OASIS OPEN PROJECTS

OSLC Core Version 3.0. Part 1: Overview

OASIS Standard
26 August 2021

This stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/oslc-core.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/oslc-core.pdf

Previous stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/oslc-core.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/oslc-core.pdf
(published as Project Specification)

Latest stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/oslc-core.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/oslc-core.pdf

Latest version:

https://open-services.net/spec/core/latest

Latest editor's draft:

https://open-services.net/spec/core/latest-draft

Open Project:

OASIS Open Services for Lifecycle Collaboration (OSLC) OP

Project Chairs:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Editors:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Additional components:

This specification is one component of a Work Product that also includes:

- *OSLC Core Version 3.0. Part 1: Overview (this document)*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/oslc-core.html
- *OSLC Core Version 3.0. Part 2: Discovery*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/discovery.html
- *OSLC Core Version 3.0. Part 3: Resource Preview*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-preview.html
- *OSLC Core Version 3.0. Part 4: Delegated Dialogs*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/dialogs.html
- *OSLC Core Version 3.0. Part 5: Attachments*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/attachments.html
- *OSLC Core Version 3.0. Part 6: Resource Shape*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-shape.html
- *OSLC Core Version 3.0. Part 7: Vocabulary*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.html
- *OSLC Core Version 3.0. Part 8: Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.html

- *OSLC Core Version 3.0. Machine Readable Vocabulary Terms*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.ttl
- *OSLC Core Version 3.0. Machine Readable Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.ttl

Related work:

This specification is related to:

- *OSLC Core Version 3.0: Link Guidance*. Work in progress. Current draft: https://oslc-op.github.io/oslc-specs/notes/link-guidance.html

RDF Namespaces:

http://open-services.net/ns/core#

Abstract:

Defines the overall approach to Open Services for Lifecycle Collaboration (OSLC) based specifications and capabilities that extend and complement W3C Linked Data Platform [LDP]. OSLC Core 3.0 constitutes the approach outlined in this document and capabilities referenced in other documents.

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the “Latest stage” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Open Project are listed at https://open-services.net/about/.

Comments on this work can be provided by opening issues in the project repository or by sending email to the project’s public comment list oslc-op@lists.oasis-open-projects.org.

The English version of this specification is the only normative version. Non-normative translations may also be available. Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:
[OSLC-Core-3.0-Part1]
OSLC Core Version 3.0. Part 1: Overview. Edited by Jim Amsden and Andrii Berezovskyi. 26 August 2021. OASIS Standard. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/oslc-core.html. Latest stage: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/oslc-core.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

Table of Contents

1. Introduction

This section is non-normative.

Information Technology (IT) enterprises are constantly addressing demands to do more with less. To meet this demand they need more efficient development processes and supporting tools. This has resulted in demand for better support of integrated system and software processes. Enterprises want solutions (such as software or hardware development tools) from different vendors, open source projects and their own proprietary components to work together. This level of integration, however, can become quite challenging and unmanageable. In order to enable integration between a heterogeneous set of tools and components from various sources, there is a need for a sufficient supporting architecture that is loosely coupled, minimal, and standardized. OSLC is based on World Wide Web and Linked Data principles, such as those defined in the W3C Linked Data Platform [LDP], to create a cohesive set of specifications that can enable products, services, and other distributed network resources to interoperate successfully [LDP].

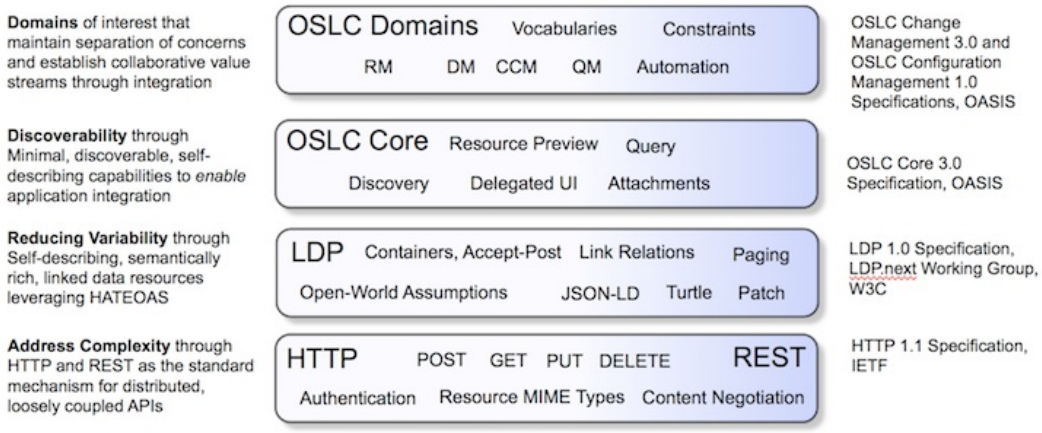

Fig. 1 OSLC Core 3.0 Architecture

OSLC is motivated by domain-driven scenarios that inspire standardization of common capabilities across disciplines such as change management, requirements management, and quality management, as well as by cross-domain scenarios such as Application Lifecycle Management (ALM) & DevOps, Product Lifecycle Management (PLM), and Integrated Service Management (ISM). The OSLC approach focuses on software lifecycle management to ensure it meets a core set of scenarios and requirements. Nonetheless, it can be used by tools belonging to any other domains and cross-domain scenarios such as Internet of Things, back office application integration, and customer relationship management.

The OSLC Core specifications provide additional capabilities that expand on the W3C LDP capabilities, as needed, to enable key integration scenarios. These capabilities define the essential and common technical elements of OSLC domain specifications and offer guidance on common concerns for creating, updating, retrieving, and linking to lifecycle resources based on W3C [LDP].

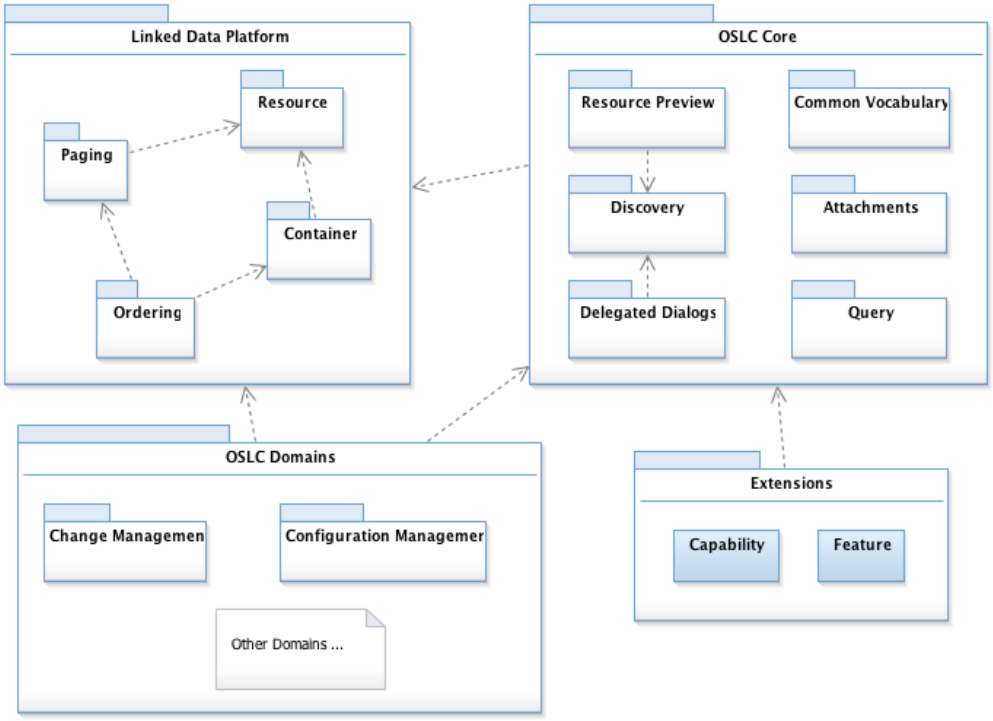

Fig. 2 OSLC Core 3.0 Overview

As seen in Fig. 2 OSLC Core 3.0 Overview, there are a number of capabilities developed in different standards organizations and working groups. The arrows represent either dependencies or extensions to some specifications or capabilities. OSLC domain specifications may depend on OSLC Core 3.0 specifications as scenarios motivate. However, a leading goal is to minimize and eliminate unnecessary dependencies to simplify adoption, which may result in no dependency on OSLC Core 3.0 specifications for some OSLC domains.

This work is an evolution from the OSLC Core 2.0 [OSLCCore2] efforts, taking the experience gained from that effort along with the common foundation on W3C LDP, to produce an updated set of specifications that are simpler, built on layered capabilities and easier to adopt.

1.1 Terminology

Terminology uses and extends the terminology and capabilities of W3C Linked Data Platform [LDP], W3C's Architecture of the World Wide Web [WEBARCH] and Hyper-text Transfer Protocol [HTTP11].

OSLC Server

LDP Server that also supports capabilities defined by at least on OSLC-based specification. See Server [HTTP11] and LDP Server [LDP].

OSLC Client

LDP Client that uses capabilities defined by some OSLC-based specifications. See Client [HTTP11] and LDP Client [LDP]. A particular software component or application could be an OSLC Server supporting a set of domains, and an OSLC Client of other domains depending on its needs.

Domain

A topic area of a specific focus area and/or collection of disciplines. Often OASIS OSLC-affiliated TCs are organized around a domain.

OSLC Core Specifications

Specifications that cover specific capabilities that are often needed across various domains. They are created, authored and endorsed by the OASIS OSLC Open Project. Can be abbreviated to Core Specifications.

OSLC Domain Specifications

Specifications that cover a domain need, including existing open-services.net specifications and new specifications created and authored by OASIS OSLC-affiliated TCs. Can be abbreviated to Domain Specifications

Resource Shape

The set of properties (triples) that constrain a resource for specific operations (i.e. creation, update or query), and for each property, their value types, allowed values and cardinality.

Some industry terms that are often referred to (not exhaustive):

Product Lifecycle Management (PLM)

The process of managing the entire lifecycle of a product from its conception, through design and manufacture, to service and disposal.

Systems Engineering

An interdisciplinary field of engineering that focuses on how to design and manage complex engineering systems over their life cycles.

Application Lifecycle Management (ALM)

The marriage of business management to software engineering made possible by tools that facilitate and integrate requirements management, architecture, coding, testing, tracking, quality and release management.

DevOps

A software development method that stresses communication, collaboration and integration between software developers and Information Technology(IT) professionals in support of continuous delivery.

IT Service Management (ITSM)

The implementation and management of quality information technology services. IT service management is performed by IT service providers through people, process and information technology.

1.1.1 Deprecated terms

Previous revisions of OSLC-based specifications [OSLCCore2], used terminology that may no longer be relevant, accurate or needed any more. Some of those deprecated terms are:

Provider (*deprecated*)

See Server [HTTP11].

Consumer (*deprecated*)

See Client [HTTP11].

1.2 References

1.2.1 Normative references

[ABNF]

D. Crocker; P. Overell. *Augmented BNF for Syntax Specifications: ABNF*. IETF. Internet Standard. URL: https://tools.ietf.org/html/rfc5234

[CORS]

WHATWG contributors. *Fetch standard*. WHATWG. Living Standard. URL: https://fetch.spec.whatwg.org/commit-snapshots/d070ea245c6eb66cf4196324f063dc6ab608d9e6/#http-cors-protocol

[HTTP11]

R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. IETF, June 2014.

Proposed Standard. URL: https://httpwg.org/specs/rfc7230.html

[LDP]

Steve Speicher; John Arwe; Ashok Malhotra. *Linked Data Platform 1.0*. W3C, 26 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/ldp/

[LDPPaging]

S. Speicher; J. Arwe; A. Malhotra. *Linked Data Platform Paging 1.0*. http://www.w3c.org. W3C Working Group Note. URL: https://www.w3.org/TR/ldp-paging/

[OSLCCCM1]

Nick Crossley. *OSLC Configuration Management 1.0*. OASIS. OASIS Working Draft. URL: https://oslc-op.github.io/oslc-specs/specs/config/oslc-config-mgt.html

[OSLCCore2]

S. Speicher; D. Johnson. *OSLC Core 2.0*. http://open-services.net. Finalized. URL: https://open-services.net/bin/view/Main/OslcCoreSpecification

[OSLCQuery]

Jim Amsden; S. Padgett; S. Speicher; David Honey. *OSLC Query Version 3.0*. OASIS. Project Specification. URL: https://docs.oasis-open-projects.org/oslc-op/query/v3.0/oslc-query.html

[OSLCTRS3]

Nick Crossley, Axel Reichwein. *OSLC Tracked Resource Set 3.0*. OASIS. OASIS Working Draft. URL: https://oslc-op.github.io/oslc-specs/specs/trs/tracked-resource-set.html

[OpenIDConnect]

OpenID Connect. openid.net. URL: http://openid.net/connect/

[RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF, March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

[RFC8174]

B. Leiba. *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*. IETF, May 2017. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc8174

[rfc6749]

D. Hardt, Ed.. *The OAuth 2.0 Authorization Framework*. IETF, October 2012. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc6749

1.2.2 Informative references

[HTML5]

Ian Hickson; Robin Berjon; Steve Faulkner; Travis Leithead; Erika Doyle Navara; Theresa O'Connor; Silvia Pfeiffer. *HTML5*. W3C, 27 March 2018. W3C Recommendation. URL: https://www.w3.org/TR/html5/

[LDBestPractices]

Linked Data Best Practices. jazz.net. URL: https://jazz.net/wiki/bin/view/LinkedData/BestPractices

[SHACL]

Holger Knublauch; Dimitris Kontokostas. *Shapes Constraint Language (SHACL)*. http://www.w3.org/. W3C Recommendation. URL: https://www.w3.org/TR/shacl/

[Turtle]

Eric Prud'hommeaux; Gavin Carothers. *RDF 1.1 Turtle*. W3C, 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/turtle/

[WEBARCH]

Ian Jacobs; Norman Walsh. *Architecture of the World Wide Web, Volume One*. W3C, 15 December 2004. W3C Recommendation. URL: https://www.w3.org/TR/webarch/

[rdf11-concepts]

Richard Cyganiak; David Wood; Markus Lanthaler. *RDF 1.1 Concepts and Abstract Syntax*. W3C, 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/rdf11-concepts/

1.3 RDF Namespaces

OSLC Core defines the namespace URI of `http://open-services.net/ns/core#` with a namespace prefix of `oslc`.

OSLC Core uses the following prefixes:

Prefix	Namespace
dcterms	http://purl.org/dc/terms/
foaf	http://xmlns.com/foaf/0.1/
ldp	http://www.w3.org/ns/ldp#
owl	http://www.w3.org/2002/07/owl#
prov	http://www.w3.org/ns/prov#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
vann	http://purl.org/vocab/vann/
vs	http://www.w3.org/2003/06/sw-vocab-status/ns#
xsd	http://www.w3.org/2001/XMLSchema#

1.4 Typographical Conventions and Use of RFC Terms

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this specification are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Goals/Motivation

This section is non-normative.

