是一个基本类型,结构化类型,枚举类型或基本类型集合,结构化类型集合或枚举类型集合。

7.4.4.1.1 基本类型（Primitive Types）

基本类型使用"Edm"命名空间前缀作为前缀。
Redfish 服务可能使用下列任何基本类型:

类型	意义
Edm.Boolean	真或假
Edm.DateTimeOffset	带有时区的日期和时间
Edm.Decimal	有固定精度和小数位数的数值
Edm.Double	IEEE 754 二进制的 64 浮点数(小数位数 15 - 17)
Edm.Guid	全局唯一标识符
Edm.Int64	有符号的 64 位整数
Edm.String	utf - 8 字符的序列

7.4.4.1.2 结构化类型（Structured Types）

在使用 ComplexType 的元素的命名空间内定义结构化的类型。复杂类型的 Name 属性指定了结构化类型的名称。如果有的话，复杂类型可以包括一个 BaseType 属性来指定基本类型。
结构化类型可以在不同资源类型的不同属性之间重用。

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

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

      </ComplexType>
```

结构化类型可以包含属性、引用属性和注释。
结构化类型应包括 Description 和 LongDescription 注释。

7.4.4.1.3 枚举

在使用 EnumType 元素的命名空间内定义枚举类型。枚举类型的 Name 属性指定了枚举类型的名称。

Enumeration 类型可以在不同资源类型的不同属性之间重用。

EnumType 元素包含 Member 元素，用于定义枚举的成员。Member 元素包含一个 Name 属性用于指定成员名字的字符串值。

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

枚举类型应包括 Description 和 LongDescription 注释。

Enumeration 成员应包括 Description 注释。

7.4.4.1.4 集合

类型属性可以指定一个基本,结构化或枚举类型的集合。对于集合属性，类型属性值的形式为:

Collection(NamespaceQualifiedTypeName)

其中，NamespaceQualifiedTypeName 是基本的、结构化的、枚举类型的命名空间限定名。

7.4.4.2 附加属性

AdditionalProperties 注释术语被用于指定：是否一个类型可以包含那些定义之外的附加属性。有 AdditionalProperties 注释并且值为“False”的类型不能包含附加属性。

附加属性注释术语定义在

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

7.4.4.3 非空属性

属性可能包含有 false 值的 Nullable 属性，用于指定该属性不能包含空值。相反，一个包含值为“true”的 nullable 属性的属性,或没有 nullable 属性,可以接受 null 值。

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

7.4.4.4 只读属性（Read-only properties）

为了指定属性是只读的,Permissions 注释术语可以应用于有 OData.Permissions/Read 值的属性。

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

Permissions 注释术语被定义在 http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Core.V1.xml。

7.4.4.5 必需的属性（Required Properties）

Required 注释或 Nullable 属性被用来指定：一个属性必须被服务支持。必需的属性应该使用 Required 注释进行解释，或者使用有 false 值的 Nullable 属性进行解释。所有其它属性是可选的。

如果一个实现支持一个属性,它总是为那个属性提供一个值。如果一个值是未知的,那么在大多数情况下，null 是可以接受的值。GET 操作不返回的属性应该表明：当前的属性不能被实现支持。

<Annotation Term="Redfish.Required"/>

必需注释术语定义在 http://redfish.dmtf.org/schemas/v1/RedfishExtensions.1.0.0。

7.4.4.6 创建时所需的属性（Required Properties On Create）

RequiredOnCreate 注释术语是用来指定：在创建资源时必须被指定的一个属性。没有

RequiredOnCreate 注释的属性,或用一个值为“false”的布尔属性的注释时，在创建资源时不需要指定一个属性。

<Annotation Term="Redfish.RequiredOnCreate"/>

RequiredOnCreate 注释术语定义在 http://redfish.dmtf.org/schemas/v1/RedfishExtensions.1.0.0。

7.4.4.7 计量单位（Units of Measure）

除了以下命名约定,为了指定属性的计量单位，代表计量单位的属性应当使用 Units 注释术语进行注释。

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

Unit 注释术语被定义在 http://docs.oasis-open.org/odata/odata/v4.0/os/vocabularies/Org.OData.Measures.V1.xml。

7.4.5 引用属性(Reference Properties)

通过使用 NavigationProperty 元素，引用其它资源的属性被描述为引用属性。NavigationProperty 元素指定相关资源的 Name 和命名空间限定类型。

如果属性引用一个单一类型,类型属性的值是相关资源类型的命名空间限定名。

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

如果属性引用资源的集合,type 属性的值的形式为:
Collection(NamespaceQualifiedTypeName)
其中，NamespaceQualifiedTypeName 是相关资源类型的命名空间的限定名。

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

所有引用属性应包括 Description 和 LongDescription 注释。

7.4.5.1 包含资源（Contained Resources）

引用属性的成员被引用资源包含。引用属性被指定为值为“true”的 ContainsTarget 属性。

例如,为了指定一个 Chassis 资源包含 Power 资源,您将在资源的属性上指定 ContainsTarget = true，用于代表 Chassis 类型定义内的 Power Resource。

