2013 年第 10 期（总第 25 期）

数字图书馆标准规范
跟踪扫描

主办单位：中国科学院国家科学图书馆

2013 年 12 月

为传播科学知识，促进业界交流，特编译《标准规范跟踪扫描》，仅供个人学习、研究使用。

目 录

【标准规范报道】

1、国际云研讨会突出首个公开的 OASIS TOSCA 标准的互操作性演示，用于管理云应用程序

2013 年 10 月 15 日，在卢森堡举办的国际云研讨会（ICS）上，7 个公司参与了云应用 OASIS 拓扑和编排规范（TOSCA）的首个公开互操作演示。它展示了能够在不同提供者平台和基础设施的云中容易地被传输和管理的应用程序。此活动是 EuroCloud Congress 的一部分，它展示了应用程序能够从来自不同提供者平台和基础设施的云中被轻松地移植和管理。

“TOSCA 提供了扩大客户选择、改善可靠性并降低成本的可能，”OASIS 的 CEO Laurent Liscia 说。“使用 TOSCA 服务模板，由于应用程序开发人员构建的知识，企业用户可以很容易地将他们的应用程序从一个云移动到另一个云中，并进行精心安排。”

在 ICS 展示期间，一个客户关系管理（CRM）应用程序及其关系数据库在云编排工具和运行之间被交换为一个服务模板。特色产品主要来自 Fujitsu, HP, Huawei, IBM, SAP, Vnomic 和 Zenoss。

TOSCA 能够实现便携式的跨云部署，并简化了现有应用程序向云的迁移。它支持动态、灵活扩展和多重云应用程序的出现。TOSCA 提供全生命周期的业务流程和服务的可组合性，在服务提供商和实施两者中提供更大的选择自由性。

关于OASIS

OASIS是一个非盈利的国际联盟，它为全球信息社会实现开放标准的开发、融合和使用。OASIS促进行业共识并为云计算、安全、隐私、M2M、物联网、内容技术、商业交易、应急管理和其他应用程序生产全球标准。OASIS开放标准提供潜力来降低成本、激励创新、全球市场增长和保护技术免费选择的权利。联盟超过 5000 个参与者，他们代表了 65 多个国家中的 600 个组织和个体成员。

（编译自：International Cloud Symposium Features First Public Interoperability Demo of OASIS TOSCA Standard for Managing Applications across Clouds https://www.oasis-open.org/news/pr/international-cloud-symposium-features-first-public-interoperability-demo-of-oasis-tosca-sta [2013-10-15]）

（王妍编译，吴贝贝校对）

2、网络研讨会系列：打破标准间的障碍

对象管理组织（OMG）很高兴与其他四个标准开发组织（SDOs）进行一个小时的互动研讨会，该会议聚焦于 SDO 合作的成功案例，目标在于实现技术和语义互操作的不同级别。通过成功组织的具体案例来学习如何打破 SDOs 之间的障碍。

首个研讨会的主题是：

- 标准互操作性与 HL7——将 OMG 和 HL7 在开发更好标准中的优势最大化，该标准针对医疗保健 IT 解决方案服务。由 OMG 的 Richard Soley 汇报。
- 云标准协作——在 SNIA 和 OGF 之间如何进行协调，来引导多个开发和追踪云计算标准的组织能够参与。由存储网络行业协会的 Mark Carlson 和开放网格论坛的 Alan Sill 进行汇报。
- SDO 合作的障碍——哥伦比亚大学商学院的研究成果，它有关成功合作的障碍。由普华永道的 Eric E. Cohen 进行汇报。

参与者：对标准开发感兴趣的人，希望探讨不同商业信息标准之间合作、协调与集成的优势。

时间：2013 年 12 月 17 日，星期二，上午 11：00（美国东部标准时间）

“行业标准创建了一个市场，所有人都可以使用它来确保质量和互操作性，”OMG 的主席和首席执行官 Richard Soley 博士说。“通过联合起来进行合作，标准组织，例如 OMG，将确保一个通用的特定标准，并更加强大。”

（编译自：Webinar Series: Breaking Down Barriers Between Standards http://www.omg.org/news/releases/pr2013/Webinar_SICase-study.pdf[2013-11-14]）

（王妍编译，吴贝贝校对）

3、资源同步

NISO 和开放存档倡议的联合计划

2013 年 12 月 3 日 11：00am-12:00pm（美国东部时间），资源同步核心组成员 Bernhard Haslhofer 和 Simeon Warner 将汇报资源同步规范，并提供应用程序的实践案例和应用场景。资源同步将研究、开发、原型、测试和部署大规模网络资源同步的机制。

2013 LITA 论坛（11 月中旬在路易斯维尔举办）的参与者将被邀请参与资源同步指南，该会议将在 1：:3-4:30pm 举行（主要会议之后）。Herbert van de Sompel 引导这三个小时的会议。在会议上，参会人员将了解如何使用资源同步标准来将服务器之间的网络资源进行同步。

资源同步将为网络资源的大规模同步进行研究、开发、原型、测试和部署机制。资源同步，开始于 2011 年底，是 NISO 和开放存档倡议团队（OAI）之间的一个联合合作，由斯隆

基金会资助。建立在同步元数据的 OAI-PMH 策略的基础上，这个项目将加强使用现代网络技术的规范，但允许考虑对象自身的同步，而不仅仅是元数据。

由于互联网上作品或数据复制拷贝的扩散，保持存储库的持有物是最新的、准确的，这是一个越来越有挑战性的难题。通过自动化的复制和更新过程，新标准将会节省大量的时间、精力、资源，增加存储库中可用内容的总体可用性，以及减轻由互联网上存在的过时、不准确、作废的内容而产生的各种问题。