The primary goal of OSLC is to enable integration of federated, shared information across tools that support different, but related domains. OSLC was initially focused on development of Information Technology (IT) solutions involving processes, activities, work products and supporting tools for Application Lifecycle Management (ALM). However, OSLC capabilities could be applicable to other domains. The specific goals for OSLC Core 3.0 are to build on the existing OSLC Core 2.0 specifications to further facilitate the development and integration of domains and supporting tools that address additional integration needs. Specifically:

- Integration is based on an open standard, and not controlled by any single vendor
- OSLC 3.0 is based on the new W3C Linked Data Platform standard which provides a solid foundation for reading and writing linked data resources
- The specifications are simpler, more consistent and will potentially be more attractive to, and easier to consume by new integrations
- There are some new capabilities including Attachments, inverse link labels, traceability and impact types
- Domain vocabularies can be improved for data consistency and removing data gaps
- All Resource Shapes are provided in machine readable [Turtle] files

The following guiding principles were used to govern the evolution of OSLC and guide the development of the OSLC Core 3.0 specifications.

Scenario-driven

Every capability should be linked back to key integration scenarios that motivate its need. These are important not only for knowing that the correct specification content is being developed but also to assist with implementers understanding the intended usage and in developing relevant test cases.

Incremental

Specifications should be developed in an incremental fashion that not only validates the technical approaches but also delivers integration value sooner.

Loose-coupling

Specifications should support a model where clients have little to no knowledge about server implementation-specific behaviors in order to support key integration scenarios. As a result, clients should be unaffected by any server application software or data model implementation changes. Similarly, client software should be able to be independently changed without changes to server software.

Minimalistic

Specification authors should strive to find not only the simplest solution that would work for a given scenario but allows for easy adoption. Authors should avoid solutions that offer additional capabilities which may inhibit adoption of necessary capabilities.

Capability Based

A capability is the ability to perform actions to achieve outcomes described by scenarios through the use of specific technologies. Capabilities should be incrementally defined as independent focused specifications and independently discoverable at runtime. Even though there may be some generally agreed upon best practices for capability publication and discovery, each capability should define how it is published and discovered. The Core OSLC capabilities are defined in this specification.

Vocabularies

Various OSLC MS-affiliated TCs, or any specification development body that is authoring specifications for specific domains of knowledge, should minimally define vocabularies and the semantics behind the various terms. Some consideration should be given for global reuse when terms are used for cross domain queries and within other domain resource shape definitions. Domain specifications are the definition of an OSLC capability, and how those vocabulary terms are used in LDP interactions by both the clients and servers of that capability. The specification should include defining resource shapes that describe resources based on a set of vocabulary terms, which introduces any domain specific constraints on the vocabulary's usage.

OSLC domain vocabularies should follow the recommended best practices for managing RDF vocabularies described at [LDBestPractices].

3. Architecture

This section is non-normative.

In support of the previously stated goals and motivation, it is desired to have a consistent and recommended architecture. The architecture needs to support scenarios requiring a protocol to access, create, update and delete resources. [LDP] is the foundation for this protocol. Resources need to relate, or link, to one another utilizing a consistent, standard and web-scale data model. Resource Description Framework (RDF) [rdf11-concepts] is the foundation for this. The ability to work with these data models over HTTP protocols, is based on [LDP].

Some scenarios require the need to integrate user interface components: either within a desktop or mobile web-browser, mobile device application, or rich desktop applications. For these scenarios the technology is rapidly evolving and changing. Priority should be based on existing standards such as HTML5, with use of `iframe` and `postMessage()`. [HTML5]

OSLC Core specification documents elaborate on the conformance requirements leveraging these various technologies and approaches.

As the primary goals have been outlined around lifecycle integration, some scenarios may require exploration of new (or different) approaches and technologies. As with all specification development efforts, the OSLC Open Project will advise, develop and approve such efforts.

4. OSLC Core 3.0 Capabilities

This section is non-normative.

The following sections and referenced documents define the capabilities for OSLC Core 3.0. These documents comprise the multi-part specification for OSLC Core 3.0. They represent common capabilities that servers may provide and that may be discovered and used by clients. Although OSLC Core could be useful on its own, it is intended to specify capabilities that are common across many domains. Servers will generally specify conformance with specific domain specifications, and those domain specifications will describe what parts of OSLC Core are required for conformance. This allows servers to implement the capabilities they need in a standard way without the burden of implementing capabilities that are not required. The purpose of the OSLC Core Discovery capability is to allow clients to determine what capabilities are provided by a server. Any provided capability must meet all the conformance criteria for that capability as defined in the OSLC Core 3.0 specifications.

This implies that any capability that is discoverable is essentially optional, and once discovered, the capability is provided as defined in the applicable OSLC specifications. Servers should support OSLC Discovery, but Discovery itself is also an optional capability as servers could provide other means of informing specific clients of supported OSLC capabilities that could be utilized directly. For example, a server might provide only preview dialogs on specific resources and nothing else.

4.1 Resource Constraints

The shape of an RDF resource is a description of the set of triples it is expected to contain and the integrity constraints those triples are required to satisfy. Applications of shapes include validating RDF data, documenting RDF APIs, and providing metadata to tools that handle RDF data such as form and query builders.

Shapes are different than vocabularies in that shapes may change with new revisions of resource definitions, whereas vocabularies should evolve in place in a compatible manner.

Constraints on OSLC Core and Domain resources **SHOULD** be described using OSLC Core Version 3.0. Part 6: Resource Shape which is included as part of the OSLC Core multi-part specifications. Servers **MAY** use other constraint languages such as [SHACL] to define resource constraints. [core-1]

OSLC Domain specifications **SHOULD** use the following URI pattern when publishing each individual resource shape: [core-2]

```
http://open-services.net/ns/[vocab-short-name]/shapes/[version]#[shape-name]
```

For example, for Change Management 3.0, a shape describing the base Change Request resource type might have the shape URI:

```
http://open-services.net/ns/cm/shapes/3.0#ChangeRequestShape
```

Not all shapes would necessarily be updated at the same time.

To allow different versions of individual shapes to be reused in different versions of a domain specification while still allowing a client to browse the set of possible shapes, domains **SHOULD** provide an resource for all the shapes for a spec version, at a URI defined by the following pattern:

```
http://open-services.net/ns/[vocab-short-name]/shapes/[SPEC-version]
```

[core-3]

For example, for Change Management, there should be a resource at: `http://open-services.net/ns/cm/shapes` with members such as:

```
http://open-services.net/ns/cm/shapes/3.0#TaskShape
http://open-services.net/ns/cm/shapes/3.1#WeaknessShape
http://open-services.net/ns/cm/shapes/2.0#ChangeRequestShape
```

4.1.1 Authentication

Authentication determines how a user of a client identifies themselves to a server to ensure the user has sufficient privileges to access resources from that server, and provides a mechanism for servers to control access to resources.

OSLC 3.0 servers **MAY** protect resources with HTTP Basic Authentication. OSLC Services that use HTTP Basic Authentication **SHOULD** do so only via SSL. [core-4]

OSLC 3.0 servers **SHOULD** protect resources with [rfc6749] Authentication utilizing [OpenIDConnect]. [core-5]

4.1.2 Resource Discovery

Resource Discovery defines a common approach for HTTP/LDP-based servers to be able to publish their RESTful API capabilities and how clients can discover and use them.

4.1.3 Resource Representations

OSLC resource representations come in many forms and are subject to standard HTTP and mechanisms for content negotiation.

OSLC domain specifications specify the representations needed for the specific scenarios that they are addressing, and should recognize that different representations are appropriate for different purposes. For example, browser oriented scenarios might be best addressed by JSON or Atom format representations.

OSLC domain specifications are also expected to follow common practices and conventions that are in concert with existing industry standards and which offer consistency across domains. All of the OSLC specifications are built upon the standard RDF data model, allowing OSLC to align with the Linked-Data Platform [LDP]. In addition, all OSLC specifications have adopted the convention to illustrate most examples using Turtle and/or JSON-LD representations and will typically require these representations to enable consistency across OSLC implementations.

OSLC Services **MUST** support at least one RDF resource serialization format, and **SHOULD** support as many serialization formats as possible through content negotiation. [core-6]

OSLC Services **SHOULD** provide and accept RDF documents in Turtle format (identified by the MIME-type 'text/turtle') and in JSON-LD format (identified by the MIME-type 'application/ld+json') representations for each OSLC resource for compatibility with LDP 1.0. [core-7]

OSLC Services **SHOULD** provide and accept RDF/XML representations for each OSLC resource to preserve compatibility with [OSLCCore2]. [core-8]

OSLC Services **MAY** provide and accept existing standard or emerging standard formats such as XML, HTML, and the Atom Syndication Format. [core-9]

OSLC servers **SHOULD** support the Accept header and whenever possible, respond with one of the content types requested by the client. When the Accept header requests an unsupported RDF serialization, the OSLC server **SHOULD** return an RDF serialization format that it does support, indicating what it is in the Content-Type header. If the server gets an Accept header for some non-RDF content type, say ATOM, that it does not support, then it **SHOULD** return 406 Unacceptable. [core-10]

OSLC servers **SHOULD** support the CORS protocol [CORS] in order to allow browser-based OSLC clients to interact with the server. [core-11] In addition to following the general practice, server implementers are recommended to apply the recommendations listed below:

- OSLC servers **SHOULD** use the `Access-Control-Allow-Headers` header [CORS] to allow the use of the `Content-Type` header in OSLC requests. Despite the `Content-Type` header being a CORS-safelisted request header, it needs to be explicitly allowed in order to expand the range of its permitted values and allow OSLC clients to send RDF contents to the server. [core-12]
- OSLC servers **SHOULD** use the `Access-Control-Allow-Headers` header [CORS] to allow the use of the `OSLC-Core-Version` header in OSLC requests. [core-13]

4.1.4 Common Vocabulary

Common Vocabulary Terms defines a number of commonly used RDF vocabulary terms and resources (shapes), that have broad applicability across various domains.

4.1.5 Resource Operations

Resource Operations specify how clients create, read, update and delete resources managed by servers.

OSLC Core defines a set of HTTP REST services for a set of capabilities that facilitate flexible, loosely coupled tool integration. These services may be augmented by varous OSLC domain specifications that specify the capabilities, vocabularies and constraints that define specific resources managed by these integration capabilities.

OSLC Servers **MUST** implement standard HTTP protocols and **MUST** at least support `GET` requests that respond with resources in some RDF serialization format. [core-14]

OSLC Services use HTTP for create, retrieve, update and delete operations on resources. OSLC Services **MUST** comply with the HTTP specification [HTTP11]. [core-15]

Because the update process may involve first getting a resource, modifying it and then later putting it back to the server, there is the possibility of a conflict, e.g. some other client may have have updated the resource since the GET. To mitigate this problem, OSLC Services **SHOULD** use the HTTP `If-Match` header on a PUT request: [core-16]

- If the HTTP `If-Match` header is missing OSLC Services **SHOULD** return HTTP Bad Request (400) status code to indicate that the header is required. [core-17]
- If the HTTP `If-Match` header is present OSLC Services **MUST** behave as described in the HTTP specification, returning an HTTP Precondition Failed (412) error to indicate that the header does not match. [core-18]
- If the HTTP `If-Match` header is present and it matches, but there is some other problem or conflict with the update then OSLC Services **MAY** return an HTTP Conflict (409) to indicate that problem. [core-19]

An OSLC Service **SHOULD** preserve property values that are not part of the resource definition or Resource Shape known by the server (unknown property values). An OSLC Service **MUST** return a 4xx status code if it decides not to persist any of the unknown property values (in accordance with the LDP specification §4.2.4.4). [core-20]

An OSLC Client **MUST** preserve any unknown property values between requests to the OSLC Services for all HTTP verbs except PATCH (in accordance with the LDP specification §4.3.1.11). [core-21]

4.1.6 Selective Properties

OSLC Services **MAY** support a technique called Selective Properties to enable clients to retrieve only selected property values. [core-22]

By adding the key=value pair `oslc.properties`, specified below, to a resource URI, a client can request a new resource with a subset of the original resource's values. An additional key=value pair `oslc.prefix` can be used to define prefixes used to identify the selected properties.

The `oslc.properties` key=value pair lets you specify the set of properties to be included in the response. Both immediate and nested properties may be specified. A nested property is a property that belongs to the resource referenced by another property. Nested properties are enclosed in brace brackets, and this nesting may be done recursively, i.e. a nested property may contain other nested properties.

For example, suppose we have a bug report resource at the following URL:

> EXAMPLE 1: Bug Report URL
>
> `http://example.com/bugs/4242`

Suppose this bug resource has properties such as `dcterms:title`, `dcterms:description`, and `dcterms:creator`, and that `dcterms:creator` refers to a person resource that has properties such as `foaf:givenName` and `foaf:familyName`. Suppose you want a representation of the bug report that includes its `dcterms:title` and the `foaf:givenName` and `foaf:familyName` of the person referred to by its `dcterms:creator`. The following URL illustrates the use of the `oslc.properties` query value to include those properties:

> EXAMPLE 2: Selective Properties URL
>
> `http://example.com/bugs/4242?oslc.properties=dcterms:title,dcterms:creator{foaf:givenName,foaf:familyName}`

The `oslc.properties` pair is defined by the oslc_properties term in the following BNF grammar:

```
oslc_properties ::= "oslc.properties=" properties
properties      ::= property ("," property)*
property        ::= identifier | wildcard | nested_prop
nested_prop     ::= (identifier | wildcard) "{" properties "}"
wildcard        ::= "*"
identifier      ::= PrefixedName
PrefixedName    ::= /* see "SPARQL Query Language for RDF", http://www.w3.org/TR/rdf-sparql-query/#rPrefixedName */
```

In our examples of `oslc.properties`, property names include a URI prefix, i.e. dcterms: or foaf:. Here we assume that OSLC has predefined the Dublin Core (dcterms:) and Friend of a Friend (foaf:) prefixes. However, OSLC resources should also be open to new content, which means that new properties may not have predefined URI prefixes. We therefore need a way to define new URI prefixes in resource requests.

The oslc.prefix value lets you specify URI prefixes used in property names. For example, suppose the foaf: prefix was not predefined. The following URL illustrates the use of the oslc.prefix value to define it:

> EXAMPLE 3: Example URL with oslc.prefix
>
> `http://example.com/bugs/4242?oslc.prefix=foaf=<http://xmlns.com/foaf/0.1/>&oslc.properties=foaf:lastName,...`

The syntax of the **`oslc.prefix`** is defined by the oslc_prefix term in the following BNF grammar:

```
oslc_prefix ::= "oslc.prefix=" prefix_defs
prefix_defs ::= prefix_def ("," prefix_def)*
prefix_def  ::= prefix "=" uri_ref_esc
prefix      ::= PN_PREFIX
PN_PREFIX   ::= /* see "SPARQL Query Language for RDF", http://www.w3.org/TR/rdf-sparql-query/#rPN_PREFIX */
uri_ref_esc ::= /* an angle bracket-delimited URI reference in which > and \ are \-escaped. */
```

OSLC Core specifies a number of predefined PrefixDefinitions for convenience. OSLC Domain specifications may specify additional pre-defined PrefixDefinitions for their purposes.