```
        <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>
```

7.4.5.2 扩展的（Expanded）引用

在 Redfish JSON 负载中的引用属性被扩展用于包括相关资源 id 或相关资源 id 的集合。通过使用 AutoExpandReferences 注释，这个行为被表示。

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

AutoExpandReferences 注释术语被定义在 https://tools.oasis-open.org/version-control/browse/wsvn/odata/trunk/spec/vocabularies/Org.OData.Core.V1.xml。

7.4.5.3 扩展的（Expanded）资源

这个术语可以被应用于引用属性,为了指定：在响应中，服务的默认行为将要扩展相关的资源或服务集合。

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

AutoExpand 注释术语被定义在

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

7.4.6 资源行为

Actions 被分组在一个名为“Actions”的属性下。

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

Actions 属性的类型是一个结构化类型，以及一个单一 OEM 属性的类型。它的类型是一个结构化类型并且没有定义的属性。

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

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

单个行为是在使用Action元素的命名空间内定义的。动作的Name属性指定动作的名称。IsBound 属性指定：动作被绑定到(作为一个成员出现)一个资源或结构化类型。

Action 元素包含一个或多个 Parameter 元素，其指定每个参数的 Name 和 Type。

第一个参数是被称为“绑定参数”，并指定资源或结构类型，其中，动作作为成员出现(资源上的 Actions 属性的类型)。其余 Parameter 元素描述被传递到动作的额外的参数。

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

7.4.7 资源的可扩展性（Resource Extensibility）

通过使用资源、链接和动作上的 Oem 属性，公司、Oem 和其他组织为通用的 Redfish 资源定义了额附件属性,链接,动作。

虽然这些扩展的信息和语义是此标准之外,代表数据、资源本身、协议周围语义的模式应当符合本规范的要求。

7.4.7.1 Oem 属性（Oem Property）

在本节的上下文中,术语“OEM”指的是任何公司、制造商、或组织，其为 DMTF 已发行

的模式和 Redfish 功能提供或定义的一个扩展。Redfish 特定的资源的基本模式包括被称为“Oem”的空的复杂类型属性,其值可以用来封装一个或多个 OEM 特定的复杂属性。在标准 Redfish 模式中的 Oem 属性是一个预定义的占位符,用于定义 OEM 特定的属性。

Oem 属性的正确使用需要定义可以在 Oem 属性中引用的 OEM 特定的复杂类型的元数据。下面的片段是 XML 模式的一个例子,它在复杂类型"AnvilType1"下，定义了一对 OEM 特定的属性。(为了简化示例，代表性的其他模式元素（比如 XML 和 OData 模式描述标识符）没有被显示)。

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

下一个片段显示了一个示例，说明之前的模式和“AnvilType1”属性类型如何出现在 Oem 属性的实例化中作为资源上 GET 结果。在 Oem 属性的使用中，例子展示了两个必需的元素：对象的名字、对象的类型。这些元素的详细要求在以下部分中提供。

```
...
  "Oem": {
    "Contoso": {
      "@odata.type": "http://Contoso.com/schemas/extensions.v.v.v#contoso.AnvilType1",
      "slogan": "Contoso anvils never fail",
      "disclaimer": "* Most of the time"
    }
  }
...
```

7.4.7.2 Oem 属性格式和内容

OEM 特定的对象包含在 Oem 属性中，其必须是有效的 JSON 对象,并遵循 Redfish 复杂类型的格式。对象(属性)的名称唯一地标识 OEM 或组织，其中，组织管理定义属性的命名空间的顶部。这在下一节中详细描述。OEM 特定的属性还包括一个类型属性，用于提供模式的位置，以及在这个模式中的属性的类型定义。Oem 属性可以同时拥有多个 OEM 特定的对象,包括多于一个公司或组织的对象。

包含在 OEM 特定的复杂类型中的任何其他属性的定义,连同对那个内容的功能规范、验证、或其他要求，是 OEM 特定的而且超出了本规范的范围。尽管在一个 OEM 特定的 JSON 对象中的 OEM 特定的元素的大小或复杂性方面，没有 Redfish 特定的限制,其目的是 OEM 属性通常只用于少量简单的属性,从而增加 Redfish 资源。如果大量的对象或大量的数据(与 Redfish 资源的大小相比)被要求支持,OEM 应该考虑 OEM 特定的对象用于指向它们扩展的一个单独的资源。

7.4.7.3 Oem 属性命名

对于在其下面定义属性的顶级命名空间，在 Oem 属性中的 OEM 特定的对象命名使用一个独特的 Oem 标识符进行命名。有两个指定的标识符的形式。标识符应是一个 ICANN 公认的域名（包括顶级域名后缀），或一个以"EID:"开始的 IANA 分配的企业号码。

使用'.com'域名的组织可能省略'.com'后缀(例如 Contoso.com 可能使用“Contoso”，但 Contoso.org 必须使用“Contoso.org”作为他们的 OEM 属性名)。 一个 OEM 标识符的域名部分应当被认为是大小写无关的。也就是说，文本“Contoso.biz”、“contoso.BIZ”、“conTOso.biZ”，等等识别相同的 OEM 和顶级命名空间。

属性名的 OEM 标识符部分可以后跟一个冒号和任何附加字符串，从而允许进一步对 OEM 特定的对象执行命名空间（如 OEM 所需）。例如，“Contoso.com:xxxx”或"EID:412:xxxx"。冒号之后的任何文本的形式和意义是完整的、特定的 OEM。OEM 特定的扩展后缀可能是大小写敏感的，这取决于 OEM。通用客户端软件应该处理这样的扩展（如果存在，是不透明的），而不是试图解析和解释的内容。

根据 OEM 的需要，这个后缀可以很多种用法。例如，Contoso 公司可能有子组织"Research "，在这种情况下，OEM 特定的属性名可能被延伸到"Contoso:Research"。另外，它可以被用来识别对于功能区域、地理、子公司等的命名空间。

名字的 OEM 标识符部分一般会确定创建并维护属性模式的公司或组织。然而，这不是必须的。标识符仅仅被要求去识别一个顶级命名空间管理者的独特部分，从而阻止不同供应商或组织的 OEM 属性定义之间的冲突。因此，顶级命名空间的组织与提供 OEM 特定属性定义的组织可能会是不同的。例如，Contoso 可能允许他们的客户之一（如，“CustomerA”） 用某些 CustomerA 专有属性延长 Contoso 产品。在这种情况下，尽管 Contoso 分配“contosos:customers.CustomerA"的名字，但是它可能是 CustomerA 在命名空间下面定义了内容和功能。在所有情况下，OEM 标识符不应被使用，除非公司和组织允许和声明可以使用。

7.4.8 Oem 属性示例

以下片段给出了当访问资源时可能会用到一些命名的例子以及 Oem 属性的用法。例子表明：OEM 的标识符可以是不同的形式，OEM 特定内容可以是简单或复杂的，OEM 标识符的扩展格式和使用是 OEM 特定的。