同步对于高完整性的或基本的网络资源尤为重要。例如，提供高质量服务，有关文化或学术资源集成的的门户网站，将明显受益于可靠的、统一的、可扩展的技术来与它们建立的集合保持同步。当我们从一个文档网络移动到数据网络时，同步变得更加重要：基于非同步化、不连贯科学或经济数据的决策会产生严重的不良影响。

此工作的最终产品将是一个规范，由专家和测试实施进行审查，它将详细说明一个方法，以互操作的方式将大规模的网络资源进行同步。

(编译自：ResourceSync (Resource Synchronization)A Joint NISO and Open Archives Initiative (OAI) project
ttp://www.niso.org/workrooms/resourcesync/[2013-11-26])

（王妍编译，吴贝贝校对）

4、NISO 网络研讨会：图书馆关联数据：从愿景走向现实

图书馆和文化机构群体都接受了向一个关联数据环境转移的愿景，将调整和汇集资源到更宏伟的语义网中。但是从愿景到实现既不简单，也难以理解的。很多机构已经开始开发所需的基础架构和工具，以便在对馆藏和资源的发现和导航服务中提供结构化数据。

参加 NISO 的这个网络研讨会，发言者将关注现实的关联数据项目——从展望模型到实施以及学到的经验——并汇报他们关于关联数据如何使研究、学术交流和出版受益的想法。

会议将于 2013 年 12 月 11 日 1:00 - 2:30 pm（东方时间）举行。

日程：

Jon Voss——行动改变世界，战略合作关系总监

LODLAM + Historypin:一个合作性的全球社区

Jon 将涉及到 LODLAM 的过去和未来，探索 Historypin 是如何运作来添加到关联历史数据的生态系统。

Jon Voss 致力于“big picture” 问题的革新方法和社区参与已经 15 年了。他的早期工作引导他为音乐节和摇滚明星开发了新的商业实践。十年后，他帮助宗教社区和其它机构发展和实施了技术基础设施和战略（与他们的信仰、文化和日常实践相适应）。在开展他自己的 IT 咨询公司前，他担任了 7 年的 San Francisco Zen Center 的 IT 总监。

Matt Miller——纽约公共图书馆的 NYPL 实验室，前端开发人员

关联的爵士乐项目：揭示爵士群体的关系

这个谈话将回顾关联的爵士乐项目如何将口述的历史抄本转换到一个关联开放的数据集和进程启用的应用程序中。因为我们要开始把数据关联到其它机构，所以我们也将展望项目的下一个步骤。

Silvia Southwick——UNLV 大学图书馆，数字集合元数据馆员

Cory Lampert——UNLV 大学图书馆，数字集合负责人

关联数据去神秘化：UNLV 关联数据项目

Cory Lampert 是 University of Nevada, Las Vegas（内华达大学）数字集合的领导，她负责大学图书馆数字行动的管理和战略计划。她的研究兴趣包括数字图书馆的最佳实践和技术世界中图书馆员发展的角色。 Cory 是“Jump Start Your Career as a Digital Librarian”这本书的贡献者，与他人合作了几篇论文、演示和项目来支持数字集合元数据的发展。

Silvia Southwick 的学术兴趣主要是数字图书馆的发展领域。特别感兴趣的主题包括元数据管理、数字图书馆技术和关联数据。

（编译自：NISO Webinar: Library Linked Data: From Vision to Reality http://www.niso.org/news/events/2013/webinars/linked_data ）

（吴贝贝编译，王妍校对）

【标准规范推介】

一、开放数据 4.0 版本第一部分：协议

1 介绍

开放数据协议（OData）支持基于 REST 数据服务的创建，允许资源可以通过 Web 客户端使用简单的 HTTP 消息来发布和编辑，该资源使用统一资源定位符（URLs）进行标识，并且在一个数据模型中被定义。这个规范定义了协议的核心语义和行为方面。

[OData-URL]规范定义了构建 URLs 的一系列规则来标识数据和元数据(由开放数据服务揭示)，同时也定义了一系列保留的 URL 查询选项。

[OData-CSDL]规范定义了实体数据模型（由开放数据服务揭示的）的一个 XML 表达。

[OData-Atom]和[OData-JSON]文档具体指定了被交换的资源表达(使用开放数据)的格式。

1.1 术语

这个文档中的关键词 “MUST”，“MUST NOT”，“REQUIRED”，“SHALL”，“SHALL NOT”，“SHOULD”，“SHOULD NOT”，“RECOMMENDED”，“MAY”，和 “OPTIONAL”在[RFC2119]中被解释。

2 概述

开放数据协议是一个通过 RESTful 界面进行数据交互的应用层协议。这个协议支持数据模型的描述并根据那些模型进行数据编辑和查询。为以下提供条件：

- 元数据：由一个特定数据提供商揭示的数据模型的一个机读描述。
- 数据：数据实体和它们之间关系的集合。
- 查询：请求服务执行过滤和其它向数据的转换，然后返回结果。
- 编辑：创建、更新和删除数据。
- 操作：调用自定义逻辑。
- 词表：附加自定义语义。

开放数据协议与其他基于 REST 的网络服务方法不同点在于，它提供一个统一的方法来描述数据和数据模型。这改善了系统之间的语义互操作性并且允许生态系统的产生。

为此，开放数据协议遵循以下设计原则：

- 更喜欢在各种数据仓储中运行的机制。尤其是不设定一个关系型数据模型。
- 可扩展性是重要的。服务应该能够支持扩展的功能，而不需要打破未注意到那些扩展的客户端。
- 遵循 REST 原则。
- 开放数据应该递增地构建。一个基本的、兼容的服务应当容易地被构建。
- 保持简捷。解释通用案例并在需要的时候提供扩展。