An OSLC Server **SHOULD** support the following PrefixDefinitions:

- dcterms: <http://purl.org/dc/terms/>
- foaf: <http://xmlns.com/foaf/0.1/>
- owl: <http://www.w3.org/2002/07/owl#>
- rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- xsd: <http://www.w3.org/2001/XMLSchema#>
- rdfs: <http://www.w3.org/2000/01/rdf-schema#>
- ldp: <http://www.w3.org/ns/ldp#>
- oslc: <http://open-services.net/ns/core#>
- trs: <http://open-services.net/ns/core/trs#>

[core-23]

4.1.7 Resource Preview

Resource Preview specifies a technique to get a minimal HTML representation of a resource identified by a URL. Applications often use this representation to display a link with an appropriate icon, a label, or display a small or large preview when a user makes some gesture over a link.

4.1.8 Delegated Dialogs

Delegated Dialogs allow one application to embed a creation or selection UI into another using HTML **`iframe`** elements and JavaScript code. The embedded dialog notifies the parent page of events using HTML5 **`postMessage`**.

4.1.9 Query

OSLC servers will often manage large amounts of potentially complex link-data entities. Practical use of this information will require some query capability that minimally supports selection of matching elements, filtering of desired properties and ordering. OSLC Core defines a query capability that is relatively simple, can be implemented on a wide range of existing server architectures, and provides a standard, data source independent query mechanism. The purpose of this query capability is to support tool integration through a common query mechanism.

OSLC Servers **MAY** support a *Query Capability* as defined in [OSLCQuery] to enable clients to perform selection and projection operations in order to retrieve a selected subset of resources and property values from an LDPC. [core-24]

4.1.10 Resource Paging

When a client requests a resource, the client should expect that the entire resource will be returned in the response, with all property values. This can be problematic because, in some cases, resources may be so large that a client might not want to retrieve the entire resource in one HTTP response.

One solution for clients that are sensitive to response size is to check the response size before loading. This method is described in the next section. Another solution is to use Resource Paging.

Resource Paging specifies a capability for servers to make the state of large resources or long lists of resources available as a list of smaller subset resources (pages) whose representation is easier to produce by the server and consume by clients. Resource paging is particularly useful in handling results from the query capability or the contents of an LDP container.

4.1.10.1 Checking Response Size

Clients that do not wish to load large resources may use the HTTP HEAD method to determine the size of a resource.

When responding to an HTTP HEAD request, according to [HTTP11] the server **SHOULD** include an HTTP Content-Length header that indicates the size of the resource as the "decimal number of octets." [core-25]

4.1.10.2 Paging Stability

Because HTTP is a stateless protocol and OSLC Services manage resources that can change frequently, OSLC clients should assume that the resources returned on a given page are unstable and may change over time.

Some OSLC Servers might wish to guarantee stable paging, meaning that the sequence of pages returned from the server represent a snapshot in time and will not change as the client pages through them. OSLC specifications that require stable paging should state this requirement and specify to which resources it applies.

Note that because stable paging implementations are based on server-side state, it is possible that the state will expire. Implementations **MAY** use the HTTP response code 410 (Expired) to indicate to clients that the link they requested has expired. [core-26]

4.1.10.3 Response Information

A response info resource representation describes information about a paged HTTP response body in which it appears.

Resource representations returned via Resource Paging **MUST** include a resource of type `oslc:ResponseInfo`, as defined in part 7 of this multi-part specification: Vocabulary. [core-27]

The subject resource URI of the `oslc:ResponseInfo` resource representation **MUST** be the HTTP request URI or the URI from subsequent redirects. The response representation **MAY** also include properties from subject resources different from the one identified by the request URI. [core-28]

When responding to a request that includes `oslc.pageSize` in the URI, a server **MAY** divide the response into a number of pages and use the value as a hint about the maximum number of RDF statements to be included in each page. Servers **MAY** return pages containing more or fewer RDF statements than specified. [core-29]

Below is an example using the OSLC Core RDF/XML representation guidance, that illustrates how the `oslc:ResponseInfo` resource representation is included in addition to the blog entry resource representation.

By adding `oslc.pageSize` to the query component of the resource URI, the client requests that the server break each response into pages using the specified value as a hint for maximum number of RDF statements in each page. Clients should make no assumptions about the maximum number of RDF statements returned in a page.

> EXAMPLE 4: Resource Paging, partial response with response info resource representation

```
GET http://example.com/blogs/entry/1?oslc.paging=true&amp;pageSize=200&amp;pageno=2

<rdf:RDF
   xmlns:oslc_blog="http://open-services.net/ns/bogus/blogs#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf="http://http://xmlns.com/foaf/0.1/"
   xmlns:dcterms="http://purl.org/dc/terms/">

   <oslc_blog:Entry
      rdf:about="http://example.com/blogs/entry/1">
      &lt;!-- partial property values of of the blog entry --&gt;
   </oslc_blog:Entry>

   <oslc:ResponseInfo rdf:about="http://example.com/blogs/entry/1?oslc.paging=true&amp;pageSize=200&amp;pageno=2">
      <oslc:nextPage rdf:resource="http://example.com/blogs/entry/1?oslc.paging=true&amp;pageSize=200&amp;pageno=3" />
   </oslc:ResponseInfo>
```



```
</rdf:RDF>
```

Refer to the OSLC vocabulary specification for further information on the ResponseInfo resource.

4.1.10.4 Using Resource Paging

OSLC Services **MAY** support [LDPPaging] to enable clients to retrieve large LDP resources (LDPRs) a page at a time. [core-30]

OSLC Services **SHOULD** support OSLC paging as described in this section to ensure compatibility with OSLC 2.0 server implementations. [core-31]

To request a paged version of a resource, a client **MUST** add at least one "key=value" pair to the query component of the resource URI: Either oslc.paging=true, or oslc.pageSize, or both. [core-32]

When responding to a request that includes oslc.paging=true in the URI, a server **MAY** return a representation that contains a subset of the resource's property values. [core-33]

By adding oslc.pageSize to the query component of the resource URI, the client requests that the server respond with a specific number of property values. For example, oslc.pageSize=20 indicates to the server that the client would like 20 values per page.

When responding to a request that includes oslc.pageSize in the URI, a server **SHOULD** return a representation that contains the requested number of property values. [core-34]

When Resource Paging is used, the values of a multi-valued property **MAY** be split across resource pages. [core-35]

When Resource Paging is used, each property value **MUST** be represented in its entirety and not split across multiple partial resource pages. [core-36]

When a page is returned and it is NOT the last page in the sequence, then it **SHOULD** include an oslc:ResponseInfo (see above), which contains a resource-valued property oslc:nextPage that links to a resource that represents the next page of property-values. [core-37]

When paging is unstable, by the time a client follows an oslc:nextPage link there may no longer be a next page, in this case the server **MAY** respond with an HTTP 404 Page Not Found status code. [core-38]

4.1.10.5 Provider Response Size Limitations

When a client requests a resource that an OSLC Service considers to be too large to return in one response and the client has not indicated that it desires paging (via oslc.paging or oslc.pageSize), the OSLC Service **MAY** redirect the client to a representation that contains partial information about the resource. [core-39]

When the OSLC Service opts to redirect the client to a partial representation, it **MUST** return an HTTP Status 302 redirect to a URL that includes either oslc.paging, or oslc.pageSize, or both. [core-40]

If the URL includes oslc.pageSize then the value should be a value that is acceptable to the service. The client may choose to follow the redirect and receive a representation that contains partial information about the resource.

On receiving a resource representation, OSLC Clients should check for the presence of an oslc:nextPage value to determine if the representation contains partial information about the resource. If the value is present, then paging is in effect and the representation contains partial information about the resource.

4.1.11 Attachments

Attachments describes a minimal way to manage attachments related to web resources using LDP-Containers and Non-RDF Source [LDP].

4.1.12 Tracked Resource Sets

OSLC defines a Tracked Resource Set capability that allows servers to expose a set of resources in a way that enables clients to discover the exact set of resources in the set, to track ongoing changes affecting resources in the set. This allows OSLC servers to expose a live feed of linked data in a way that permits clients to build, aggregate, and maintain live, searchable information based on that linked data.

OSLC Servers **MAY** support a *Tracked Resources Set* capability as defined in [OSLCTRS3] to enable OSLC data consumers and providers flexible ways of sharing information. [core-41]

4.1.13 Configuration Management

OSLC defines a Configuration Management capability for managing versions and configurations of linked data resources from multiple domains. Using client and server applications that implement the configuration management capability, a team administrator can create configurations of versioned resources contributed from tools and data sources across the lifecycle. These contributions can be assembled into aggregate (global) configurations that are used to resolve references to artifacts in a particular and reproducible context.

OSLC Clients and Servers **MAY** support a *Configuration Management* capability as defined in [OSLCCCM1]. [core-42]

4.1.14 Error Responses

Error responses returned by servers in response to requests are defined in Common Vocabulary Terms, Errors.

4.2 Version Compatibility

4.2.1 Overview

This section is non-normative.

OSLC is intended to provide a foundation for (lifecycle) application interoperability. A significant number of OSLC domains, and client and server implementations already exist and are in common use. Interoperability issues between applications on incompatible OSLC versions could result in negative impact to end users. One of the goals of the OSLC initiative is to mitigate or eliminate the need for lock-step version upgrades, where clients or servers target one version of a specification and break when new versions are introduced -- requiring all services to be upgraded simultaneously.

This section introduces a capability for the clients and servers to pass an OSLC version. However, exposing version numbers in OSLC implementations could lead to interoperability issues. Ultimately each domain needs to decide its compatibility needs.

4.2.2 Versioning support in clients and servers

OSLC Clients and Servers **SHOULD** use the `OSLC-Core-Version` header to indicate what OSLC version they expect or support. [core-43] When returning an RDF response, an OSLC Server **MUST** return the `OSLC-Core-Version` header set to the Core specification with which the representation complies. [core-44]

The `OSLC-Core-Version` header consists of a major and minor version components and **MUST** conform to the following ABNF grammar [ABNF]:

```
SP = " "
HTAB = %x09
WSP = SP / HTAB
DIGIT = %x30-39

header = oslc-version
MAJOR = 1*DIGIT
MINOR = 1*DIGIT
oslc-version = "OSLC-Core-Version:" *WSP MAJOR "." MINOR
```

[core-45]

Example:

```
OSLC-Core-Version: 3.0
```

The major version component in the `OSLC-Core-Version` header **MUST** be greater or equal to "2". [core-46] An OSLC Server **SHOULD** reject a request with the `OSLC-Core-Version` header that has a major version value below "2" with a response 400 Bad Request. [core-47]

If the `OSLC-Core-Version` header is present in the request and set to a version that an OSLC Server can support, then the Server **MUST** return a representation that is complies with the specified version. [core-48] If the `OSLC-Core-Version` header is present and indicates a specification version that the Server cannot support, the Server **SHOULD** respond with what it determines is the most compatible version that it can return; however, the server **MAY** reject the request with 400 Bad Request. [core-49] If the `OSLC-Core-Version` header is not present then the Server **SHOULD** respond by returning a resource that conforms to the earliest or most compatible (as determined by the implementation) specification version's representation. [core-50]

OSLC Core 3.0 Servers that comply with OSLC Core 2.0 requirements **MAY** continue to identify themselves as OSLC Core 2.0 servers with

the `OSLC-Core-Version: 2.0` headers in the response. [core-51]

If the `OSLC-Core-Version` header is not present, an OSLC Server **MAY** return an OSLC 1.0 response. [core-52]OSLC Clients **SHOULD** supply the `OSLC-Core-Version: 2.0` header (or newer version) in the request to avoid an OSLC 1.0 response. [core-53]

Machine-readable content **SHALL** have precedence over specification text if any discrepancy is discovered. [core-54] It is worth noting that in OSLC 2.0 specifications, normative specification text had precedence over machine-readable content, as opposed to this specification.

4.3 Conformance

OSLC Servers **MUST** implement all the mandatory requirements in the normative sections of specification, and **SHOULD** follow all the guidelines and recommendations in these specifications. [core-55]

OSLC Servers **MAY** implement any of the capabilities described in this specification, and if a capability is supported, all of the mandatory requirements specified in the normative sections of the multi-part specification elaborating the capability **MUST** be implemented. [core-56]

OSLC Servers **MUST** implement the OSLC Core vocabulary as defined in OSLC Core Version 3.0. Part 7: Vocabulary. [core-57]

Part 2 of this document defines the mandatory requirements for the OSLC discovery capability. OSLC Servers **SHOULD** provide the discovery capability in order for clients to determine what services are supported by the server and how to access them. OSLC Servers that implement discovery **MUST** implement all of the mandatory requirements specified in the normative sections of part 2 of this multi-part specification. [core-58]

Part 3 of this document defines the mandatory requirements for the OSLC resource preview capability. OSLC Servers that implement resource preview **MUST** implement all of the mandatory requirements specified in the normative sections of part 3 of this multi-part specification. [core-59]

Part 4 of this document defines the mandatory requirements for the OSLC delegated creation and selection dialog capability. OSLC Servers that implement delegated dialogs **MUST** implement all of the mandatory requirements specified in the normative sections of part 4 of this multi-part specification. [core-60]

Part 5 of this document defines the mandatory requirements for the OSLC attachments capability. OSLC Servers that implement attachments **MUST** implement all of the mandatory requirements specified in the normative sections of part 5 of this multi-part specification. [core-61]

Part 6 of this document defines the mandatory requirements for the OSLC resource shape constraints capability. OSLC Servers **SHOULD** support resource shapes in order to describe OSLC server managed resources, and validate creation and update requests. OSLC Servers that implement resources shapes **MUST** implement all of the mandatory requirements specified in the normative sections of part 6 of this multi-part specification. [core-62]

All conformance clauses are summarized in the following table.

Clause Number	Requirement
core-1	Constraints on OSLC Core and Domain resources **SHOULD** be described using OSLC Core Version 3.0. Part 6: Resource Shape which is included as part of the OSLC Core multi-part specifications. Servers **MAY** use other constraint languages such as [SHACL] to define resource constraints.
core-2	OSLC Domain specifications **SHOULD** use the following URI pattern when publishing each individual resource shape:
core-3	To allow different versions of individual shapes to be reused in different versions of a domain specification while still allowing a client to browse the set of possible shapes, domains **SHOULD** provide an resource for all the shapes for a spec version, at a URI defined by the following pattern: `http://open-services.net/ns/[vocab-short-name]/shapes/[SPEC-version]`
core-4	OSLC 3.0 servers **MAY** protect resources with HTTP Basic Authentication. OSLC Services that use HTTP Basic Authentication **SHOULD** do so only via SSL.
core-5	OSLC 3.0 servers **SHOULD** protect resources with [rfc6749] Authentication utilizing [OpenIDConnect].
core-6	OSLC Services **MUST** support at least one RDF resource serialization format, and **SHOULD** support as many serialization formats as possible through content negotiation.