```
        "targetSetting" : "rabbit"
      }
    }
  ...
```

7.4.8.1 自定义动作

通过定义绑定到资源行动属性类型的 OEM 属性，特定 OEM 行动能够被定义。

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

</Schema>
```

这种绑定动作出现在 JSON 负载中，并作为 Oem 类型的属性，被嵌套在一个动作属性的下面。

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

7.4.8.2 定制化的注释

这个规范定义了一组常见的注释，用于扩展 Redfish 所使用的资源类型的定义。此外，服务可以定义自定义注释。

为了提供服务特定的（service-specific）的关于类型的信息,服务可以应用注释到资源上，如服务是否支持特定属性的修改。

服务可以应用注释到现有的资源,其中，那些资源没有为注释定义一个值。服务不能改变一个注释的值，其中，注释被应用作为资源定义的一部分。

因为服务注释可能被应用到现有的资源定义,它们通常在服务元数据引用的特定服务元数据文档中被指定。

7.5 常见的 Redfish 资源属性

本节包含一组所有 Redfish 资源的共同属性。本节中的属性名称不得用于任何其他目的,即使他们没有在一个特定的资源中被实现。

常见的属性被定义在基础资源 Redfish 模式中。对于 OData 模式表示是在 Resource.xml 中。JSON 模式表示在 Resource.1.0.0.json 中。

7.5.1 Id

资源的 Id 属性识别一个集合中的资源。一个集合中，Id 值应该是唯一的。

7.5.2 名字

Name 属性用于传达资源的一个人类可读的绰号。Name 属性类型是字符串。在一个集合中，多个资源实例的名字值可以不是唯一的。

7.5.3 描述

Description 属性用于传递资源的一个人类可读的描述。Description 属性类型应该是字符串。

7.5.4 状态

Status 属性代表一个资源的状态。

状态属性的值是一种常见的状态对象类型，如规范中的定义。通过有状态的一个共同的表示,客户可以依赖一致的语义。状态对象能够指示当前预期的状态、资源要求改变的状态、当前的实际状况、以及影响资源的当前状态的任何问题。

7.5.5 链接

Links 属性代表了与资源相关的连接（如资源模式定义中的定义）。为一个资源定义的所有相关的参考属性应当嵌套在连接属性下面。资源定义的所有直接(下属)引用的属性应当是资源的根。

7.5.6 相关项（Related Item）

Related Item 属性代表资源的链接(或资源的一部分)，如该资源模式定义。与其他引用类似，这不是为了成为一个强大的连接方法。相反,它是用来显示服务的不同部分的元素或子元素之间的一个关系。例如,因为 Fans 可能在一个领域实现而处理器另一个地方,Related Item 被用来通知客户端,这两个是相关的(在这种情况下, Fan 是冷却处理器)。

7.5.7 行动（Actions）

Actions 属性包含资源支持的行动。

7.5.8 OEM

OEM 属性用于 OEM 扩展，如模式可扩展性（Schema Extensibility）中的定义。

7.6Redfish 资源

统称为 Redfish 模式,资源描述的集合包含符合这个规范的实现上的规范的要求。

Redfish 资源是多个通用种类之一。

- 根服务资源
  - 对于可应用的对象资源，包含一个特定的服务实例的映射。
  - 包含一个服务实例的 UUID。这个 UUID 与通过 SSDP 发现返回的 UUID 是相同的。
- 当前配置资源,包含下面的混合:
  - 库存(静态和只读)
  - 健康遥测(动态和只读)
  - 当前配置设置(动态和读/写)
  - 当前的度量值
- 设置资源
  - 动态的、读/写、等待配置设置
- 服务
  - 公共服务如事件、任务、会话
- 注册中心资源
  - 对于事件和消息注册的静态的、只读的 JSON 编码信息