3 数据模型

这一部分提供了实体数据模型（EDM）的一个高层描述：一个用来描述数据（由开放数据服务揭示）的抽象数据模型。一个开放数据元数据文档是一个服务数据模型的表达。

EDM 中的中心概念是实体、关系、实体集、行为和功能。

实体是实体类型的实例（如 Customer， Employee 等 。

实体类型是带有一个键的命名的结构化类型。它们定义了一个实体的命名属性和关系。实体类型可以源自其它实体类型的单一继承。

一个实体类型的键由实体类型的原始属性（如 CustomerId, OrderId, LineId 等）的一个子集构成。

复杂类型是无键的被命名的结构化类型，由属性集组成。值类型的实例不能参考它们包

含实体之外的实例。复杂类型在一个实体中通常用作属性值，或用作操作参数。

被声明为一个结构化类型定义的一部分的属性被称作已声明的属性。结构化类型的实例可以包括其它未声明的动态属性。一个动态属性不能有与已声明属性相同的名称。允许客户端维持其它未声明属性的实体或复杂类型被称为开放类型。

一个实体到另一个实体的关系被表达为导航属性。导航属性通常被定义为一个实体类型的一部分，但是也可以作为未声明的动态导航属性出现在实体实例中。每一个关系都有一个基数。

枚举类型是被命名的原始类型，它们的值是带有整数值的命名常量。

类型定义是带有固定方面值如最大长度或精度的被命名的原始类型。类型定义可以用在原始的被归类属性的地方，如，在属性定义中。

实体集是被命名的实体集合（如 Customers 是包含 Customer 实体的实体集）。一个实体的键在一个实体集中唯一地标识实体。一个实体在一个给定点最多是一个实体集的成员。实体集提供进入数据模型的入口点。

操作允许在一个数据模型的部分区域里自定义逻辑的执行。Functions（功能）是那些没有副作用并可以被进一步组成的操作，例如，其它的过滤操作、功能或一个行为。Actions（行为）是允许副作用的操作，如数据修改，并且为了避免非确定性行为而不能被进一步构成。Actions 和 functions 可以绑定一个类型，使它们被称作那个类型的一个实例成员，也可以不绑定，被称为静态操作。行为导入和功能导入使不绑定的行为和功能被称作服务权限。

Singletons 是单独实体，它被获取为实体容器的子容器。

一个开放数据资源是模型里可以被解释的任何事物。

3.1 注释

模型和实例可以用注释来修饰。

注释可以用来规定关于一个元素的个体事实，例如它是否只读，或定义一个通用的概念，例如一个人或一部电影。

应用注释由一个术语、一个目标和一个值组成。值可以是静态值，或是一个表达式，它包含到注释实体的一个或更多属性的路径。

注释术语在元数据里被定义并且拥有一个名称和一个类型。

在一个通用命名空间中的一个相关术语集合构成了一个 Vocabulary（词表）。

4 服务模型

开放数据服务使用一个通用的数据模型来被定义。这个服务以机读格式展示了它的具体的数据模型，允许客户以一种良好定义的方式来与服务交互。

一个开放数据服务揭示两种良好定义的资源，它描述数据模型，一个服务文档和一个元数据文档。

服务文档列入了实体集、功能和可以检索到的单件（singleton）。客户端可以用服务文

档来导航模型（以超链接驱动的方式)。

元数据文档描述了被开放数据服务理解的类型、集合、功能和行为。客户端可以使用元数据文档来理解如何在服务里查询实体和与实体互动。

除了这两个“固定”资源，开放数据服务还包括动态资源。这些资源的 URLs 可以在元数据文档的信息里被运算。

4.1 实体 IDs 和实体参考

鉴于数据模型中的实体由实体集中的主键值唯一地标识，那么一个有效负载里的实体应被一个长久的、不透明的、全球唯一的实体 id(entity-id)标识。实体 id MUST 是在[RFC3987]被定义的一个 IRI，MAY 在有效负荷和 URLs 里被表达为合适的相关参考。当客户端准备接受 IRI 时，服务必须使用这个规范里的有效 URIs。

强烈建议服务使用在[OData-URL]中定义的典型 URL 作为实体 id，但客户端不能假定实体 id 可以被用来定位实体，除非 Core.DereferenceableIDs 术语被用在实体容器里，客户端也不能假定来自实体 id 结构的任何语义。典型资源$entity 提供了一个将实体 id 解析为实体表达的一个普通机制。

实体 ids 使用标准 URL 的服务在 Core.ConventionalIDs 术语里注释它们的实体容器，见[OData-VocCore]。

实体参考（Entity References）使用实体的 entity-id 来参考一个实体。

4.2 可读 URLs 和编辑 URLs

一个实体的可读 URLs 是可以用来读取实体的 URL。

一个实体的编辑 URLs 是可以用来更新或删除实体的 URL。

强烈建议服务要使用在[OData-URL]定义的典型 URL 作为可读 URL 和编辑 URL。

5 版本化

版本化可以是客户端和服务独立地发展。开放数据为协议和数据模型版本化定义了语义。

5.1 协议版本化

开放数据请求和响应根据 OData-Version 标题进行版本化。

开放数据客户端在请求里包含 OData-MaxVersion 标题来规定最大可接受的响应版本。

服务用最大支持版本(小于或等于被请求的 OData-MaxVersion)来响应。OData-Version 和 OData-MaxVersion 标题字段的句法在[OData-ABNF]中被规定。

规范的这个版本定义了数据服务版本 4.0。

5.2 模型版本化

由开放数据服务揭示的数据模型定义了开放数据服务和客户端之间的契约。允许服务在没有打破现有客户端的条件下扩展它们的模型。

以下的数据模型附加被认为是安全的并且不需要服务来将它们的入口点版本化。

- 增加一个无值的或有一个默认值的属性，并且不与现有的动态属性相冲突。
- 增加一个无值或集合级别的导航属性，并且不与现有的动态导航属性相冲突。
- 增加一个新的实体类型到模型。
- 增加一个新的复杂类型到模型。
- 增加一个新的实体集合。
- 增加一个新的单件。
- 增加一个行为、一个功能、一个行为导入或功能导入。
- 增加一个空白值的行为参数。
- 增加一个类型定义或枚举。
- 增加一个注释到模型元素。