Clause Number	Requirement
core-7	OSLC Services **SHOULD** provide and accept RDF documents in Turtle format (identified by the MIME-type 'text/turtle') and in JSON-LD format (identified by the MIME-type 'application/ld+json') representations for each OSLC resource for compatibility with LDP 1.0.
core-8	OSLC Services **SHOULD** provide and accept RDF/XML representations for each OSLC resource to preserve compatibility with [OSLCCore2].
core-9	OSLC Services **MAY** provide and accept existing standard or emerging standard formats such as XML, HTML, and the Atom Syndication Format.
core-10	OSLC servers **SHOULD** support the Accept header and whenever possible, respond with one of the content types requested by the client. When the Accept header requests an unsupported RDF serialization, the OSLC server **SHOULD** return an RDF serialization format that it does support, indicating what it is in the Content-Type header. If the server gets an Accept header for some non-RDF content type, say ATOM, that it does not support, then it **SHOULD** return 406 Unacceptable.
core-11	OSLC servers **SHOULD** support the CORS protocol [CORS] in order to allow browser-based OSLC clients to interact with the server.
core-12	OSLC servers **SHOULD** use the `Access-Control-Allow-Headers` header [CORS] to allow the use of the `Content-Type` header in OSLC requests. Despite the `Content-Type` header being a CORS-safelisted request header, it needs to be explicitly allowed in order to expand the range of its permitted values and allow OSLC clients to send RDF contents to the server.
core-13	OSLC servers **SHOULD** use the `Access-Control-Allow-Headers` header [CORS] to allow the use of the `OSLC-Core-Version` header in OSLC requests.
core-14	OSLC Servers **MUST** implement standard HTTP protocols and **MUST** at least support `GET` requests that respond with resources in some RDF serialization format.
core-15	OSLC Services use HTTP for create, retrieve, update and delete operations on resources. OSLC Services **MUST** comply with the HTTP specification [HTTP11].
core-16	Because the update process may involve first getting a resource, modifying it and then later putting it back to the server, there is the possibility of a conflict, e.g. some other client may have have updated the resource since the GET. To mitigate this problem, OSLC Services **SHOULD** use the HTTP `If-Match` header on a PUT request:
core-17	If the HTTP `If-Match` header is missing OSLC Services **SHOULD** return HTTP Bad Request (400) status code to indicate that the header is required.
core-18	If the HTTP `If-Match` header is present OSLC Services **MUST** behave as described in the HTTP specification, returning an HTTP Precondition Failed (412) error to indicate that the header does not match.
core-19	If the HTTP `If-Match` header is present and it matches, but there is some other problem or conflict with the update then OSLC Services **MAY** return an HTTP Conflict (409) to indicate that problem.
core-20	An OSLC Service **SHOULD** preserve property values that are not part of the resource definition or Resource Shape known by the server (unknown property values). An OSLC Service **MUST** return a 4xx status code if it decides not to persist any of the unknown property values (in accordance with the LDP specification §4.2.4.4).
core-21	An OSLC Client **MUST** preserve any unknown property values between requests to the OSLC Services for all HTTP verbs except PATCH (in accordance with the LDP specification §4.3.1.11).
core-22	OSLC Services **MAY** support a technique called Selective Properties to enable clients to retrieve only selected property values.
core-23	An OSLC Server **SHOULD** support the following PrefixDefinitions: • dcterms: <http://purl.org/dc/terms/> • foaf: <http://xmlns.com/foaf/0.1/> • owl: <http://www.w3.org/2002/07/owl#> • rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> • xsd: <http://www.w3.org/2001/XMLSchema#> • rdfs: <http://www.w3.org/2000/01/rdf-schema#> • ldp: <http://www.w3.org/ns/ldp#> • oslc: <http://open-services.net/ns/core#> • trs: <http://open-services.net/ns/core/trs#>

Clause Number	Requirement
core-24	OSLC Servers **MAY** support a *Query Capability* as defined in [OSLCQuery] to enable clients to perform selection and projection operations in order to retrieve a selected subset of resources and property values from an LDPC.
core-25	When responding to an HTTP HEAD request, according to [HTTP11] the server **SHOULD** include an HTTP Content-Length header that indicates the size of the resource as the "decimal number of octets."
core-26	Implementations **MAY** use the HTTP response code 410 (Expired) to indicate to clients that the link they requested has expired.
core-27	Resource representations returned via Resource Paging **MUST** include a resource of type `oslc:ResponseInfo`, as defined in part 7 of this multi-part specification: Vocabulary.
core-28	The subject resource URI of the `oslc:ResponseInfo` resource representation **MUST** be the HTTP request URI or the URI from subsequent redirects. The response representation **MAY** also include properties from subject resources different from the one identified by the request URI.
core-29	When responding to a request that includes `oslc.pageSize` in the URI, a server **MAY** divide the response into a number of pages and use the value as a hint about the maximum number of RDF statements to be included in each page. Servers **MAY** return pages containing more or fewer RDF statements than specified.
core-30	OSLC Services **MAY** support [LDPPaging] to enable clients to retrieve large LDP resources (LDPRs) a page at a time.
core-31	OSLC Services **SHOULD** support OSLC paging as described in this section to ensure compatibility with OSLC 2.0 server implementations.
core-32	To request a paged version of a resource, a client **MUST** add at least one "key=value" pair to the query component of the resource URI: Either oslc.paging=true, or oslc.pageSize, or both.
core-33	When responding to a request that includes oslc.paging=true in the URI, a server **MAY** return a representation that contains a subset of the resource's property values.
core-34	When responding to a request that includes oslc.pageSize in the URI, a server **SHOULD** return a representation that contains the requested number of property values.
core-35	When Resource Paging is used, the values of a multi-valued property **MAY** be split across resource pages.
core-36	When Resource Paging is used, each property value **MUST** be represented in its entirety and not split across multiple partial resource pages.
core-37	When a page is returned and it is NOT the last page in the sequence, then it **SHOULD** include an oslc:ResponseInfo (see above), which contains a resource-valued property oslc:nextPage that links to a resource that represents the next page of property-values.
core-38	When paging is unstable, by the time a client follows an oslc:nextPage link there may no longer be a next page, in this case the server **MAY** respond with an HTTP 404 Page Not Found status code.
core-39	When a client requests a resource that an OSLC Service considers to be too large to return in one response and the client has not indicated that it desires paging (via oslc.paging or oslc.pageSize), the OSLC Service **MAY** redirect the client to a representation that contains partial information about the resource.
core-40	When the OSLC Service opts to redirect the client to a partial representation, it **MUST** return an HTTP Status 302 redirect to a URL that includes either oslc.paging, or oslc.pageSize, or both.
core-41	OSLC Servers **MAY** support a *Tracked Resources Set* capability as defined in [OSLCTRS3] to enable OSLC data consumers and providers flexible ways of sharing information.
core-42	OSLC Clients and Servers **MAY** support a *Configuration Management* capability as defined in [OSLCCCM1].
core-43	OSLC Clients and Servers **SHOULD** use the `OSLC-Core-Version` header to indicate what OSLC version they expect or support.
core-44	When returning an RDF response, an OSLC Server **MUST** return the `OSLC-Core-Version` header set to the Core specification with which the representation complies.

Clause Number	Requirement

core-45	The `OSLC-Core-Version` header consists of a major and minor version components and **MUST** conform to the following ABNF grammar [ABNF]:
	
	`SP = " "`
	`HTAB = %x09`
	`WSP = SP / HTAB`
	`DIGIT = %x30-39`
	
	`header = oslc-version`
	`MAJOR = 1*DIGIT`
	`MINOR = 1*DIGIT`
	`oslc-version = "OSLC-Core-Version:" *WSP MAJOR "." MINOR`

core-46	The major version component in the `OSLC-Core-Version` header **MUST** be greater or equal to "2".
core-47	An OSLC Server **SHOULD** reject a request with the `OSLC-Core-Version` header that has a major version value below "2" with a response 400 Bad Request.
core-48	If the `OSLC-Core-Version` header is present in the request and set to a version that an OSLC Server can support, then the Server **MUST** return a representation that is complies with the specified version.
core-49	If the `OSLC-Core-Version` header is present and indicates a specification version that the Server cannot support, the Server **SHOULD** respond with what it determines is the most compatible version that it can return; however, the server **MAY** reject the request with 400 Bad Request.
core-50	If the `OSLC-Core-Version` header is not present then the Server **SHOULD** respond by returning a resource that conforms to the earliest or most compatible (as determined by the implementation) specification version's representation.
core-51	OSLC Core 3.0 Servers that comply with OSLC Core 2.0 requirements **MAY** continue to identify themselves as OSLC Core 2.0 servers with the `OSLC-Core-Version: 2.0` headers in the response.
core-52	If the `OSLC-Core-Version` header is not present, an OSLC Server **MAY** return an OSLC 1.0 response.
core-53	OSLC Clients **SHOULD** supply the `OSLC-Core-Version: 2.0` header (or newer version) in the request to avoid an OSLC 1.0 response.
core-54	Machine-readable content **SHALL** have precedence over specification text if any discrepancy is discovered.
core-55	OSLC Servers **MUST** implement all the mandatory requirements in the normative sections of specification, and **SHOULD** follow all the guidelines and recommendations in these specifications.
core-56	OSLC Servers **MAY** implement any of the capabilities described in this specification, and if a capability is supported, all of the mandatory requirements specified in the normative sections of the multi-part specification elaborating the capability **MUST** be implemented.
core-57	OSLC Servers **MUST** implement the OSLC Core vocabulary as defined in OSLC Core Version 3.0. Part 7: Vocabulary.
core-58	Part 2 of this document defines the mandatory requirements for the OSLC discovery capability. OSLC Servers **SHOULD** provide the discovery capability in order for clients to determine what services are supported by the server and how to access them. OSLC Servers that implement discovery **MUST** implement all of the mandatory requirements specified in the normative sections of part 2 of this multi-part specification.
core-59	Part 3 of this document defines the mandatory requirements for the OSLC resource preview capability. OSLC Servers that implement resource preview **MUST** implement all of the mandatory requirements specified in the normative sections of part 3 of this multi-part specification.
core-60	Part 4 of this document defines the mandatory requirements for the OSLC delegated creation and selection dialog capability. OSLC Servers that implement delegated dialogs **MUST** implement all of the mandatory requirements specified in the normative sections of part 4 of this multi-part specification.
core-61	Part 5 of this document defines the mandatory requirements for the OSLC attachments capability. OSLC Servers that implement attachments **MUST** implement all of the mandatory requirements specified in the normative sections of part 5 of this multi-part specification.
core-62	Part 6 of this document defines the mandatory requirements for the OSLC resource shape constraints capability. OSLC Servers **SHOULD** support resource shapes in order to describe OSLC server managed resources, and validate creation and update requests. OSLC Servers that implement resources shapes **MUST** implement all of the mandatory requirements specified in the normative sections of part 6 of this multi-part specification.

4.4 Acknowledgements

This section is non-normative.

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

James Amsden, IBM (Chair)
Nick Crossley, IBM
Jad El-khoury, KTH Royal Institute of Technology
Ian Green, IBM
David Honey, IBM
Jean-Luc Johnson, Airbus Group SAS
Harish Krishnaswamy, Software AG, Inc.
Arnaud LeHors, IBM
Sam Padget, IBM
Martin Pain, IBM
Arthur Ryman, IBM
Martin Sarabura, PTC (Chair)
Steve Speicher, IBM

4.5 Change History

This section is non-normative.

Revision	Date	Editor	Changes Made
CSD03	31 May 2018	Jim Amsden	Added predefined prefixes for common namespaces. Relaxed RDF serialization format requirements. Added normative references to OSLC Query, TRS and Configuration Management specifications.
PSD04	20 December 2019	Jim Amsden	Added conformance section and migrated to Open Project
PS01	17 September 2020	Jim Amsden	Added vocabulary and constraints documents to additional content. Recommendations on URI patterns for published shapes now includes fragments. See Milestone Core v3.0 PS 01 for additional information.

OASIS OPEN PROJECTS

OSLC Core Version 3.0. Part 2: Discovery

OASIS Standard
26 August 2021

This stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/discovery.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/discovery.pdf

Previous stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/discovery.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/discovery.pdf
(published as Project Specification)

Latest stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/discovery.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/discovery.pdf

Latest version:

https://open-services.net/spec/core/latest

Latest editor's draft:

https://open-services.net/spec/core/latest-draft

Open Project:

OASIS Open Services for Lifecycle Collaboration (OSLC) OP

Project Chairs:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Editors:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Additional components:

This specification is one component of a Work Product that also includes:

- *OSLC Core Version 3.0. Part 1: Overview.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/oslc-core.html
- *OSLC Core Version 3.0. Part 2: Discovery (this document).* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/discovery.html
- *OSLC Core Version 3.0. Part 3: Resource Preview.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-preview.html
- *OSLC Core Version 3.0. Part 4: Delegated Dialogs.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/dialogs.html
- *OSLC Core Version 3.0. Part 5: Attachments.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/attachments.html
- *OSLC Core Version 3.0. Part 6: Resource Shape.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-shape.html

- *OSLC Core Version 3.0. Part 7: Vocabulary*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.html
- *OSLC Core Version 3.0. Part 8: Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.html
- *OSLC Core Version 3.0. Machine Readable Vocabulary Terms*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.ttl
- *OSLC Core Version 3.0. Machine Readable Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.ttl

Related work:

This specification is related to:

- *OSLC Core Version 3.0: Link Guidance*. https://oslc-op.github.io/oslc-specs/notes/link-guidance.html

RDF Namespaces:

http://open-services.net/ns/core#

Abstract:

This document outlines a common approach for HTTP/LDP-based servers to be able to publish their RESTful API capabilities and how clients discover and use them.

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the “Latest stage” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Open Project are listed at https://open-services.net/about/.

Comments on this work can be provided by opening issues in the project repository or by sending email to the project’s public comment list oslc-op@lists.oasis-open-projects.org.

The English version of this specification is the only normative version. Non-normative translations may also be available. Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:
[OSLC-Discovery-3.0]
OSLC Core Version 3.0. Part 2: Discovery. Edited by Jim Amsden and Andrii Berezovskyi. 26 August 2021. OASIS Standard. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/discovery.html. Latest stage: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/discovery.html.

Notices

Table of Contents

1. Introduction

This section is non-normative.

A common problem with building interoperable solutions is having a mechanism for a client to explore a server API or end-point to learn if the target application supports a set of capabilities. Client applications would then provide features based on what is discovered. For example, a person using a quality management tool wants to be able to record a defect in a change management tool. The integration from the quality management tool will want to be able to do a number of things on behalf of the user and provide an integrated experience to streamline the users workflow. To do this, the quality management tool will need to discover information about the change management tool including:

- How to authenticate
- If the target tool is capable of receiving creation requests for defects
- Whether defect creation can be handled via a user interface
- What fields are required, with what types of data
- If the defect record can be pre-filled with some data from the test currently being executed

OSLC Discovery 3.0 defines a capability providing client applications a standard way to introspect servers to determine what resource types the server supports, how to preview, select or create instances of those resources, and any constraints on resource creation or update. Discovery capabilities allow clients to determine what capabilities are provided by a server so they can adapt to, and integrate with different servers in support of end user integration scenarios.

1.1 Terminology

Terminology uses and extends the terminology and capabilities of OSLC Core Overview, W3C Linked Data Platform [LDP], W3C's Architecture of the World Wide Web [WEBARCH], Hyper-text Transfer Protocol [HTTP11].

Discovery

The act of an OSLC Client to be able to determine if an OSLC Server supports a given capability.