7.6.1 当前配置

当前配置资源代表当前状态和资源配置的服务的知识。这可能是使用 PATCH 直接可更新

的或它可能是由客户只读，并且客户必须 PATCH 一个单独的设置资源。

7.6.2 设置

设置资源代表资源未来的状态和配置。这个属性总是与通过 Redfish.Settings 注释的资源相关。其中，资源代表当前的状态,设置资源代表着未来预期的状态。资源的状态直接被改变（如 POST 一个动作或 PUT 请求），或者间接被改变（比如当用户重新启动 Redfish 服务之外的机器）。

7.6.3 服务

服务资源代表 Redfish 服务本身以及相关资源的组件。虽然完整的列表只有通过遍历 Redfish 服务树才能发现，但是，列表包括的服务有 Eventing 服务、任务管理和会话管理等。

7.6.4 注册（Registry）

Registry 资源是可以协助客户解释 Redfish 模式定义以外的 Redfish 资源的那些资源。注册的例子包括：消息注册、事件注册和枚举注册,如用于 BIOS 的注册。在注册中,一个标识符用于检索给定资源、事件、消息或其他项的更多信息。这可以包括其他属性、属性限制等。注册是它们自己的资源。

7.7 特殊资源的情况（Special Resource Situations）

出现特定类型资源的某些情况下,需要表现出共同的语义行为。

7.7.1 缺少资源（Absent Resources）

当客户端请求该资源的信息时，资源可能会缺少或状态未知。为了删除 URI 预期保持不变(如，删除一个风扇）的资源时，资源应该表示状态对象的状态属性为"Absent"。在这种情况下，没有已知值的任何要求或支持的属性应表示为 null。

7.7.2 模式变化(Schema Variations)

一些情况下，与出版的 Redfish 模式的偏离是必要的。一个例子是 BIOS，其中，在可用的配置设置中，不同的服务器可能有较小的变化。提供者可能构建一个模式，该模式是一个个体实现的超集。为了支持这些变化，在当前配置对象中，Redfish 支持定义在类模式的省略参数。以下规则适用:

- 通过在 Setting Data Apply 状态架构中将其设置为例外，所有 Redfish 服务必须支持设置 Setting Data 中不支持配置的元素的尝试（attempts），但并不是使整个配置操作失败。
- 在当前配置对象中，一个资源的特定属性的支持是通过其属性的表现进行通知。如果元素从当前配置中缺失，客户可能假设：在那个资源上元素不被支持。
- 对于 ENUM 配置项，其可能在允许值中有变化，一个特殊的只读的能力元素将被添加到当前配置中，用于对元素指定限制。这是一个覆盖的模式，只在必要时才能使用。

提供者可能分裂模式资源成单独的文件（如 Schema + String Registry），每个都有一个独立的 URI 和不同的内容编码。

- 通过当前配置对象(如果适用)，资源可能从出版模式中传达省略。

8.服务细节

8.1 事件

本节涵盖了基于 REST 的机制订阅和接收事件消息，。

Redfish 服务需要一个客户端或管理员创建接收事件的订阅。当管理员发送一个 HTTP POST 消息到订阅资源的 URI 时，一个订阅被创建。这个请求包含一个事件接收者的客户端期望事件被发送到的 URI 以及事件的类型。那么，当服务端中的一个事件被触发时，Redfish 服务将发送一个事件到那个 URI。

- 对所有可以发送事件的资源，服务应当支持“push”类型事件。
- 服务不得“push”事件(使用 HTTP POST),除非创建一个事件订阅。通过删除订阅，客户端或服务在任何时候都可以终止事件流。如果交付错误的数量超过预先配置的阈值，服务可能删除订阅。
- 服务应该对成功的订阅以 HTTP 状态 201 进行响应，并且设置 HTTP 位置头为一个新的订阅资源的地址。订阅是持久的，并且可以在事件服务重启后保留。
- 通过发送一个 HTTP DELETE 消息到订阅资源的 URI，客户应当终止订阅。
- 通过发送一个特殊的“subscription terminated”事件作为最后一条消息，服务可能终止订阅。未来相关的订阅资源的请求将使用 HTTP 状态 404 作为响应。

在 Redfish 服务（生命周期和警报）中生成两种类型的事件。

当资源被创建、修改或销毁时，生命周期事件发生。并不是每一个资源的修改将生成一

个事件（这个似于 etag 被改变）；并且对于每个资源改变实现不能发送一个事件。例如，如果对于每一个 Ethernet 数据包被接收或每一次传感器改变了 1 度，就发送一个事件，这可能导致事件太多而不适合可扩展的接口。这个事件通常表示资源被改变了，以及（可选地）任何属性被改变了。

当一个资源需要表示某种意义的事件时，警报事件发生。这可能与资源是直接或间接相关。事件的这种风格通常采用消息注册方法，类似于在那个包含 Message Id 中处理的扩展的错误。这类事件的例子包括：一个机箱被打开、按钮被按下、电缆未插入或者超过阈值。这些事件通常不能很好地反映生命周期类型事件，因此他们有自己的类别。