客户端应该（SHOULD）为服务准备好来对它们的模型做出渐进式的变化。尤其，客户端应该做好准备来接受不是事先在服务器中定义的属性。

服务器不应该（SHOULD NOT）根据已认证的用户来改变它们的数据模型。如果数据模型是独立于用户或用户组的，那么所有改变必须（MUST）是安全的变化。

6 扩展性

开放数据协议通过一个版本、惯例和明确扩展点的结合来支持用户驱动和版本驱动的扩展。

6.1 查询选项扩展

请求 URL 里的查询选项可以控制一个特定请求如何被服务器处理。

开放数据定义系统查询选项的前缀是“$”。服务器可以支持不在开放数据规范中定义的其它查询选项，但是它们禁止（MUST NOT）以“$”或“@”字符开头。

开放数据服务不应该（SHOULD NOT）在一个请求里要求查询选项是明确的。服务器应该（SHOULD）回绝包含它们不理解的查询选项的请求，必须（MUST）回绝包含不支持这个规范定义的开放数据查询选项的请求。

6.2 有效负荷扩展

开放数据根据特定的格式，在有效负荷里支持扩展。

如果接受者需要理解它来根据特定的 OData-Version 标题来正确解释有效负荷，除了格式，其它内容禁止出现。

6.3 行为/功能扩展

行为和功能扩展了操作集合，它们 可以在一个服务器或资源里执行。行为可以有副作用。功能禁止有副作用。

完全合格的行为和功能名称包含一个命名空间或别名前缀。保留 Edm，odata 和 geo 命名空间是为了使用这个规范。

一个开放数据服务器必须回绝任何包含它不理解行为或功能的请求。

6.4 词表扩展

在一个体系中被定义的注释集合构成了一个词表。共享的词表为开放数据提供了一个有力的扩展点。

元数据注释可以用来定义一个元数据元素的其它特征或能力，例如一个服务器、实体类型、属性、功能、行为或参数。

实例注释可以被用来定义与一个特定结果、实体、属性或错误相关的其它信息。

一个服务器禁止要求客户端理解自定义注释，目的是精确地解释一个响应。

开放数据定义了一个核心词表和一个描述了行为方面的基本术语表，以及可以被用在定义其它词表的术语。

6.5 标题字段扩展

开放数据定义了特定 HTTP 请求和响应标题的语义。

单项服务器定义了自定义标题。这些标题禁止以 OData.开头。当对服务器做出请求时，自定义标题应该是可选的。一个服务器禁止要求客户端理解自定义标题以精确解释响应。

6.6 格式扩展

一个开放数据服务必须至少支持[OData-JSON]或[OData-Atom]中的一种，也可以支持其它附加格式。

7 格式

客户端可以通过一个 Accept 标题或系统查询选项$format 来请求一个在[RFC2616]中定义的特定响应格式。

在一个请求里 Accept 标题和$format 查询选项都明确规定的情况下，在$format 查询选项中被明确规定的值必须被使用。

8 标题字段

开放数据定义了以下请求和响应标题的语义。其它标题可以是明确规定的，但是没有在开放数据里定义的唯一语义。

8.1 通用标题

Content-Type, Content-Length,和 OData-Version 标题在开放数据请求和响应里是通用的。

8.1.1 标题 Content-Type

正如在[RFC2616]里明确规定的，单独请求或响应体的格式必须在请求或响应 Content-Type 标题里被明确规定。

8.1.2 标题 Content-Encoding

正如在[RFC2616]里明确规定的，Content-Encoding 标题字段被用作一个对媒体类型的修饰。当它出现时，它的值指明了被应用在实体个体里的其它内容编码。

8.1.3 标题 Content-Length

正如在[RFC2616]里明确规定的，当决定消息的长度先于被转化时，一个请求或响应应该包含 Content-Length 标题。开放数据对于写 Content-Length 对 HTTP 没有其它的要求。

8.1.4 标题 OData-Version

开放数据客户端应该在一个请求里使用 OData-Version 来明确协议（用于生成请求）的版本。

8.2 请求标题

除了通用标题，客户端可以规定以下请求标题的任何组合。

8.2.1 标题 Accept

正如在[RFC2616]里明确规定的，客户端在在 Accept 标题里规定了可以接受的格式集合。

如果在 Accept 标题里规定的媒体类型包含一个 charset 格式参数，并且请求也包含一个 Accept-Charset 标题，那么必须使用 Accept-Charset 标题。

如果在 Accept 标题里规定的媒体类型不包含一个 charset 格式参数，那么响应的 Content-Type 标题禁止包含一个 charset 格式参数。

8.2.2 标题 Accept-Charset

正如在[RFC2616]里明确规定的，客户端可以在 Accept-Charset 标题里明确规定可以接受的符号集合。

8.2.3 标题 If-Match

正如在[RFC2616]里明确规定的，一个客户端可以在一个 GET,PUT,PATCH 或 DELETE 请求里包含一个 If-Match 标题。If-Match 的值必须是一个 ETag 值，或“*”来匹配任何值。

客户端可以在一个 PUT 或 PATCH 请求里包含一个 If-Match 标题来确保请求被处理为作为一个 update 而不是一个 upsert。

8.2.4 标题 If-None-Match

正如在[RFC2616]里明确规定的，一个客户端可以在一个 GET,PUT,PATCH 或 DELETE 请求里包含一个 If-None-Match 标题。If-None-Match 的值必须是一个 ETag 值，或“*”来匹配任何值。