1.2 References

1.2.1 Normative references

[HTTP11]

R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. IETF, June 2014. Proposed Standard. URL: https://httpwg.org/specs/rfc7230.html

[LDP]

Steve Speicher; John Arwe; Ashok Malhotra. *Linked Data Platform 1.0*. W3C, 26 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/ldp/

[OSLCCore2]

S. Speicher; D. Johnson. *OSLC Core 2.0*. http://open-services.net. Finalized. URL: http://open-services.net/bin/view/Main/OslcCoreSpecification

[RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF, March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

[RFC8174]

B. Leiba. *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*. IETF, May 2017. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc8174

[ROOT-SERVICES]

J. des Rivieres; S. Speicher. *Root Services Specification*. IBM. Approved. URL: https://jazz.net/wiki/bin/view/Main/RootServicesSpec

[WELL-KNOWN]

M. Nottingham. *RFC 8615: Well-Known Uniform Resource Identifiers (URIs)*. IETF. Proposed Standard. URL: https://tools.ietf.org/html/rfc8615

1.2.2 Informative references

[WEBARCH]

Ian Jacobs; Norman Walsh. *Architecture of the World Wide Web, Volume One*. W3C, 15 December 2004. W3C Recommendation. URL: https://www.w3.org/TR/webarch/

1.3 Typographical Conventions and Use of RFC Terms

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this specification are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Motivation

This section is non-normative.

Management and use of shared information in complex domains such as IT application lifecycle management and systems and software engineering often involve the integration of many data sources supported by tools developed by different vendors on different technical architectures, and introduced at different times. Integrating these tools in order to support a wide range of evolving end user scenarios requires flexible and loosely coupled interactions between consumers and providers of this shared information. An important way of achieving this flexibility and loose coupling is to allow clients to incrementally discover the capabilities of any given server, and then adapt to what is discovered in order to maximize end user capabilities. Although a client may not know ahead of time what capabilities a given server might provide, they can know a standard means of discovering those capabilities, and can be developed to dynamically adapt to the discovered capabilities.

Key usage scenarios that motivate a number of requirements for discovery include determining if the target tool supports:

- A certain authentication model (OAuth2, OpenID Connect, HTTPS Basic)
- Creating resources with a given type
- User interface previews of a given resource
- User interface dialogs for creating or selecting resources of given types
- Prefilling dialogs for resource creation
- Describing the constraints on a resource creation or update request (allowable properties or values)
- Querying resources to select specific instances and property values
- Adding attachments to resources

Additionally there is motivation to establish a common way for tools to support similar mechanisms so that each new capability doesn't introduce a new discovery model. It is also desired to have capabilities defined, whether within a standards development group or proprietary, to be able to leverage a similar approach to discovery.

Security concerns are also important when managing shared information across organization and tool boundaries. The specific security needs of any application however are difficult to predict. Experience has shown that this variability results in complexity for tool integration and therefore some standard mechanism for authentication discovery is highly desired.

The basis for how clients discover capabilities should be based on the methods established by [LDP] and [HTTP11], and service provider resources defined by [OSLCCore2].

3. Basic Concepts

This section is non-normative.

The following sections introduce the basic concepts of OSLC discovery including how clients find discovery URIs, different approaches clients might use to discover server capabilities, and how server capabilities might be extended.

3.1 Bootstrapping Discovery

This section is non-normative.

Discovery will always have to start with at least one discovery resource URI to bootstrap discovery on that server. Servers must provide some way for clients to learn about, find, or discover such LDPC URIs. For example, servers could provide such information:

- In their user documentation or UI
- Using HTTP `OPTIONS *` to return a ServiceProviderCatalog or link headers to root LDPCs describing the discovery capabilities offered

The URI could be for a ServiceProviderCatalog or context LDPC resource on which either static up-front or dynamic incremental discovery can be performed. Different server implementation architectures and extensibility mechanisms may require different approaches for discovering OSLC discovery resource URIs.

This specification specifies how servers respond to discovery requests through LDPC Link headers or ServiceProviderCatalog and/or ServiceProvider resources. It does not specify how servers organize their LDPCs, how they make distinguished LDPCs known to end users to start the discovery process, or how servers provide efficient access to discovery information that may be distributed over many LDPCs managed by the server. Servers may choose to support the OSLC Query capability on OSLC LDPCs and discovery resources in order to facilitate access to discovery information.

3.2 Approaches to Discovery

This section is non-normative.

There are various approaches for how servers define and advertise their capabilities, and how clients can efficiently discover what is available. The following sections will provide guidance on approaches that should be used.

OSLC defines two broad approaches for clients to discover capabilities provided by a server, loosely categorized as "Static Up-Front" and "Dynamic Incremental".

Static Up-Front discovery, which is compatible with [OSLCCore2], is an up-front or somewhat more static approach to discovery that utilizes ServiceProviderCatalog, ServiceProvider, and Service resources. Typically a client would perform discovery on startup by accessing the Services defined by any ServiceProvider resources the client might need. A client could also access one or more ServiceProviderCatalog resources in order to locate the available ServiceProvider resources. The client would then configure its capabilities based on what was discovered. In many, or possibly most instances, the ServiceProvider resources will be dynamically created by servers based on the state of the information they manage. Clients may choose to periodically refresh their capabilities by re-reading the ServiceProviders and adapting to the newly available services. Therefore this approach to discovery is not completely static, or up-front, but that does represent a possible common usage pattern.

Figure Fig. 1 Service Provider concepts and relationships illustrates the Service Provider Catalog and Service Provider concepts and relationships. There are two resources defined: Service Provider Catalog and Service Provider, that provide the discovery information. There are also a set of local in-line resources that are provided inside these resources to define namespaces, OAuth configurations, contributors as well as services and their capabilities.

To allow clients to discover the RDF vocabularies supported by a server, those vocabularies should be referenced from the service discovery documents, and the vocabularies themselves and their constraining ResourceShapes should be readable RDF resources. The `oslc:domain` property references a namespace that should resolve to the vocabulary document.

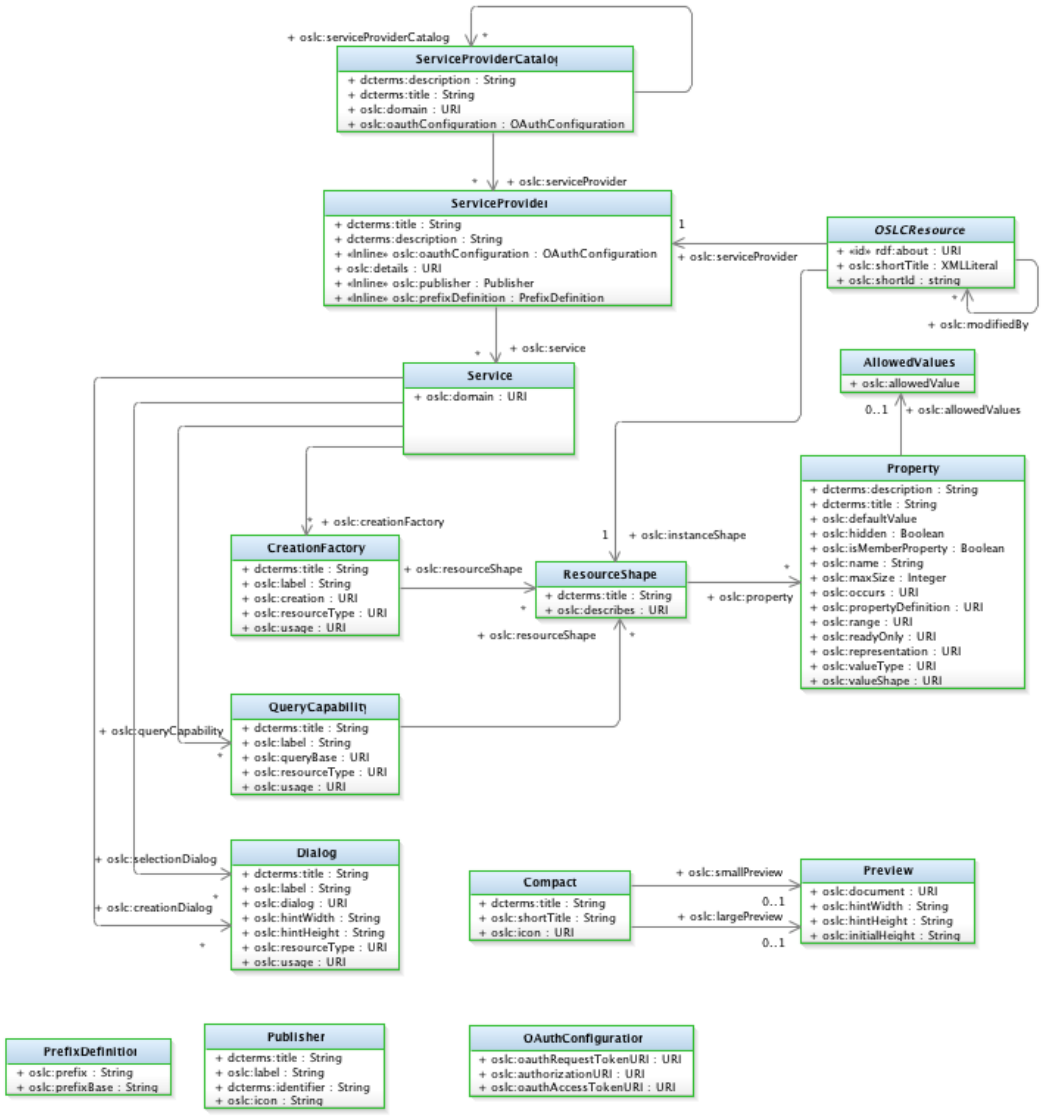

Fig. 1 Service Provider concepts and relationships

Dynamic Incremental discovery is a second approach that utilizes lazy or deferred discovery, getting just the information that is needed for any client capability when it is needed, and not getting information about server capabilities that might not ever be used. Clients typically utilize HTTP `OPTIONS` or `HEAD` methods on LDPCs and get discovery information from Link headers included in the HTTP response. This approach is more applicable for situations where the services provided by a server are changing rapidly as the result of resource creation or update, and clients will need to do incremental discovery before executing the next operations. This approach is also useful for clients that only need to do very specific things and are not necessarily involved in a long-running conversation with a server. Discovery in this case can be a simple HTTP `OPTIONS` request on the required LDPC and the client can immediately complete its operation without needing to deal with potentially large discovery documents.

Both of these approaches are based on a uniform discovery capability. ServiceProviderCatalog, ServiceProvider and Service resources from [OSLCCore2] are specific kinds of LDP Containers. The members of a ServiceProviderCatalog resource include ServiceProviderCatalogs and ServiceProvider LDPCs. ServiceProvider resource members include Service LDPCs. Each of these LDPCs have additional properties as defined by the OSLC vocabulary and shapes. This allows clients to use either approach to discover server provided OSLC capabilities, and maintains compatibility with [OSLCCore2] while providing new, simple and flexible approaches to service discovery.

3.3 Updating Discovery Information

This section is non-normative.

Servers may chose to support changes to their configurations in order to support adding new domains and services, extending existing domains and services, and/or integrating domains. This specification does not specify how servers provide extensibility mechanisms. Some possible approaches could include:

- Configuration information could be provided through files that are read at startup to define the service providers and services for the supported domains. These files could utilize the ServiceProviderCatalog and ServiceProvider resource formats.
- Servers may provide administration facilities or operational modes (through commands, REST services, GUI, etc.) that:
  - Create and configure additional LDPCs that provide new services
  - Define or extend domain vocabularies
  - Define or extend resource shapes to constrain vocabularies for specific purposes
- Servers may choose to support these configuration changes only on restart, or dynamically at runtime.

The ServiceProviderCatalog, ServiceProvider and Service shapes specify that much of the discovery information provided in these resource representations is read-only. Therefore clients accessing these resources cannot expect to change read-only properties via HTTP `PUT` operation on the discovery resources as a means of updating server configurations. However, these constraints only apply to these particular discovery resource representations and do not prevent servers from providing other means of modifying their configuration information. These modifications would then be reflected in read-only properties in the discovery resource representations when they are accessed.

4. Discovery Capabilities

The following sections define the OSLC Core discovery capabilities.

4.1 General Discovery Methods

The following clauses apply to all discovery capabilities including resource creation, resource preview, delegated dialogs for resource creation and selection, and resource constraints discovery. OSLC discovery capabilities may also apply to OSLC resources themselves, including LDPCs, and the discovery LDPC resources including ServiceProviderCatalog, ServiceProvider and Service resources. This allows servers to dynamically configure their capabilities, or provide users with a means of selecting the capabilities they need from those provided by a server.

Clients **SHOULD** use HTTP `OPTIONS` to fetch various headers and other configuration information that may be exposed in the response content body from other HTTP methods. [dis-1]

Servers **SHOULD** minimize the use of HTTP response headers on various HTTP operations as to avoid unnecessary additional response content for clients to consume. [dis-2]

This is also to avoid the complexity on server implementations that would be needed to provide such additional content.

4.2 Well-known URI Bootstrapping

An OSLC Server **MAY** serve a Root Services document at the `/.well-known/oslc/rootservices.xml` URI [WELL-KNOWN] if the Server provides access to such a document. The Root Services document **SHALL** conform to the [ROOT-SERVICES] specification. If no specific content type is requested, or if the requested content type is `application/xml` or `application/rdf+xml`, the server **SHALL** return the Root Services document using the RDF/XML-ABBREV format. Servers **MAY** return other formats in response to other requested content types. [dis-3]

An OSLC Server **SHOULD** serve a Service Provider Catalog at the `/.well-known/oslc/sp-catalog` URI if the Server provides access to such a document. [dis-4]

An OSLC Server **MAY** serve an LDP Container at the `/.well-known/oslc/ldpc` URI if the Server provides access to such a document. [dis-5]

An OSLC Server **MAY** use an HTTP redirect to a URI that satisfies the conformance clauses listed above in response to any request under `/.well-known/oslc/` URIs. [dis-6]

An OSLC Server **SHALL NOT** serve documents on any vendor-specific URIs under `/.well-known/oslc/` not defined in any official OSLC specification. [dis-7]

4.3 Resource Creation Discovery

Resource creation is done by sending an HTTP `POST` to a URI that supports resource creation, providing the resource content in the entity request body. Clients can discover resources that support resource creation either through the `http://open-services.net/ns/core#creationFactory` property of a Service in a ServiceProvider resource, or by using an `OPTIONS` request on an LDPC to determine if it accepts the `POST` method.

Servers **MAY** provide one or more creation factories to enable creation of new resources. Creation factories are LDPCs whose URI **MAY** be given in the `oslc:creationFactory` property of a Service resource. [dis-8]

The existence of an `Accept-Post` header on an HTTP response to a given Request-URI indicates i) that an HTTP POST will be accepted for authorized requests and ii) what types of content are supported in the entity body of the HTTP POST request. Restating of [LDP] conformance clause about `Accept-Post`. [dis-9]

The existence of a `Link: <http://www.w3.org/ns/ldp#Container>; rel="type"` header on an HTTP response to a given Request-URI will indicate that the resource is a LDP Container. Restating conformance clause for [LDP] `Link` header and resource types. [dis-10]

In a response to Request-URI on an LDPC, servers **SHOULD** include `Link` headers with the relation-types set to `rel="http://open-services.net/core#resourceType"` and the Target URIs set to the `rdf:type` of resources that can be created in the LDPC. [dis-11]

Note: An LDPC can contain multiple types of resources, and the supported member types may change over time. Since there is always

some time between when the test is done and when the creation request is sent, and that there may be additional server enforced constraints on the creation resource representation, there is no guarantee that a future creation request will succeed.