注意:对于事件的安全影响，请参考安全部分章节。

8.1.1 消息订阅事件

通过遍历 Redfish 服务接口，客户可以定位事件服务。为了在请求事件的事件服务中订阅消息，当事件服务已经被发现时，通过发送一个 HTTP POST 到集合的 URL，客户订阅消息。这应该是被发现了的根服务，如那个服务的 Redfish 模式中描述所示。

在 Redfish 模式中的订阅主体的特定语法。

如果成功，“subscribe”行动应当返回 HTTP 状态 201 (创建)，并且响应中的位置头应当包含一个 URI。该 URI 可以提供新创建的"subscription"资源的位置。如果有的话,响应的主体应包含订阅资源的表示。发送一个 HTTP GET 到订阅资源，应该返回订阅的配置。

一旦订阅已经被服务注册,客户开始接收事件，并且不追溯接收事件。服务不保留历史事件。

8.1.2 事件消息对象

POST 到指定的客户端端点的事件消息对象，应当包含属性，如 Redfish 事件模式中的描述。

这个事件消息结构支持消息注册。在信息注册方法中，有一个消息注册，其有一个列表或 Message Ids 的数组，并使用众所周知的格式。这些 Message Ids 在本质上是简洁的,因此他们远小于实际的消息,使他们适合于嵌入式环境。在注册表中,还有一个消息。对于严重程度和推荐的行为，消息本身可以有参数以及默认值。

消息 Id 属性内容的形式

RegistryName.MajorVersion.MinorVersion.MessageKey

其中：

- RegistryName 是注册表的名称。注册表名称应当是 Pascal 大小写格式（Pascal-cased）。
- MajorVersion 是一个正整数，用于表示注册表的主要版本
- MinorVersion 是一个正整数，用于表示注册表的小版本
- MessageKey 是注册中的一个人类可读的键。消息键应是大小写相关,不包括空格、时间或特殊字符。

8.1.3 订阅清理（Subscription Cleanup）

取消与这个订阅相关的消息时，客户端或管理员只需发送一个 HTTP DELETE 请求到订阅资源的 URI。

这些是一些全局设置的配置属性，这些全局设置为所有事件订阅定义了行为。
为了可配置服务的行为参数的细节，参见在事件服务 Redfish 模式中定义的属性。

8.2 异步操作(Asynchronous Operations)

支持异步操作的服务将实现任务服务（Task service）和 任务资源（ Task resource）。

任务服务用于描述处理任务的服务。它是包含零个或多个任务资源的集合。任务资源用来描述一个长时间运行的操作。当一个请求将超过几秒钟时，这种情况就会发生，例如当服务被实例化时。如果它是成功的，客户会查询任务资源的 URI，以确定何时操作已经完成。

Redfish 模式的任务结构包含任务的准确结构。它所包含的信息类型是：开始时间、结束时间、任务方式、任务状态，以及零个或多个与任务相关的信息。

每个任务有很多可能的状态。正确的状态和它们的语义被定义在 Redfish 模式的任务资源中。

当客户端发出长时间运行操作的请求,服务返回一个 202 状态(接受)。

有状态码 202(接受)的任何响应，应包括一个位置头，用于包含监控任务的 URL，并且可能包括一个等待头用于指定时间的数量。其中，时间的数量是指操作的查询状态之前，客户应该等待的时间。

当执行一个到状态监测的 GET 请求时，客户端在接受头不应该包括 mime 类型的 application/http。

202(接受)的响应体应该包含描述任务状态的任务资源的实例。

只要操作在处理过程中,当查询状态监控在位置头中被返回时，服务将继续返回一个状态码 202(接受)。

如果是同步完成，一旦操作完成后状态监控应返回一个状态码 OK(200)，并且包括初始操作的头和响主体。如果最初的操作导致了错误,响应的主体应当包含一个错误的响应。

如果操作和服务已经删除任务完成，服务可能返回一个状态码 410(Gone)。

通过直接查询任务资源，客户机可以继续获得状态信息。其中，查询任务使用在 202(接受)响应的主体返回的资源标识符。

- 支持异步操作的服务应当实现任务的资源
- 异步操作的响应应返回一个状态码 202(接受)和设置 HTTP 响应头“Location”到与活动相关联的状态监控的 URI。响应还可能包括一个等待头,用于指定客户端轮询状态之前应该等待的时间。响应的主体应该包含一个 JSON 中任务资源的表示。
- 任务监控或任务资源的 GET 请求，应当返回没有阻塞操作的当前状态。
- 使用 HTTP GET、PUT、PATCH 的操作应该是同步的。
- 通过使用 HTTP DELETE 和 HTTPPOST 方法，客户应为请求准备处理同步和异步响应。

8.3 资源树的稳定性（Resource Tree Stability）

在实现中，资源树被定义为一组 URI 和数组元素。在跨设备重启和交流电源周期中保持一致的单一服务而且必须承受合理的配置更改(例如,添加一个适配器到服务器)。一个服务的资源树在跨设备实例方面是不一致的。客户端必须查看数据模型和发现与它们进行交互的资源。一些资源可能在系统之间保持非常稳定(例如 BMC 网络设置)——但这并不是一个架构方面的保证。