8.2.5 标题 OData-Isolation

OData-Isolation 标题规定了目前请求和外部改变的分离。这个标题唯一支持的值是 snapshot。

如果服务器不支持 OData-Isolation:snapshot，并且这个标题在请求里是明确规定的，那么服务器禁止处理此请求，并必须以 412 Precondition Failed 响应。

Snapshot isolation 保证请求所返回的所有数据，包含一个跨多个页面的多重请求，将与一个单一点保持一致。

OData-Isolation 标题对链接（除了下一个链接）没有影响。导航链接、可读链接、编

辑链接返回数据的目前版本。

8.2.6 标题 OData-MaxVersion

客户端应该规定一个 OData-MaxVersion 请求标题。

如果 OData-MaxVersion 没有被明确规定，那么服务器应该将请求解释为是与服务器支持的最大版本相同的一个 OData-MaxVersion。

8.2.7 标题 Prefer

Prefer 标题允许客户端从服务器中请求确定的行为。服务器必须忽略不被服务器支持或理解的优先值。

Prefer 标题的值是 preferences 的一个逗号分隔列表。作为对一个包含 Prefer 标题的请求的响应，服务器可以返回 Preference-Applied 标题。

9 通用响应状态代码

一个开放数据服务可以使用任何有效的、合适的 HTTP 状态代码来对任何请求进行响应。一个服务器在选择 HTTP 状态代码的时候应该尽可能的精确。

以下表达了最常用的成功响应代码。在一些情况下，一个服务器可以用一个更精确的成功代码来响应。

9.1 成功响应

以下响应代码表示了成功的请求。

9.1.1 响应代码 200 OK

一个不创建资源的请求如果被成功实现并且资源的值不是空白，会返回 200 OK。在这种情况下，响应体必须包含在请求 URL 中明确规定的资源的值。

9.1.2 响应代码 201 Created

成功创建资源的一个 Create Entity, Create Media Entity, Create Link 或 Invoke Action 请求，返回 201 Created。在这种情况下，响应体必须包含被创建的资源。

9.1.3 响应代码 202 Accepted

202 Accepted 指明数据服务请求已经被接受，还没有完全地被异步执行。

9.1.4 响应代码 204 No Content

如果被请求的资源是空白值或服务器应用了一个 return=minimal 优先，那么请求返回 204 No Content。在这种情况下，响应体必须是空白的。

9.1.5 响应代码 3xx Redirection

一个 3xx Redirection 指明客户端需要采取进一步的行为来满足请求。在这种情况下，响应应该在可以得到结果的 URL 中包含一个 Location 标题，它可以包含一个 Retry-After 标题。

9.1.6 响应代码 304 Not Modified

当客户端执行一个包含一个 If-Match 或 If-None-Match 的 GET 请求时，返回 304 Not

Modified。在这种情况下，响应不应该包含其它的标题来防止缓存的实体个体和更新标题之间的不一致。

9.2 客户端错误响应

在 4xx 范围内的错误代码指明了一个客户端错误，例如一个畸形的请求。

服务器必须确保没有任何可见的改变发生在服务状态里，由于返回一个错误状态代码的请求结果。

9.2.1 响应代码 404 Not Found

404 Not Found 表明由请求 URL 规定的资源不存在。响应体可以提供额外的信息。

9.2.2 响应代码 405 Method Not Allowed

405 Method Not Allowed 表明由请求 URL 规定的资源不支持请求方法。

9.2.3 响应代码 410 Gone

410 Gone 表明请求的资源不能再有效。

9.3 服务器错误响应

正如在[RFC2616]中明确规定的，5xx 范围内的错误代码表明服务错误。

9.3.1 响应代码 501 Not Implemented

如果客户端请求功能不能被开放数据服务执行，那么服务器必须以 501 Not Implemented 响应，并且响应体应该描述不被执行的功能。

9.4 In-Stream 错误

在这种情况下，服务器在发送一个成功状态到客户端后遇到一个错误，服务器必须在有效负荷内产生一个错误。

10 情境 URL

情境 URL 描述有效负荷的内容。

请求有效负荷通常不需要情境 URL，因为有效负荷的类型通常可以由请求 URL 决定。

下面的子部分描述了情境 URL 如何通过提供一个情境 URL 模板来被构建(针对每个有效负荷的类别)。情境 URL 模板使用以下术语：

- {context-url}是到$metadata 文档的资源路径。
- {entity-set}是一个实体集的名称或到一个控制导航属性的路径。
- {property-list}是属性的一个括号化的逗号分隔列表。
- {type-name}是一个复杂类型或实体类型的有效名称。
- {/type-name}是一个可选部分，它包含一个来源类型的有效名称。

情境 URL 的完整语法在 [OData-ABNF]中定义。

10.1 服务器文档

情境 URL 模板：{context-url}

服务器文档的情境 URL 是服务器的情境 URL。

例 9：资源 URL 和相应的情境 URL

```
http://host/service/
http://host/service/$metadata
```

10.2 实体集合

情境 URL 模板：{context-url}#{entity-set}

如果所有集合里的实体是一个实体集的成员，那么它的名称是情境 URL 片段。

例 10：资源 URL 和相应的情境 URL

```
http://host/service/Customers
http://host/service/$metadata#Customers
```

如果实体是被包含的，那么 entity-set 是最高层的实体集，紧跟其后的是包含实体的导航属性路径。

例 11：资源 URL 和包含实体的相应情境 URL

```
http://host/service/Orders(4711)/Items
http://host/service/$metadata#Orders(4711)/Items
```

10.3 实体

情境 URL 模板：{context-url}#{entity-set}/$entity

如果一个响应或响应部分是实体集被声明类型的一个单独实体，那么/$entity 被附加到情境 URL。

例 12：资源 URL 和相应的情境 URL

```
http://host/service/Customers(1)
http://host/service/$metadata#Customers/$entity
```