Servers **MAY** include a RDF triple in resource response body of the form: `<container-URI> oslc:resourceType <type-URI>`. Clients **SHOULD** use the predicate `oslc:resourceType` when converting HTTP Link headers that have `http://open-services.net/ns/core#resourceType` as the Link-relation ("rel" value) into RDF triples. [dis-12]

This is to assist with scenarios where client applications may want to use the RDF representation in a query to locate LDP Containers that can be used to create the same resource types.

The following example is an OPTIONS request on the /bugs/ resource that demonstrates some of the discovery capabilities.

EXAMPLE 1

```
OPTIONS /bugs/ HTTP/1.1
Host: example.com
```

The response to this example request indicates that `POST` is supported for creating resources while the `Accept-Post` header indicates Turtle and JSON-LD content types are supported. The `"type"` link header indicates the resource is an LDP BasicContainer. The `"resourceType"` link headers indicate which resource types are supported on `POST`. In this case the LDP Container advertises support for creating two types of resources: Bug and Feature. POSTing an entity request body that is not one of these types would result in an error.

EXAMPLE 2

```
HTTP/1.1 204 No Content
Date: Thu, 12 Jun 2014 18:26:59 GMT
Allow: POST,GET,OPTIONS,HEAD,PUT
Accept-Post: text/turtle, application/ld+json
Link: <http://www.w3.org/ns/ldp#BasicContainer>; rel="type",
    <http://www.w3.org/ns/ldp#Resource>; rel="type"
Link: <http://open-services.net/ns/cm#Bug>; rel="http://open-services.net/ns/core#resourceType",
    <http://example.com/vocab#Feature>; rel="http://open-services.net/ns/core#resourceType"
```

4.4 Resource Creation and Update Constraints Discovery

In addition to the ways one can discover if a given OSLC Server supports creation of resources and for which types, it is helpful to understand if there are server-enforced constraints on the resource representation. Clients can discover these constraints either through the `http://open-services.net/ns/core#resourceShape` property of a Creation Factory resource, or by using `Link: <constraint-URI>; rel="http://www.w3.org/ns/ldp#constrainedBy"` header on an HTTP response to a given Request-URI.

Servers **MAY** describe constraints enforced on resource representations through the `http://open-services.net/ns/core#resourceShape` property of a Creation Factory resource. [dis-13]

In a response to an HTTP `OPTIONS`, `HEAD` or `GET` method on a given Request-URI referencing an LDPC, servers **SHOULD** include a `Link` header with the relation-type set to `rel="http://open-services.net/core#constrainedBy"` and the Target URI set to the URI of a resource that defines constraints on the to-be created or updated resource representation in the LDPC. The resource referenced by Target URI is **RECOMMENDED** to be a machine-readable representation such as OSLC ResourceShapes, but **MAY** be some variant or other constraint document. See [LDP] section about server published constraints. [dis-14]

EXAMPLE 3

```
Link: <http://example.com/shapes/bug>; rel="http://www.w3.org/ns/ldp#constrainedBy"
```

The link header in the example above would be returned on an OPTIONS or HEAD request to a resource of type `<http://open-services.net/ns/cm#Bug>` to provide the URI of the creation or update constraints.

In a response to an HTTP `POST` or `PUT` method on a given Request-URI referencing an LDPC, servers **SHOULD** include a `Link` header with the relation-type set to `rel="http://open-services.net/core#constrainedBy"` and the Target URI set to the URI of a resource that defines constraints that on the to-be created or updated resource representation in the LDPC that were not satisfied. The resource referenced by Target URI is **RECOMMENDED** to be a machine-readable representation such as OSLC ResourceShapes, but **MAY** be some variant or other constraint document. See [LDP] section about server published constraints. [dis-15]

> EXAMPLE 4
>
> ```
> Link: <http://example.com/shapes/bug>; rel="http://www.w3.org/ns/ldp#constrainedBy"
> ```

The link header in the example above would be returned on a POST or PUT to a resource of type `<http://open-services.net/ns/cm#Bug>` that violated the referenced constraint.

Servers **MAY** include a RDF triple in resource response body of the form: `<container-URI> ldp:constrainedBy <shape-URI>`. Clients **SHOULD** use the predicate `ldp:constrainedBy` when converting HTTP response headers for the same Link-relation type, into an RDF triple. [dis-16]

This is to assist with scenarios where client applications may want to use the RDF representation in a query to locate LDP Containers that are constraints by the same resource.

4.5 Resource User Interface Preview Discovery

See OSLC Resource Preview for resource preview discovery using the `Link` header or the `Prefer` header.

4.6 Resource User Interface Delegated Dialogs Discovery

See Delegated dialogs for resource selection and creation delegated UI discovery using the `http://open-services.net/ns/core#selectionDialog` or `http://open-services.net/ns/core#creationDialog` properties of a Service resource, or the `Link` header or the `Prefer` header.

4.7 Authentication Discovery

Clients **SHOULD** determine what authentication schemes a server supports by parsing and processing the challenge sent by the target server in response to a request for a protected resource. [dis-17]

Servers **MAY** provide OAuth configuration information in the OAuthConfiguration member of a ServiceProviderCatalog as described in 5. Resource Constraints. [dis-18]

5. Resource Constraints

This document applies the following constraints to the OSLC Core vocabulary terms.

5.1 Resource: ServiceProviderCatalog

- **Describes:** `http://open-services.net/ns/core#ServiceProviderCatalog`
- **Summary:** Service Provider Catalog
- **Description:** An LDPC describing an OSLC server that offers one or more ServiceProvider LDPCs. Servers MAY also organize the ServiceProviders in one or more ServiceProviderCatalog LDPCs to enable OSLC clients to find ServiceProviders offered [dis-19]. The members of these catalogs may include other nested catalogs as well as service providers.

ServiceProviderCatalog Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:description`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	Description of the services provided.
`dcterms:publisher`	Zero-or-one	true	AnyResource	Inline	`oslc:Publisher`	Describes the software product that provides the implementation.
`dcterms:title`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	Title of this resource.
`oslc:domain`	Zero-or-many	true	Resource	Reference	Unspecified	Namespace URI of the specification that is implemented by this service. In most cases this namespace URI will be for an OSLC domain, but other URIs MAY be used [dis-20].
`oslc:oauthConfiguration`	Zero-or-many	true	AnyResource	Inline	`oslc:OAuthConfiguration`	Defines the three OAuth URIs required for a client to act as an OAuth consumer.
`oslc:serviceProvider`	Zero-or-many	true	AnyResource	Either	`oslc:ServiceProvider`	A service provider LDPC offered by this server.

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:serviceProviderCatalog	Zero-or-many	true	AnyResource	Either	oslc:ServiceProviderCatalog	Additional service provider catalog LDPCs used to organize services.

5.2 Resource: ServiceProvider

- **Describes:** http://open-services.net/ns/core#ServiceProvider
- **Summary:** Service Provider
- **Description:** An LDPC whose members are the Service LDPCs offered by an OSLC server.

ServiceProvider Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
dcterms:description	Zero-or-one	true	XMLLiteral	N/A	Unspecified	Description of the services provided.
dcterms:publisher	Zero-or-one	true	AnyResource	Inline	oslc:Publisher	Describes the software product that provides the implementation.
dcterms:title	Zero-or-one	true	XMLLiteral	N/A	Unspecified	Title of this resource.
oslc:details	Zero-or-many	true	Resource	Reference	Unspecified	A URL that may be used to retrieve a resource to determine additional details about the service provider such as a web page describing it.
oslc:oauthConfiguration	Zero-or-many	true	AnyResource	Inline	oslc:OAuthConfiguration	Defines the three OAuth URIs required for a client to act as an OAuth consumer.
oslc:prefixDefinition	Zero-or-many	true	AnyResource	Inline	oslc:PrefixDefinition	Defines a namespace prefix for use in JSON representations and in forming OSLC Query Syntax strings.
oslc:service	One-or-many	true	AnyResource	Inline	oslc:Service	Describes a service LDPC offered by the service provider.

5.3 Resource: Service

- **Describes:** http://open-services.net/ns/core#Service
- **Summary:** Service
- **Description:** An LDPC whose properties describe specific services offered by a server, and the URIs to use for those services in the context of that ServiceProvider.

Service Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:creationDialog	Zero-or-many	true	AnyResource	Inline	oslc:Dialog	Enables clients to create a resource via UI.
oslc:creationFactory	Zero-or-many	true	AnyResource	Inline	oslc:CreationFactory	An LDPC that enables clients to create new resources.
oslc:domain	Exactly-one	true	Resource	Reference	Unspecified	Namespace URI of the specification that is implemented by this service. In most cases this namespace URI will be for an OSLC domain, but other URIs MAY be used [dis-21].
oslc:queryCapability	Zero-or-many	true	AnyResource	Inline	oslc:QueryCapability	Enables clients query across a collection of resources.
oslc:selectionDialog	Zero-or-many	true	AnyResource	Inline	oslc:Dialog	Enables clients to select a resource via UI.
oslc:usage	Zero-or-many	true	Resource	Reference	Unspecified	An identifier URI for the domain specified usage of this resource. If a resource has multiple uses, it may designate the primary or default one that should be used with a property value of oslc:default.

5.4 Resource: CreationFactory

- **Describes:** http://open-services.net/ns/core#CreationFactory
- **Summary:** Creation Factory
- **Description:** A Creation Factory describes a capability for creating resources, including an LDPC capable of creating and containing new resources via HTTP POST.

CreationFactory Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
dcterms:title	Exactly-one	true	XMLLiteral	N/A	Unspecified	Title string that could be used for display.
oslc:creation	Exactly-one	true	Resource	Reference	ldp:Container	To create a new resource via the factory, post it to this URI.
oslc:label	Zero-or-one	true	string	N/A	Unspecified	Very short label for use in menu items.
oslc:resourceShape	Zero-or-many	true	Resource	Reference	oslc:ResourceShape	A Creation Factory MAY provide Resource Shapes that describe shapes of resources that may be created [dis-22].
oslc:resourceType	Zero-or-many	true	Resource	Reference	rdfs:Class	The expected resource type URI of the resource that will be created using this creation factory. These would be the URIs found in the result resource's rdf:type property.

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:usage	Zero-or-many	true	Resource	Reference	Unspecified	An identifier URI for the domain specified usage of this resource. If a resource has multiple uses, it may designate the primary or default one that should be used with a property value of oslc:default.

5.5 Resource: QueryCapability

- **Describes:** http://open-services.net/ns/core#QueryCapability
- **Summary:** Query Capability
- **Description:** A Query Capability describes a query capability, capable of querying resources via HTTP GET or POST.

QueryCapability Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
dcterms:title	Exactly-one	true	XMLLiteral	N/A	Unspecified	Title string that could be used for display.
oslc:label	Zero-or-one	true	string	N/A	Unspecified	Very short label for use in menu items.
oslc:queryBase	Exactly-one	true	Resource	Reference	Unspecified	The base URI to use for queries. Queries are invoked via HTTP GET on a query URI formed by appending a key=value pair to the base URI, as described in Query Capabilities section.
oslc:resourceShape	Zero-or-one	true	Resource	Reference	oslc:ResourceShape	The Query Capability SHOULD provide a Resource Shape that describes the query base URI [dis-23].
oslc:resourceType	Zero-or-many	true	Resource	Reference	rdfs:Class	The expected resource type URI that will be returned with this query capability. These would be the URIs found in the result resource's rdf:type property.
oslc:usage	Zero-or-many	true	Resource	Reference	Unspecified	An identifier URI for the domain specified usage of this query capability. If a service provides multiple query capabilities, it may designate the primary or default one that should be used with a property value of oslc:default.

5.6 Resource: Publisher

- **Describes:** http://open-services.net/ns/core#Publisher
- **Summary:** Publisher
- **Description:** A Publisher identifies and describes the software product that provides the OSLC implementation.

Publisher Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
dcterms:identifier	Exactly-one	unspecified	string	N/A	Unspecified	A URN that uniquely identifies the implementation.

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
dcterms:title	Exactly-one	true	XMLLiteral	N/A	Unspecified	Title string that could be used for display.
oslc:icon	Zero-or-one	true	Resource	Reference	Unspecified	URL to an icon file that represents the provider. This icon should be a favicon format and 16x16 pixels in size.
oslc:label	Zero-or-one	true	string	N/A	Unspecified	Very short label for use in menu items.

5.7 Resource: PrefixDefinition

- **Describes:** http://open-services.net/ns/core#PrefixDefinition
- **Summary:** Prefix Definition
- **Description:** Service Providers **MUST** provide a Prefix Definition for each prefix supported by the service [dis-24]. Each Prefix Definition defines a namespace prefix that clients **MAY** use in forming OSLC Query Syntax strings [dis-25].

PrefixDefinition Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:prefix	Exactly-one	true	string	N/A	Unspecified	Namespace prefix to be used for this namespace.
oslc:prefixBase	Exactly-one	true	Resource	Reference	Unspecified	The base URI of the namespace.

5.8 Resource: OAuthConfiguration

- **Describes:** http://open-services.net/ns/core#OAuthConfiguration
- **Summary:** OAuth Configuration
- **Description:** Service Providers that support OAuth Authentication **SHOULD** provide a way for clients to automatically discover the three OAuth URIs necessary to act as an OAuth Consumer [dis-26].

OAuthConfiguration Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:authorizationURI	Exactly-one	true	Resource	Reference	Unspecified	URI for obtaining OAuth authorization.
oslc:oauthAccessTokenURI	Exactly-one	true	Resource	Reference	Unspecified	URI for obtaining OAuth access token.
oslc:oauthRequestTokenURI	Exactly-one	true	Resource	Reference	Unspecified	URI for obtaining OAuth request token.

6. Conformance

Implementations of this specification need to satisfy the following conformance clauses.