- 在跨服务重启和次要设备配置更改时，资源树应该保持稳定。因此，URI 的集合和数组元素的索引应该保持不变。
- 对于资源树来说，服务实例之间的客户拥有一致性，并没有被期待。

8.4 发现

支持Redfish可扩展的平台管理API的管理设备的自动化发现是通过使用简单服务发现协议(Simple Service Discovery Protocol ，SSDP)实现的。对于网络有效性的发现，这个

协议是被允许的，并且不用依赖于 ping-sweeps、路由器表搜索、或限制性的 DNS 命名方案。SSDP 的使用是可选的。如果实施,通过管理器网络服务的资源，SSDP 的使用应当允许用户禁用协议。

当发现的目标是为客户端软件来定位符合 Redfish 管理设备时,主 SSDP 功能整合是 M-SEARCH 查询。Redfish 也遵循 SSDP 扩展，以及在适用情况下被 UPnP 所使用的名字，例如，符合 Redfish 系统还可以没有冲突的实现 UPnP。

8.4.1 UPnP 兼容性（Compatibility）

对于通用目的 SSDP 客户端软件的兼容性,主要的 UPnP、TCP 端口 1900 应该被所有 SSDP 流量所使用。此外，对于 SSDP 多播消息的 Time-to-Live (TTL)跳数设置，应该默认为 2。对于 UPn P 根设备(用 NT:upnp:rootdevice),建议设备以适当的描述符和 XML 文档响应 M-SEARCH 查询。

8.4.2 USN 格式

USN 领域提供的 UUID 应当等于根服务的 UUID 属性。如果有多个/冗余的管理者，不考虑冗余故障转移时，UUID 应当保持静态。惟一的 ID 应是规范化 UUID 格式,随后是“::dtmf-org”。

8.4.3 M-SEARCH 响应

从 AL 点的客户到 Redfish 服务根 URI，对于 Redfish 服务的搜索目标，管理设备必须响应M-SEARCH查询搜索。对于 "ssdp:all"的搜索目标类型,Redfish设备还应回应M-SEARCH查询。

Redfish 服务根搜索目标(ST)： URN:dmtf-org:service:redfish-rest:1

回复中 URN 应使用“redfish-rest:”服务名称，其后是 Redfish 规范的主要版本。如果 Redfish 规范(它的服务与其符合)的小版本是一个非零值,并且该版本是与先前的小修订版本是向后兼容的,那么应当附加小版本,并且在版本之前增加一个冒号。例如,符合 Redfish 规范版本“1.4” 的一个服务，应该使用“redfish-rest:1:4”的服务作为回复。

回应 M-SEARCH 组播或单播查询的一个例子应当遵循如下所示的格式。对于特定于设备的值，在括号中的字段是占位符。

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

8.4.4 通知,活着（Alive）,和关闭的消息

Redfish 设备可以实现被 UPnP 定义的额外的 SSDP 消息，从而宣布对于软件的可用性。如果实现,这种能力必须允许最终用户从 M-SEARCH 响应功能分别地禁用流量。这允许用户以产生的最小网络流量情况下使用发现功能,。

9 安全

9.1 目标

✧ 监视和管理的特权模式:

- ■ 系统设置
  - ✓ BIOS 配置
  - ✓ 系统电源状态
  - ✓ 传感器信息(功率/温度/健康)
  - ✓ 网络设置
  - ✓ 存储设置
  - ✓ 日志
- ■ Redfish 服务配置
  - ✓ 账户管理
  - ✓ 网络设置
  - ✓ 日志
- ■ 固件版本
- ■ OEM 特定供应商的特性和功能

✧ （Permission/ authorization）批准/授权模型在 Redfish 兼容设备之间应该是一致的

- ■ 对于 permission/ authorization 模型定义一个最低基准

✧ 基础设施的认证

- ✧ CURL 的兼容性
- ✧ 自动化客户端
- ✧ 嵌入式服务处理器

9.2 协议

9.2.1 TLS

实现应当支持 TLS v1.1 或更高版本

9.2.2 密码套件

实现应该支持 TLS 套件的基于 AES - 256 的密码。

Redfish 实现应该考虑支持密码（类似于下面的内容）。该密码不需要使用信任的证书，就可以执行身份验证和识别。

```
TLS_PSK_WITH_AES_256_GCM_SHA384
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384
TLS_RSA_PSK_WITH_AES_256_GCM_SHA384
```

使用以上推荐密码的额外优势是：

“AES-GCM 不仅是有效的和安全的,而且，硬件实现可以实现高速度、低成本和低延迟,因为模式可以被安排。”