如果实体是被包含的，那么 entity-set 对包含实体的导航属性来说是规范的 URL，例如 Orders(4711)/Items。

例 13：资源 URL 和包含实体的相应情境 URL

```
http://host/service/Orders(4711)/Items(1)
http://host/service/$metadata#Orders(4711)/Items/$entity
```

10.4 单件

情境 URL 模板：{context-url}#{singleton}

如果一个响应或响应部分是一个单件，那么它的名称是情境 URL 片段。

例 14：资源 URL 和相应的情境 URL

```
http://host/service/Contoso
http://host/service/$metadata#Contoso
```

10.5 来源实体的集合

情境 URL 模板：{context-url}#{entity-set}{/type-name}

如果一个实体集专有地由来源实体组成，那么一个 type-cast 部分被添加到情境 URL。

例 15：资源 URL 和相应的情境 URL

```
http://host/service/Customers/Model.VipCustomer
http://host/service/$metadata#Customers/Model.VipCustomer
```

10.6 来源实体

情境 URL 模板：{context-url}#{entity-set}{/type-name}/$entity

如果一个响应或响应部分是来源于实体集被声明类型的一个单独实体，那么/<Type>/$entity 被附加到情境 URL。

例 16：资源 URL 和相应的情境 URL

```
http://host/service/Customers(2)/Model.VipCustomer
http://host/service/$metadata#Customers/Model.VipCustomer/$entity
```

10.7 预计实体的集合

情境 URL 模板：{context-url}#{entity-set}{/type-name}{property-list}

如果一个结果只包含属性的一个子集，所选属性的括号化逗号分隔列表被附加到{entity-set}上，在可选的 type-cast 部分之后。

例 17：资源 URL 和相应的情境 URL

```
http://host/service/Customers?$select=Address,Orders
http://host/service/$metadata#Customers(Address,Orders)
```

10.8 预计实体

情境 URL 模板：{context-url}#{entity-set}{/type-name}{property-list}/$entity

如果一个单独实体包含属性的一个子集，那么所选属性的括号化逗号分隔列表被添加到{entity-set}，在可选的 type-cast 部分之后，并优先于添加到/$entity。

例 18：资源 URL 和相应的情境 URL

```
http://host/service/Customers(1)?$select=Name,Rating
http://host/service/$metadata#Customers(Name,Rating)/$entity
```

例 19：资源 URL 和响应的情境 URL

```
http://host/service/Customers$select=Name&$expand=Address/Country
http://host/service/$metadata#Customers(Name,Address/Country)
```

10.9 预计的可扩展实体集合

情境 URL 模板：{context-url}#{entity-set}{/type-name}{property-list}

如果一个导航属性是可扩展的，那么属性的括号化逗号分隔列表要包含可扩展的导航属性。如果$expand 包含一个嵌套的$select，那么导航属性要以从相关实体中选取的属性的括号化逗号分隔列表为后缀。

例 20：资源 URL 和相应的情境 URL

```
http://host/service/Employees/Sales.Manager?
                  $expand=DirectReports($select=FirstName,LastName;$levels=4)
http://host/service/$metadata
               #Employees/Sales.Manager(DirectReports+(FirstName,LastName))
```

10.10 预计的可扩展实体

情境 URL 模板：{context-url}#{entity-set}{/type-name}{property-list}/$entity

如果一个单独实体是可扩展的和可预计的，那么所选属性的括号化逗号分隔列表添加到{entity-set}，在可选的 type-cast 部分之后，并优先于添加到/$entity。

例 21：资源 URL 和相应的情境 URL

```
http://host/service/Employees(1)/Sales.Manager?
                  $expand=DirectReports($select=FirstName,LastName;$levels=4)
http://host/service/$metadata
      #Employees/Sales.Manager(DirectReports+(FirstName,LastName))/$entity
```

10.11 实体参考集合

情境 URL 模板：{context-url}#Collection($ref)

如果一个响应是实体参考的集合，那么情境 URL 不包含被参考实体的类型。

例 22：资源 URL 和实体参考集合的相应情境 URL

```
http://host/service/Customers('ALFKI')/Orders/$ref
http://host/service/$metadata#Collection($ref)
```

10.12 实体参考

情境 URL 模板：{context-url}#$ref

如果一个响应是单独的实体参考，那么$ref 是情境 URL 部分。

例 23：资源 URL 和单独实体参考的相应情境 URL

```
http://host/service/Orders(10643)/Customer/$ref
http://host/service/$metadata#$ref
```

10.13 复杂或初始类型集合

情境 URL 模板：{context-url}#Collection({type-name})

如果一个响应是一个复杂类型或初始类型的集合，那么情境 URL 名称是属性的类型。

例 24：资源 URL 和相应的情境 URL

```
http://host/service/Customers(1)/Hobbies
http://host/service/$metadata#Collection(Edm.String)
```

10.14 复杂或初始类型

情境 URL 模板：{context-url}#{type-name}

如果一个响应是一个复杂类型或初始类型，那么情境 URL 包含属性的完整有效类型。

例 25：资源 URL 和相应的情境 URL

```
http://host/service/Customers(1)/Name
http://host/service/$metadata#Edm.String
```

10.15 操作结果

情境 URL 模板：

{context-url}#{entity-set}

{context-url}#{entity-set}/$entity

{context-url}#Collection({type-name})

{context-url}#{type-name}

如果来自一个行为或功能的响应是实体或一个单独实体的集合，那么情境 URL 标识了实体集，否则它标识由操作返回的类型。

例 26：资源 URL 和相应的情境 URL

```
http://host/service/TopFiveCustomers
http://host/service/$metadata#Customers
```