Clause Number	Requirement
dis-1	Clients **SHOULD** use HTTP `OPTIONS` to fetch various headers and other configuration information that may be exposed in the response content body from other HTTP methods.
dis-2	Servers **SHOULD** minimize the use of HTTP response headers on various HTTP operations as to avoid unnecessary additional response content for clients to consume.
dis-3	An OSLC Server **MAY** serve a Root Services document at the `/.well-known/oslc/rootservices.xml` URI [WELL-KNOWN] if the Server provides access to such a document. The Root Services document **SHALL** conform to the [ROOT-SERVICES] specification. If no specific content type is requested, or if the requested content type is `application/xml` or `application/rdf+xml`, the server **SHALL** return the Root Services document using the RDF/XML-ABBREV format. Servers **MAY** return other formats in response to other requested content types.
dis-4	An OSLC Server **SHOULD** serve a Service Provider Catalog at the `/.well-known/oslc/sp-catalog` URI if the Server provides access to such a document.
dis-5	An OSLC Server **MAY** serve an LDP Container at the `/.well-known/oslc/ldpc` URI if the Server provides access to such a document.
dis-6	An OSLC Server **MAY** use an HTTP redirect to a URI that satisfies the conformance clauses listed above in response to any request under `/.well-known/oslc/` URIs.
dis-7	An OSLC Server **SHALL NOT** serve documents on any vendor-specific URIs under `/.well-known/oslc/` not defined in any official OSLC specification.
dis-8	Servers **MAY** provide one or more creation factories to enable creation of new resources. Creation factories are LDPCs whose URI **MAY** be given in the `oslc:creationFactory` property of a Service resource.
dis-9	The existence of an `Accept-Post` header on an HTTP response to a given Request-URI indicates i) that an HTTP POST will be accepted for authorized requests and ii) what types of content are supported in the entity body of the HTTP POST request. Restating of [LDP] conformance clause about `Accept-Post`.
dis-10	The existence of a `Link: <http://www.w3.org/ns/ldp#Container>; rel="type"` header on an HTTP response to a given Request-URI will indicate that the resource is a LDP Container. Restating conformance clause for [LDP] `Link` header and resource types.
dis-11	In a response to Request-URI on an LDPC, servers **SHOULD** include `Link` headers with the relation-types set to `rel="http://open-services.net/core#resourceType"` and the Target URIs set to the `rdf:type` of resources that can be created in the LDPC.
dis-12	Servers **MAY** include a RDF triple in resource response body of the form: `<container-URI> oslc:resourceType <type-URI>`. Clients **SHOULD** use the predicate `oslc:resourceType` when converting HTTP Link headers that have `http://open-services.net/ns/core#resourceType` as the Link-relation ("rel" value) into RDF triples.
dis-13	Servers **MAY** describe constraints enforced on resource representations through the `http://open-services.net/ns/core#resourceShape` property of a Creation Factory resource.
dis-14	In a response to an HTTP `OPTIONS`, `HEAD` or `GET` method on a given Request-URI referencing an LDPC, servers **SHOULD** include a `Link` header with the relation-type set to `rel="http://open-services.net/core#constrainedBy"` and the Target URI set to the URI of a resource that defines constraints on the to-be created or updated resource representation in the LDPC. The resource referenced by Target URI is **RECOMMENDED** to be a machine-readable representation such as OSLC ResourceShapes, but **MAY** be some variant or other constraint document. See [LDP] section about server published constraints.
dis-15	In a response to an HTTP `POST` or `PUT` method on a given Request-URI referencing an LDPC, servers **SHOULD** include a `Link` header with the relation-type set to `rel="http://open-services.net/core#constrainedBy"` and the Target URI set to the URI of a resource that defines constraints that on the to-be created or updated resource representation in the LDPC that were not satisfied. The resource referenced by Target URI is **RECOMMENDED** to be a machine-readable representation such as OSLC ResourceShapes, but **MAY** be some variant or other constraint document. See [LDP] section about server published constraints.
dis-16	Servers **MAY** include a RDF triple in resource response body of the form: `<container-URI> ldp:constrainedBy <shape-URI>`. Clients **SHOULD** use the predicate `ldp:constrainedBy` when converting HTTP response headers for the same Link-relation type, into an RDF triple.

Clause Number	Requirement
dis-17	Clients **SHOULD** determine what authentication schemes a server supports by parsing and processing the challenge sent by the target server in response to a request for a protected resource.
dis-18	Servers **MAY** provide OAuth configuration information in the OAuthConfiguration member of a ServiceProviderCatalog as described in 5. Resource Constraints.
dis-19	Servers **MAY** also organize the ServiceProviders in one or more ServiceProviderCatalog LDPCs to enable OSLC clients to find ServiceProviders offered
dis-20	In most cases this namespace URI will be for an OSLC domain, but other URIs **MAY** be used
dis-21	In most cases this namespace URI will be for an OSLC domain, but other URIs **MAY** be used
dis-22	A Creation Factory **MAY** provide Resource Shapes that describe shapes of resources that may be created
dis-23	The Query Capability **SHOULD** provide a Resource Shape that describes the query base URI
dis-24	Service Providers **MUST** provide a Prefix Definition for each prefix supported by the service
dis-25	Each Prefix Definition defines a namespace prefix that clients **MAY** use in forming OSLC Query Syntax strings
dis-26	Service Providers that support OAuth Authentication **SHOULD** provide a way for clients to automatically discover the three OAuth URIs necessary to act as an OAuth Consumer

OASIS OPEN PROJECTS

OSLC Core Version 3.0. Part 3: Resource Preview

OASIS Standard
26 August 2021

This stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-preview.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-preview.pdf

Previous stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/resource-preview.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/resource-preview.pdf
(published as Project Specification)

Latest stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/resource-preview.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/resource-preview.pdf

Latest version:

https://open-services.net/spec/core/latest

Latest editor's draft:

https://open-services.net/spec/core/latest-draft

Open Project:

OASIS Open Services for Lifecycle Collaboration (OSLC) OP

Project Chairs:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Editors:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Additional components:

This specification is one component of a Work Product that also includes:

- *OSLC Core Version 3.0. Part 1: Overview.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/oslc-core.html
- *OSLC Core Version 3.0. Part 2: Discovery*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/discovery.html
- *OSLC Core Version 3.0. Part 3: Resource Preview(this document)*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-preview.html
- *OSLC Core Version 3.0. Part 4: Delegated Dialogs*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/dialogs.html
- *OSLC Core Version 3.0. Part 5: Attachments*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/attachments.html
- *OSLC Core Version 3.0. Part 6: Resource Shape*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-shape.html
- *OSLC Core Version 3.0. Part 7: Vocabulary*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.html
- *OSLC Core Version 3.0. Part 8: Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.html
- *OSLC Core Version 3.0. Machine Readable Vocabulary Terms*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.ttl
- *OSLC Core Version 3.0. Machine Readable Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.ttl

Related work:

This specification is related to:

- *OSLC Core Version 3.0: Link Guidance.* https://oslc-op.github.io/oslc-specs/notes/link-guidance.html

RDF Namespaces:

http://open-services.net/ns/core#

Abstract:

Resource Preview defines a technique to get a minimal HTML representation of a resource identified by a URL. Applications often use this capability to render or display links with more appropriate icons and labels, or display a preview when a user mouses over a link.

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the "Latest stage" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Open Project are listed at https://open-services.net/about/.

Comments on this work can be provided by opening issues in the project repository or by sending email to the project's public comment list oslc-op@lists.oasis-open-projects.org.

The English version of this specification is the only normative version. Non-normative translations may also be available. Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:
[OSLC-ResourcePreview-3.0]
OSLC Core Version 3.0. Part 3: Resource Preview. Edited by Jim Amsden and Andrii Berezovskyi. 26 August 2021. OASIS Standard. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-preview.html. Latest stage: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/resource-preview.html.

Notices

Table of Contents

1. Introduction

This section is non-normative.

This specification describes how a client application can display links and embed rich previews for resources from other applications. Links may have a label and an icon, and previews are HTML markup provided by a server and displayed directly inside the client application.

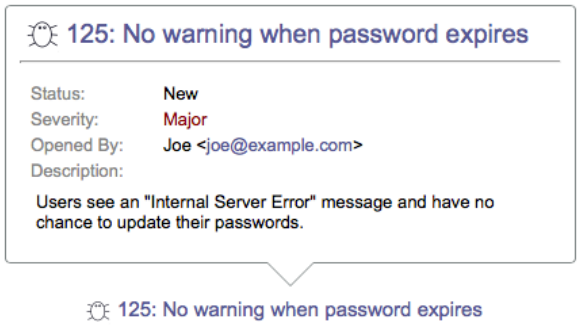

Fig. 1 Preview for a Link

Previews often appear as a pop-up when a user mouses over a link as in Fig. 1 Preview for a Link. However, a client may wish to display a preview differently depending on the kind of application, the size of the screen, and the capabilities of the device. For example, a desktop application on a PC might handle previews differently than a mobile application running on a small touchscreen.

Servers can provide both small and large previews. A client might show a small preview first, then if the user gestures, show additional details from the large preview. Servers suggest sizes for previews, and previews can ask to be resized after they are displayed.

A client only needs to know the URI of a resource to display a link and a preview. It doesn't need to know anything else about the resource. Clients don't need to copy, synchronize, or cache any data.

1.1 Terminology

Terminology uses and extends the terminology and capabilities of OSLC Core Overview, W3C Linked Data Platform [LDP], W3C's Architecture of the World Wide Web [WEBARCH], and Hyper-text Transfer Protocol [HTTP11].

The following terms are used in discussions of previews:

Compact resource
: A resource describing how to display a link and preview for another, associated resource.

Preview
: An HTML representation of a resource that can be embedded in another user interface.

1.2 References

1.2.1 Normative references

[CSS21]

Bert Bos; Tantek Çelik; Ian Hickson; Håkon Wium Lie. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C, 7 June 2011. W3C Recommendation. URL: https://www.w3.org/TR/CSS21/

[HTTP11]

R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. IETF, June 2014. Proposed Standard. URL: https://httpwg.org/specs/rfc7230.html

[JSON-LD]

Manu Sporny; Gregg Kellogg; Markus Lanthaler. *JSON-LD 1.0*. W3C, 3 November 2020. W3C Recommendation. URL: https://www.w3.org/TR/json-ld/

[JSONSchema]

A. Wright; H. Andrews; B. Hutton. *JSON Schema*. https://www.ietf.org. Informational (Draft). URL: https://tools.ietf.org/html/draft-bhutton-json-schema-00

[LDP]

Steve Speicher; John Arwe; Ashok Malhotra. *Linked Data Platform 1.0*. W3C, 26 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/ldp/

[OSLCUIPreview20]

S. Bosworth. *OSLC Core 2.0 - UI Preview*. http://open-services.net. Finalized. URL: http://open-services.net/bin/view/Main/OslcCoreUiPreview

[RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF, March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

[RFC4627]

D. Crockford. *The application/json Media Type for JavaScript Object Notation (JSON)*. IETF, July 2006. Informational. URL: https://www.rfc-editor.org/rfc/rfc4627

[RFC5988]

M. Nottingham. *Web Linking*. IETF, October 2010. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc5988

[RFC7240]

J. Snell. *Prefer Header for HTTP*. IETF, June 2014. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc7240

[RFC8174]

B. Leiba. *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*. IETF, May 2017. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc8174

[turtle]

Eric Prud'hommeaux; Gavin Carothers. *RDF 1.1 Turtle*. W3C, 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/turtle/

[whatwg-web-messaging]

WHATWG contributors. *HTML. Part 9.4: Cross-document messaging*. WHATWG. Living Standard. URL: https://html.spec.whatwg.org/multipage/web-messaging.html#web-messaging

1.2.2 Informative references

[WEBARCH]

Ian Jacobs; Norman Walsh. *Architecture of the World Wide Web, Volume One*. W3C, 15 December 2004. W3C Recommendation. URL: https://www.w3.org/TR/webarch/

1.3 Typographical Conventions and Use of RFC Terms

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this specification are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The namespace for OSLC Core is `http://open-services.net/ns/core#`.

Commonly used namespace prefixes:

```
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix oslc:    <http://open-services.net/ns/core#>.
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#>.
```


2. Motivation

This section is non-normative.

Applications often display links to related resources from other applications. For example, a quality management application might show a link to a related defect when displaying a test result. In some cases, the quality management application only has a URI pointing to the defect. The application needs a way to display the link with an appropriate label and icon so testers find out the status of the defect without leaving the quality management application.

Compact resources and resource preview provide these capabilities. Clients can discover the link label and a resource icon given only the resource URI. The client can then use the link label and icon as a means of displaying the link in a meaningful way. Previews allow users to see see information about related resources quickly without leaving the application they're in, even when the resources are from another server.

3. Working with Previews

This section is non-normative.

To enable previews of a resource, servers supply an associated Compact resource describing the preview. The Compact resource can contain a link label, icon, and small and/or large previews of the resource. Compact resources always have a JSON representation [RFC4627], but they can also have other representations such as XML, Turtle [turtle] or JSON-LD [JSON-LD]. Here is a simple example of a Compact resource:

EXAMPLE 1: A Compact Resource

```
{
    "title": "Screenshot of the problem",
    "icon": "http://example.com/icons/attachment.jpg",
    "smallPreview": {
        "document": "http://example.com/bugs/324/screenshot?preview=small"
    }
}
```

To display a link with a label or a preview of a resource, clients request its Compact resource. The URI of the Compact resource is found through an HTTP `Link` header [RFC5988] in HTTP responses to the resource URI. Alternatively, a client can use a `Prefer` request header in requests for the resource to ask for its Compact resource. The server can then inline the Compact resource directly in the response, saving an HTTP request.

The Compact resource may contain small and/or large previews. Each preview can have size hints and a link to an HTML representation designed to be embedded in other user interfaces. To display the preview, the client creates an HTML `iframe` element in its user interface and sets the `iframe` element's `src` to the preview document URI. This sandboxes the preview from other content in the client application and allows the preview to use its own stylesheets and scripts.

3.1 Getting the Compact Resource

This section is non-normative.

Clients can discover Compact resources in three ways:

1. Discovering Compact Resources Using the HTTP Accept Header
2. Discovering Compact Resources Using the HTTP Link Header
3. Discovering Compact Resources Using the HTTP Prefer Header

The HTTP `Accept` header allows clients to access previews using the `application/x-oslc-compact+xml` MIME type extension to request the Compact resource representation for a URI instead of the content of the resource itself. This usage is considered deprecated and is included only to maintain compatibility with [OSLCUIPreview20].

The HTTP `Link` header allows servers to offer previews for any resource, even binary resources with no RDF or JSON representations. To discover previews using the `Link` header, a client typically performs an HTTP HEAD or OPTIONS request on the resource URI. The response contains a `Link` header with the URI of the Compact resource. The client then performs a GET request on the Compact URI to retrieve the Compact resource.

Clients might instead use the HTTP `Prefer` header to get the Compact resource in a single HTTP request. If the client is confident that the resource has a preview, it makes an HTTP GET request on the resource URI using the `return=representation` preference [RFC7240] and an `include` parameter [LDP] asking for the Compact resource to be included in the response. The provider responds with a minimal representation of the resource and the Compact resource in the HTTP response body.

3.1.1 Discovering Compact Resources Using the HTTP Accept Header

This section is non-normative.

[OSLCUIPreview20] utilizes a MIME type extension, `application/x-oslc-compact+xml` to allow servers to use content negotiation with the HTTP `Accept` header to access the Compact representation of a resource instead of the content of the resource itself.

EXAMPLE 2: Requesting the Compact Resource

```
GET /bugs/324/screenshot HTTP/1.1
Host: example.com
Accept: application/x-oslc-compact+xml
```

This returns a Compact resource with `Content-Type: application/x-oslc-compact+xml` as defined in Appendix B. XML Representation Format.

3.1.2 Discovering Compact Resources Using the HTTP Link Header

This section is non-normative.

In responses to HTTP requests for resources that have a preview, servers include a `Link` header [RFC5988] where the link relation is `http://open-services.net/ns/core#Compact` and the target URI is the URI of the Compact resource.

EXAMPLE 3: Requesting the Resource Headers

```
HEAD /bugs/324/screenshot HTTP/1.1
Host: example.com
```

EXAMPLE 4: Response with a Compact Link

```
HTTP/1.1 200 OK
Content-Length: 45789
Content-Type: image/png
ETag: "678609cdee68e0fb8aea5f252b84a511"
Link: <http://example.com/bugs/324/screenshot?compact>; rel="http://open-services.net/ns/core#Compact"
```

Clients can request the Compact resource using the target URI of the `Link` header.