参考 RFCs

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

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

9.2.3 证书

如果提供一个证书,实现应当支持默认证书的替换，证书有至少 4096 位 RSA 密钥和 sha512-rsa 签名。

9.3 认证

- 身份验证方法

服务应当支持“基本身份验证”和“Redfish 会话登录身份验证”(如下会话管理部分所述)。当使用基本身份验证时，服务不需要客户端创建一个会话。

服务可以实现其他身份验证机制。

9.3.1 HTTP 头安全

- 所有写活动应被验证,例如, POST, PUT/PATCH, 和 DELETE，除了：
  - 对于身份验证需要的会话服务/对象的POST操作
    - 当身份验证失败发生时，扩展错误信息不得提供特权信息
- REST对象不能用于无认证，除了：
  - 根对象，其需要识别设备和服务位置
  - $metadata对象，其需要检索资源类型
  - OData服务文档，其需要与OData的客户兼容
  - 位于/Redfish的版本对象
- 通过外部引用的外部服务链接不是本规范的一部分,并且可能有其他安全要求。

9.3.1.1 HTTP 重定向（Redirect）

- 当有一个HTTP重定向时，目标资源的权限要求应当是强制的。

9.3.2 扩展错误处理

- 当身份验证失败发生时，扩展的错误消息不应该提供特权信息

9.3.3 HTTP 头身份验证 (HTTP Header Authentication)

- 进行身份验证的HTTP头应当优先于其它头被处理。其中,其它头可能影响响应，例如:etag,If-Modified等等。
- HTTP cookie不得用于任何活动的身份验证，例如，:GET、POST、PUT/PATCH

和DELETE。

9.3.3.1 基本身份验证（BASIC authentication）

HTTP基本身份验证（如RFC2617中的定义）应当被支持，并且在任何第三方身份验证服务和客户之间只使用兼容的TLS连接来传输数据。

9.3.3.2 请求/消息级别认证（Request / Message Level Authentication）

建立一个安全通道的每个请求应伴随着一个身份验证头。

9.3.4 会话管理（Session Management）

9.3.4.1 会话的生命周期管理（Session Lifecycle Management）

会话管理是留给Redfish服务的实现。这包括孤儿会话超时和同步打开会话的数量。

- **Redfish服务应提供符合这个规范的登录会话。**

9.3.4.2Redfish 登录会话

对于需要多个Redfish操作的功能，或出于安全原因，客户可能通过会话管理接口创建Redfish登录会话。建立一个会话的URI可以在会话服务的会话属性或会话属性下服务根的连接会话中被发现。两个URI都应当是相同的。