10.16 Delta 响应

情境 URL 模板：{context-url}#{entity-set}{/type-name}{property-list}/$delta

一个 Delta 响应的情境 URL 与根实体集的情境 URL 相同。

例 27：资源 URL 和响应的情境 URL

```
http://host/service/Customers?$deltaToken=1234
http://host/service/$metadata#Customers/$delta
```

10.17 Delta 响应中的项目

情境 URL 模板：

{context-url}#{entity-set}/$deletedEntity

{context-url}#{entity-set}/$link

{context-url}#{entity-set}/$deletedLink

除了新的或改变的实体（拥有实体的规范情境 URL），一个 Delta 响应可以包含删除的实体、新链接或删除的链接。它们通过相应的情境 URL 部分被标识。{entity-set}相当于被删除的实体集或来源实体集。

10.18 $all 响应

情境 URL 模板：{context-url}#Collection(Edm.EntityType)

对虚拟集合$all 请求的响应，要使用内建的抽象实体类型。在这样一个响应中每个单独实体都有其单独的情境 URL，它标识实体集或单件。

10.19 $crossjoin 响应

情境 URL 模板：{context-url}#Collection(Edm.ComplexType)

对虚拟集合$crossjoin 请求的响应，要使用内建的抽象复杂类型。这些响应中的单独实例没有情境 URL。

11 数据服务请求

11.1 元数据请求

OData 服务是一种自描述服务，它揭示定义实体集、关系、实体类型和操作的元数据。

11.1.1 服务文档请求

服务文档使得简单的超媒体驱动客户端能够列举和探索数据服务提供的资源。

OData 服务必须支持从服务的根 URL 返回一个服务文档。

服务文档的格式取决于所选择的格式。

11.1.2 元数据文档请求

OData 元数据文档是数据模型的表示，数据模型描述了由 OData 服务揭示的数据以及操作。

[OData-CSDL]为 OData 元数据文档描述了一个 XML 的表述形式,同时也提供了一个 XML schema 来验证其内容。OData 元数据文档的 XML 表述形式的媒体类型是 application/xml。

OData 服务必须揭示一个元数据文档，此文档描述由服务揭示的数据模型。元数据文档的 URL 必须是追加$metadata 服务的根 URL。要检索该文档，客户端需要向元数据文档 URL 发出一个 GET 请求。

如果一个元数据请求没有指定格式偏好，那么 XML 的表述形式必须返回。

11.1.3 元数据服务文档请求

OData 服务可以揭示一个元数据服务。OData 元数据服务是数据模型的表示，该模型将由 OData 服务揭示的数据以及操作描述为一个带有固定的（元）数据模型的 OData 服务。

元数据服务必须使用[OData-CSDL]中定义的模式。元数据服务的根 URL 是该服务的元数据文档 URL 加上一个斜杠。要检索该文档，客户端需要向元数据服务根地址 URL 发出一个 GET 请求。

11.2 数据请求

OData 服务支持通过 HTTP GET 来进行数据请求。

URL 的路径指定了请求目标（比如：实体集合）。附加查询操作符，如筛选，排序，页面和投影操作是通过查询选项来指定。

本节描述了 OData 定义的数据请求类型，关于建立请求语法的全部细节，参见[OData-URL]。

OData 服务是超媒体驱动的服务，它将 URL 返回给客户端。如果客户端随后请求的资源以及 URL 已经过期,那么该服务应该回应 410 Gone。如果这是不可行的,该服务必须回应 404 Not Found。

返回的数据格式是取决于由客户端指定的请求和格式。

11.2.1 评价系统的查询选项

OData 定义了一系列的系统查询选项，它允许细化要求。请求的结果必须按以下列顺序进行评估：

在应用任何服务器驱动分页之前：

- $search

- $filter
- $count
- $orderby
- $skip
- $top

在应用任何服务器驱动分页之后：

- $expand
- $select
- $format

11.2.2 个别实体请求

要检索个别实体，客户端向实体的可读 URL 发出一个 GET 请求。

可读 URL 可以从包含该实例的一个响应有效负载中获得。另外，服务可以支持使用实体的关键值来构造一个可读 URL，在[OData-URL]里面做了阐述。

结构或者导航属性集的返回可以通过指定$select 或者$expand 系统查询选项。

客户端必须准备接收实体或者复杂类型实例中的附加属性。

那些不可用的属性，比如由于权限限制而不可用的属性并不会返回。在此种情况下，由[OData-VocCore] 定义的 Core.Permissions 注释必须返回属性值 Core.Permission'None'。

如果没有实体存在于 URL 请求的关键值中，该服务会返回 404 Not Found。

11.2.3 个别属性请求

要检索个别属性，客户端需要向属性 URL 发送一个 GET 请求。属性 URL 是实体的可读 URL 后面加上 “/”与属性名。

对于复杂类型的属性，路径可以被进一步扩展，并加上该复杂类型的个别属性名称。

详细信息参见[OData-URL]。

如果该属性是单值的并且有空值，会回应 204 No Content。

如果一个属性是不可用的，比如由于权限限制，会回应 404 Not Found。

实例 28：

```
http://host/service/Products(1)/Name
```

11.2.4 指定返回属性

$select 和 $expand 系统查询选项使得客户端在响应并返回信息时指定一组结构属性以及导航属性。该服务会包括在$select 和 $expand 中未被指定的附加属性或者在元数据文档中没有被定义的属性。

11.2.5 查询集合

OData 服务支持实体、复杂类型实例以及原始值的查询集合。

11.2.6 相关实体请求

要根据一个特定关系来请求相关实体，客户端需要向来源实体的请求 URL 发出一个 GET 请求，后面加上一个斜杠以及表示关系的导航属性名称。

实例 58：在产品实体集中返回 ID=1 的产品供应商

```
http://host/service/Products(1)/Supplier
```

11.2.7 实体引用请求

想要请求实体引用来代替实际的实体，客户端需要发出一个 GET 请求并且将/$ref 附加到资源路径上。

实例 59：每个 ID=0 的产品订购的实体引用集合

```
http://host/service/Products(0)/Orders/$ref
```

11.2.8 解析实体-ID

要解析一个实体 ID，客户端需要向$entity 资源发出一个 GET 请求，该资源位于 URL /$entity。

实例 60：返回一个给定实体 ID 的实体

http://host/service/$entity?$id=http://host/service/Products(0)

```
http://host/service/$entity?$id=http://host/service/Products(0)
```