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

9.3.4.3 会话登录

对于会话服务的会话集合资源，一个Redfish会话被一个HTTP POST创建,包括以下POST主体:

```
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
OData-Version: 4.0

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

对于这个会话创建，Origin头应该被保存，并且通过使用这个会话去验证请求，从而与后续请求可比。其中，请求是从一个授权客户域中被发起。

创建一个会话的POST请求的响应包括:

- 一个X-Auth-Token头包含一个“session auth token”,其中,“session auth token”表明：客户端可以使用一个后续请求,和
- 一个位置头，其中包含一个到新创建的会话资源的链接。
- JSON响应主体，包含一个完整的新创建的会话对象的表示:

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

{
    "@odata.context": "/redfish/v1/$metadata#SessionService/Sessions/$entity",
    "@odata.id": "/redfish/v1/SessionService/Sessions/1",
    "@odata.type": "#Session.1.0.0.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"Session Auth Token"，以及位置头返回的链接。"Session Auth Token"是用来验证后续请求,通过设置请求头"X-Auth-Token”与从登陆POST接收的"Session Auth Token"。客户端之后将使用POST位置头返回的链接，进行注销或终止会话。

注意，"Session ID" 和 "Session Auth Token"是不同的。Session ID惟一地标识会话资源和返回响应数据以及位置头链接的最后一段。具有足够权限的一个管理员可以查看活跃

会话，并使用相关sessionId终止任何会话。只有登录的客户端拥有会话身份验证令牌。

9.3.4.4 X-Auth-Token HTTP 头

实现只使用兼容的TLS连接传输任何第三方身份验证服务和客户端之间的数据。因此,创建一个新的会话的POST应该只支持HTTPS,并且使用基本身份验证的所有请求应当需要HTTPS。

9.3.4.5 会话生命周期

注意,Redfish会话“超时”与一个令牌有效期的不同，其中，令牌有效期的例子是一些基于令牌的方法使用。对于Redfish会话,只要一个客户连续发送会话请求，并且多于通常会话超时周期,会话将保持开放，并且会话的身份验证令牌仍然有效。如果会话超时,那么会话自动终止。注意那才是Redfish。

9.3.4.6 会话终止或注销

客户注销时 Redfish 会话终止。这是通过对会话资源执行一个 DELETE 实现，其中，资源是通过返回会话创建时位置头的连接来识别，或者被响应数据返回的 SessionId 来识别。

通过对指定会话资源 ID 来执行 DELETE 一个会话的能力，允许有足够特权的管理人员通过一个不同的会话终止其它用户的会话。

9.3.5 AccountService

- 用户密码应该使用单向加密技术进行存储。
- 实现可能支持导出用户帐号和密码,但应当使用加密方法来保护他们。
- 用户帐户应当支持 ETags,应当支持原子操作
  - 实现可能会拒绝不包括一个 ETag 的请求
- 用户管理活动是原子的
- 当授权失败发生时，扩展的错误消息不能提供特权信息

9.3.6 异步任务（Async Tasks）

无论哪个用户/权限上下文用于开始一个异步任务，状态对象信息应当用于执行访问该

对象所需的特权。

9.3.7 事件订阅（Event Subscriptions）

在将事件数据对象推送到目的地之前，为了识别目的，Redfish 设备可以验证目的地。

9.3.8 特权权限模型/授权

授权子系统使用角色和权限来控制哪些用户对哪些资源的访问

角色:

- 角色是一组特权的定义。因此,两个有相同特权的角色应该有等价的行为。
- 所有用户被分配一个准确的角色。
- 这个规范定义了一组预定义的角色。 当创建用户时，其中一个角色应当分配给一个用户。
- 预定义的角色应当按照下面的方式被创建：
  - 角色名= "管理员"
    - AssignedPrivileges =Login，ConfigureManager，ConfigureUsers，ConfigureComponents ConfigureSelf
  - 角色名称=”操作员”
    - AssignedPrivileges =Login，ConfigureComponents，ConfigureSelf
  - 角色名称=“只读”
    - AssignedPrivileges =Login，ConfigureSelf
- 实现应当支持所有预定义的角色。
- 预定义的角色可能包括OEM特权。
- 预定义角色定义的特权数组不得被修改。
- 服务可以选择支持更多的“定制”的角色,并可能允许用户创建自定义角色的:1)发布角色集合;或2)实现可能实现一个预定义的自定义角色;或3)规范之外的其他机制。

特权:

- 特权是在一个定义的管理领域(例如配置用户)中一个执行操作的权限(如读、写)。
- 在角色资源的AssignedPrivileges数组中，Redfish规范定义了一组“分配特权”。
- 一个实现可能还包括“OemPrivileges”，其被指定在角色资源的OemPrivileges数组中。
- 通过使用在特权Redfish模式文件中定义的权限映射注释，权限被映射到资源中。
- 在映射中，多个特权构成一个特权的OR。

用户管理:

- 创建用户帐户时，用户被分配的角色。
- 用户的权限被他的角色定义。

ETag 处理:

- 对于ETag相关活动，实现应执行相同的特权模式，类似的执行ETag所代表的的数据。
- 例如,当活动要求特权访问由ETag表示的读数据项时，也需要有相同的访问读ETag的特权。

9.4 数据模型验证

9.4.1 模式

对于Redfish模式，服务器和客户端实现应该检查提供的数据。通过执行验证检查,以防止后期处理错误造成的漏洞。

当Redfish模式验证时,如果服务器和客户端之间有分歧,服务器可能会执行它的版本和拒绝请求。

客户不得执行数据篡改,除非Redfish模式允许。

当没有一个强有力的安全相关的要求时,权限不应该被修改。当权限要求已经被修改时，Redfish模式验证应包括权限检查。

注意:部分Redfish模式更新/更改的权限改变应当被记录到Redfish模式更改日志中。

当有一个安全的理由这样做时，幂等操作应被拒绝。

资源定义应包括所需的权限用于资源上执行读/ RW操作。
资源树稳定——资源上的权限应该是稳定的。

自定义动作——特权模式应持续应用于主体和响应中。为URI定义的适用特权模式应该继承自定义动作。

9.5 日志

9.5.1 安全日志条目必需的数据

实现应该记录身份验证请求(包括失败的)。身份验证登录/注销日志条目应当包含一个

用户标识符,可以用来唯一地标识客户端和一个时间戳。

9.5.2 日志记录的完整性

- 对于triggered、taken等调用活动，通过每一个中介,从RESTful服务调用的发起人到调用链中的最后一个实体, 每个实体记录条目到它们的审计日志中。这对于任何RESTful服务调用是一样,对于所说的实体中的被执行的活动，审计日志条目将“被完成”。
  - 应当——所有写活动，即POST、PUT /PATCH和DELETE
    - ✓ 注意:当创建一个新的日志条目时，记录事件的发生不是必需的
  - 应当——有能力记录特权读，即GET
    - ✓ 这种能力可能被默认打开。
- 由于安全原因，拒绝幂等操作需要被记录

9.5.3 审计日志的内容

细节:下面的需要生成事件

1、登录、注销、修改用户帐户
2、成功和拒绝登录尝试
3、尝试访问节点和其他资源的成功和拒绝的连接
4、关于用户帐户修改的细节
5、系统配置的所有更改
6、运行在Redfish兼容性设备(如，低级的诊断工具)中的内嵌实用工具使用的相关信息
7、访问Redfish兼容性设备系统接口的信息
8、网络地址和协议(如工作站IP地址和用于访问的协议)
9、保护措施的激活和停止

写事件的文件,每个事件的一个或多个消息至少应该有以下信息

- 用户ID
- 日期、时间
- 事件类型
- 事件描述

10 附件 A(信息)

10.1 更改日志

版本	日期	描述
1.0.0	2015-8-4	最初版本
1.0.1	2015-9-17	勘误表发布。明确LongDescription模式文件的规范使用。澄清“rel-describedby”链接头的使用。纠正OData上下文属性中选择列表的例子中的文本。澄清处理中的接受编码请求头。删除返回扩展错误资源方面重复的和相互矛盾的声明。澄清相对URI解析规则。各种语法修正。澄清USN格式。