11.2.9 元素数目请求

要请求实体集合的元素数目或者集合属性的元素数目，服务 器需要发出一个 GET 请求并且将/$count 附加到该集合的资源路径上。

11.2.10 系统查询选项$format

系统查询选项$format 指定响应的媒体类型。

11.3 变更请求

服务通过添加注释实体集的设置和功能来实现它们的变更跟踪功能。Capabilities.ChangeTracking 在[OData-VocCap]中有定义。

客户端通过指定一个 odata.track-changes 偏好来请求服务跟踪变更。

11.3.1Delta 链接

Delta 链接是不透明的、服务生成的链接，客户端用它来检索结果的后续变化。

11.3.2 使用 Delta 链接

客户端通过调用 Delta 链接的 GET 方法来请求变化。客户端不得在 delta 链接后面附加系统查询选项。

11.4 数据修改

可更新的 OData 服务支持对部分或者全部实体进行创建、更新以及删除操作。另外，一个成功完成的数据修改请求必须不能违反数据的完整性。

11.4.1 常见的数据修改语义

数据修改请求使用下面的语义，分别是：

（1）避免更新冲突使用 ETags

（2）处理 DateTimeOffset 的值

（3）处理元数据未提供的属性

（4）处理一致性约束

11.4.2 创建一个实体

要在集合中创建一个实体，客户端需要向那个集合的 URL 发送一个 POST 请求。这个请求必须包含一个有效的实体表示。

11.4.3 更新一个实体

服务应该支持 PATCH 作为更新实体的首要方式。PATCH 通过直接修改客户端指定的值来在客户端和服务器之间提供更多的跳回。

11.4.4 Upsert 一个实体

当客户端向一个单一实体（不存在的）的有效 URL 发出一个更新请求。在这种情况下该服务必须将其当作创建一个新的实体或者完全失败的请求来处理。

11.4.5 删除一个实体

向一个实体的编辑 URL 发出一个成功的删除请求，会删除实体。请求主体应该是空的。

在成功完成删除后，返回的内容必须是 204 No Content 并且包含一个空的主体。

11.4.6 实体之间的关系修改

实体之间的关系由导航属性（在数据模型里面有提到）来表 示。导航属性的 URL 协定在[OData-URL]里面有描述。关系修改可以包括：

（1）添加一个引用到一个集合值（Collection-Valued）导航属性

（2）删除一个实体引用

（3）更改单值导航属性的引用

11.4.7 管理媒体实体

一个媒体实体是表示一个带外流的实体，比如一个图片。

一个媒体实体必须有一个可以用来读取媒体流的来源 URL。对媒体实体的管理可以包括：（1）创建媒体实体

（2）编辑媒体实体流

（3）删除媒体实体

11.4.8 管理流属性

一个实体可能有一个或多个流属性。流属性是 Edm.Stream 类型的属性。

11.4.9 直接管理值和属性

值和属性可以用 URL 来直接处理。这样可以使它们能够被单独修改。参见 [OData-URL]

的详细信息。

直接管理值的属性可以包括：

（1）更新一个原始属性

（2）设置一个空值

（3）更新一个复杂类型

（4）更新一个集合属性

11.5 操作

自定义操作在[OData-CSDL]里被表示为 Action, ActionImport, Function, 和 FunctionImport 元素。

11.5.1 将一个操作绑定到资源

Actions 和 Functions 可能被绑定到一个实体类型、原始类型、复杂类型或者一个集合上。绑定操作的第一个参数是 binding parameter。

11.5.2 在有效负载内进行可用操作

服务可能会返回可用的行为和/或绑定到特定实体的功能。

11.5.3 功能（Functions）

功能是由 OData 服务揭示的操作，它必须返回数据也必须没有可见的副作用。它包括：

（1）调用一个功能（内联参数语法等）

（2）功能重载解析

11.5.4 行为（Actions）

行为是由 OData 服务揭示的操作，当它被调用时可能会有副作用。行为可能会返回数据但是不能被进一步重组。行为操作包括：

（1）调用一个行为

（2）行为重载解析

11.6 异步请求

一个偏好的标头加上一个回应—异步（respond-Async）偏好，允许客户端异步地处理一个数据服务请求。

11.7 批量请求

批量请求允许将多个操作归类为一个单个 HTTP 请求。一个批量请求被表示为多部分的 MIME 版本 1.0 消息[RFC2046]。

12 安全注意事项

本节给应用程序开发者、信息提供者以及 OData4.0 版本的使用者提供参考。OData 是一个基于其他服的多格式服务，它既有优点也有缺点。

13 一致性（Conformance）

OData 被设计成一组公约，它可以在现有标准的上层上执行，目的是提供常用功能的共

同表达。不是所有的服务都能支持在协议中定义的公约；各种服务选择在 OData 中定义的公约作为揭示其功能的表达。

13.1 OData 服务一致性级别

OData 定义了三个 OData 服务一致性的级别，分别是：OData 最小一致性级别、OData 中级一致性级别、OData 高级一致性级别。

（编译自：OData Version 4.0 Part 1: Protocol Committee Specification 01 http://docs.oasis-open.org/odata/odata/v4.0/cs01/part1-protocol/odata-v4.0-cs01-part1-protocol.html#_Toc365046261）

（吴贝贝编译，王妍校对）