EXAMPLE 5: Requesting the Compact Resource

```
GET /bugs/324/screenshot?compact HTTP/1.1
Host: example.com
Accept: application/json
```

EXAMPLE 6: JSON Compact Response

```
HTTP/1.1 200 OK
Content-Length: 192
Content-Type: application/json; charset=UTF-8
ETag: "3bf6fbc90e11b3c2cc30acb534b452ea"
Vary: Accept,Accept-Language

{
    "title": "Screenshot of the problem",
    "shortTitle": "screenshot.png",
    "smallPreview": {
        "document": "http://example.com/bugs/324/screenshot?preview=small"
    }
}
```

Servers can also return a `Link` header in response to successful requests that create resources. This allows the client to get the Compact resource URI for new resources without making additional requests. Note that servers set the `Link` context using an `anchor` parameter if the request URI is not the same as the newly-created resource URI.

EXAMPLE 7: Creating a Resource

```
POST /bugs HTTP/1.1
Host: example.com
Content-Length: 153
Content-Type: text/turtle

@prefix dcterms: <http://purl.org/dc/terms/> .

<>      a              <http://example.com/ns#Bug> ;
        dcterms:title  "Something went wrong" .
```

EXAMPLE 8: Creation Response with a Compact Link

```
HTTP/1.1 201 Created
Content-Length: 0
Link: <http://example.com/bugs/478?compact>; rel="http://open-services.net/ns/core#Compact"; anchor="http://example.com/bugs/478"
Location: http://example.com/bugs/478
```

3.1.3 Discovering Compact Resources Using the HTTP Prefer Header

This section is non-normative.

Clients can request a Compact resource by making an HTTP GET request to the target resource's URI using the `return=representation` preference of the HTTP Prefer request header [RFC7240] and `include` parameter [LDP] value `http://open-services.net/ns/core#PreferCompact`. Servers supporting resource preview must support this method of discovery for resources with RDF or JSON representations.

EXAMPLE 9: Requesting the Compact Resource Using the Prefer Header

```
GET /bugs/324 HTTP/1.1
Host: example.com
Accept: application/json
Prefer: return=representation; include="http://open-services.net/ns/core#PreferCompact"
```

If the provider supports a preview for this resource and the request is successful, the response includes the Compact resource in its body.

The response content type is negotiated using the HTTP `Accept` request header. If the negotiated content type is `application/json`, the response body is a JSON object. The top-level JSON object has a `compact` property whose value is the JSON object describing the Compact resource in Appendix A. JSON Representation Format. The JSON may include other properties.

EXAMPLE 10: Preference-Applied Response

```
HTTP/1.1 200 OK
Content-Length: 315
Content-Type: application/json; charset=UTF-8
ETag: "f9d76afe5fbed1655c5768906db8958a"
Preference-Applied: return=representation
Vary: Accept,Accept-Language,Prefer

{
    "compact": {
        "title": "324: Need a fix <em>NOW</em>",
        "icon": "http://example.com/icons/defect.jpg",
        "largePreview": {
            "document": "http://example.com/bugs/324?preview=large",
            "hintHeight": "250px",
            "hintWidth": "400px"
        }
    }
}
```

Services may include a JSON-LD context in an `application/json` response. Clients who prefer RDF should request `text/turtle` or `application/ld+json` using the HTTP `Accept` request header, rather than `application/json`.

EXAMPLE 11: JSON Response with a JSON-LD Context

```
HTTP/1.1 200 OK
Content-Length: 788
Content-Type: application/json; charset=UTF-8
ETag: "d53d19be87fa9c61043c70bd91413dab"
Preference-Applied: return=representation
Vary: Accept,Accept-Language,Prefer

{
    "@id": "http://example.com/bugs/324",
    "@type": "http://example.com/ns#Bug",
    "@context": "http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/cs01/contexts/preview.jsonld",
    "compact": {
        "@id": "http://example.com/bugs/324?compact",
        "@type": "http://open-services.net/ns/core#Compact",
        "title": "324: Need a fix <em>NOW</em>",
        "shortTitle": "324",
        "icon": "http://example.com/icons/defect.jpg",
        "iconTitle": "Defect",
        "iconAltLabel": "Defect",
        "largePreview": {
            "document": "http://example.com/bugs/324?preview=large",
            "hintHeight": "250px",
            "hintWidth": "400px"
        },
        "smallPreview": {
            "document": "http://example.com/bugs/324?preview=small"
        }
    }
}
```


3.2 Displaying Previews

This section is non-normative.

When displaying a preview inside another HTML-based presentation, clients create an `iframe` element, setting its `src` attribute to the preview's document URI. Previews can contain HTML, stylesheets, and scripts and might not render properly outside of an `iframe`. Using the `iframe` element also sandboxes the preview, avoiding cross-site scripting vulnerabilities.

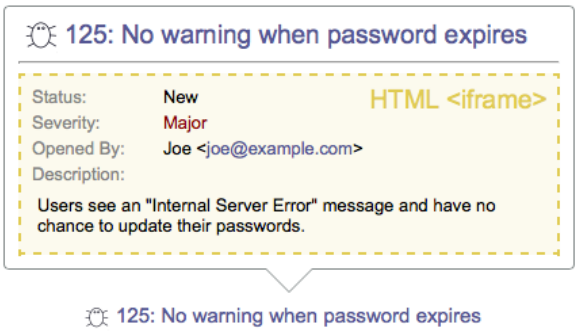

Fig. 2 Preview iframe

Typically, the server does not include the icon or title in the preview itself. The client can display these as needed. In Fig. 2 Preview iframe, the preview document is the content inside the yellow, dashed lines.

Resources can have both a small and large preview, and clients can choose which to display. Some clients display the small preview initially and let users expand to the large preview using a button or "Show more" link.

The Compact resource can include `oslc:hintHeight` and `oslc:hintWidth` properties for each preview. Values for `oslc:hintWidth` and `oslc:hintHeight` are expressed in ***length units*** specified in [CSS21].

Servers can also request dynamic resizing for small and large previews. The communication of dynamic size information happens within a web browser. JavaScript code running in the preview `iframe` sends a message that is received and acted upon by JavaScript code running in the `iframe`'s parenting window. This cross-frame communication is done using HTML 5 `postMessage`. Dynamic resizing will not work inside browsers that do not support `postMessage`.

A resize message begins with `oslc-resize:` followed by a JSON object describing the dimensions using properties `oslc:hintHeight` and `oslc:hintWidth`.

JSON Property Name	Type	Occurs	Description
`oslc:hintHeight`	String	Zero-or-one	Preferred height of the preview. Values are expressed using length units as specified in [CSS21].
`oslc:hintWidth`	String	Zero-or-one	Preferred width of the preview. Values are expressed using length units as specified in [CSS21].

For example, the following message requests that the preview be resized to height of 277 pixels and a width of 400 pixels.

EXAMPLE 12: Dyanmic Resize Message

```
oslc-resize:{"oslc:hintHeight": "277px", "oslc:hintWidth": "400px"}
```

The following JavaScript example sends a resize request to the parent window.

EXAMPLE 13: JavaScript postMessage Example

```
var size = {
    'oslc:hintHeight': '277px',
    'oslc:hintWidth': '400px'
};

if (window.postMessage && window.parent)  {
    window.parent.postMessage('oslc-resize:' + JSON.stringify(size), "*");
}
```

Clients can ignore a server's size hints and use other values. For instance, a client might choose another size if the preview is too large for the window.

4. Preview Guidance

This section is non-normative.

The following guidance is suggested for Client display of a resource link and rich previews. For purposes of this discussion, assume source resource A has a URI property that refers to target resource B:

4.1 Navigation to the target resource

This section is non-normative.

At any time, the user should be able to gesture that they would like to navigate the link to the target resource, regardless of how the link to the target resource happens to be rendered.

4.2 Default way to display the link to the target resource

This section is non-normative.

The typical way for forming and displaying the hyperlink should be based on information that is stored locally in source resource A (or that is generally known to the Client). Generally, link text can be derived from property values of resource A, and potentially from a string or literal property value in the reference to the target resource B, if such a property exists. Because these property values are part of the representation of the source resource A, they are available without consulting the target resource, and will be available even if the target resource B cannot be fetched. When available, Clients should display this string (as opposed to the URI) when presenting resource A to indicate a potential navigation to resource B. This is the basic presentation of a link to target resource B.

The default display of the link from A to B is visible to any user authorized to access resource A. The use of Compact representation information described below is only viable for users who are also authorized to access resource B.

4.3 Using the Compact representation title, short title, and icon

This section is non-normative.

A Compact representation of the resource may contain a more accurate and slightly richer label for a target resource (`dcterms:title` element), a short-form title for the resource (`oslc:shortTitle`), and a corresponding image (`oslc:icon`), possibly chosen from a set with different sizes (`oslc:iconSrcSet`), all of which may be based on the current state of the target resource. If this becomes known to the Client, the client should assume that this information is better and use it in forming the hyperlink that is displayed to the user. When available, `oslc:shortTitle` may be used instead of `dcterms:title` in presentations where visual space is severely limited.

Clients should not fetch Compact representations when there is perfectly usable default display information available. When designing a Client application, consideration should be given to the potential benefits of obtaining an improved title and icon for the target resource against the costs of preemptively fetching the Compact representation in terms of added load on servers and networks (which might only be apparent to users on slow networks or heavily loaded servers).

Clients should be wary of material obtained from non-trusted sources; in particular, the Client should check that the dcterms:title and oslc:shortTitle property values do not contain HTML markup beyond the specified set of simple elements.

4.4 Preview sizing

This section is non-normative.

When the resource does not supply preferred sizing for a preview, the Client should default to a reasonably generous value. The default value a Client uses may be different for different window and screen sizes.

4.5 Displaying a small preview

This section is non-normative.

If a user mouses or hovers over a displayed link, the Client should determine whether the target resource has a small preview (`oslc:smallPreview`). If it does, the Client should present the small preview document in a hovering widget. Since preview documents may contain arbitrary content, including HTML and scripts, Client must use iframes if embedding the preview document inside another HTML-based presentation.

The Client should not attempt to prefetch a Compact representation just to have the preview URIs in hand so that the hover can come up faster. There is a low chance that the user will make a gesture that would call for the display of a small preview. It would generally be a poor trade-off to increase overall system load just to decrease UI latency for low probability eventualities. Clients may wish to utilize lazy loading techniques to only access preview dialogs that users actually view, and can cache those dialogs for subsequent uses if needed to improve performance.

4.6 Displaying a large preview

This section is non-normative.

If the user then gestures that they want to see a bit more of the resource, the Client should determine whether the target resource has a large preview (`oslc:largePreview`). If it does, the Client should present the large preview document in a stationary widget that permits further user interaction. Again, since preview documents may contain arbitrary content, including HTML and scripts, Clients must use iframes if embedding the preview document inside another HTML-based presentation.

5. Implementation Conformance

5.1 General

Servers **MAY** choose to provide Compact resources for some resources and not others. [rp-1]

Resources with previews **MUST** support the HTTP OPTIONS method. [rp-2]

In responses to HTTP GET requests targeting resources that have a Compact resource, servers **SHOULD** either include a `Vary` response header with at least `Accept` and `Prefer` field values or a `Cache-Control` header value `no-store`. [rp-3]

Servers **MAY** consult the `Accept-Language` header on requests for a Compact resource (and on requests for preview documents) to return a resource for the requested natural language. [rp-4]

Clients **SHOULD** gracefully handle unsuccessful attempts to use previews. Previews may not be supported for all resources or may not work due to security or service availability issues. [rp-5]

5.2 Accept Header

In response to OPTIONS, HEAD or GET on resource, servers **MAY** include an `Accept: application/x-oslc-compact+xml` header to indicate the resource provides a compact representation. [rp-6]

Servers **MAY** support the `application/x-oslc-compact+xml` MIME type extension to allow clients to use content negotiation to access the Compact resource. However, this usage should be considered deprecated and is only provided for OSLC 2.0 compatibility. [rp-7]

Servers that do support the `application/x-oslc-compact+xml` MIME type extension **MUST** return the Compact resource using `Content-Type: application/x-oslc-compact+xml` as specified in [OSLCUIPreview20]. [rp-8]

5.3 Link Headers

In responses to successful HTTP requests for a target resource that has a Compact resource, servers **MUST** include a `Link` header [RFC5988] where:

- The context URI is the effective request URI,
- The link relation is `http://open-services.net/ns/core#Compact`, and
- The target URI is the URI of the Compact resource.

[rp-9]

> EXAMPLE 14: Example Link Header
>
> ```
> Link: <http://example.com/bugs/324/screenshot?compact>; rel="http://open-services.net/ns/core#Compact"
> ```

If a newly-created resource has a Compact resource, servers **MAY** return a `Link` header in response to the creation request where:

- The context URI is the URI of the newly-created resource,
- The link relation is `http://open-services.net/ns/core#Compact`, and
- The target URI is the URI of the associated Compact resource.

[rp-10]

> EXAMPLE 15: Example Link Header with Anchor
>
> ```
> Link: <http://example.com/bugs/478?compact>; rel="http://open-services.net/ns/core#Compact"; anchor="http://example.com/bugs/478"
> ```

5.4 Prefer Headers

Clients **MAY** request that the Compact resource is returned inline with the target resource using the `Prefer` request header [RFC7240] and

- Preference `return`, value `representation`
- Parameter `include` [LDP], value `http://open-services.net/ns/core#PreferCompact`.

[rp-11]

EXAMPLE 16: Example Prefer Header

```
Prefer: return=representation; include="http://open-services.net/ns/core#PreferCompact"
```

Servers **MUST** honor a client's request to in-line the Compact resource in JSON and RDF representations if the target resource has a Compact resource and the request is successful. [rp-12]

If the target resource does not exist or is not accessible to the requesting client, servers **MUST** return the same status code that it would have returned had the client not included the `Prefer` header in the request. [rp-13]

When servers in-line the Compact resource with the target resource in an `application/json` [RFC4627] response, the response entity **MUST** be a JSON object with a `compact` property where the value is the Compact resource JSON as described in Appendix A. JSON Representation Format. [rp-14]

Servers **MAY** choose to return only a subset of the target resource properties when a client requests that the Compact resource is returned in-line. [rp-15]

Clients **SHOULD** inspect the response body for the Compact resource even if the server omits the `Preference-Applied` header [RFC7240]. [rp-16]

5.5 Compact Resources

Servers **MUST** support the `application/json` [RFC4627] and `text/turtle` [turtle] media types for the Compact resource. [rp-17]

Servers **SHOULD** support the `application/ld+json` media type [JSON-LD] for the Compact resource. [rp-18]

The `application/json` Compact resource representation **MUST** be in the JSON format described in Appendix A. JSON Representation Format. [rp-19]

Servers **MAY** respond with JSON-LD in compacted document form when clients request the `application/json` representation of the Compact resource as long as the response meets the requirements for the JSON representation. [rp-20]

RDF representations of the Compact resource **MUST** conform to the shape specified in 6.1 Resource: Compact. [rp-21]

Content-Type `application/x-oslc-compact+xml` representations of the Compact resource **MUST** conform to the shape specified in Appendix B. XML Representation Format. [rp-22]

The Compact resource **MAY** have additional server-specific properties. [rp-23]

5.6 Previews

When displaying a preview inside another HTML presentation, clients **MUST** use an `iframe` element, setting its `src` attribute to the preview's document URI. [rp-24]

Servers **MUST** express the `oslc:hintWidth` and `oslc:hintHeight` properties of an `oslc:Preview` in length units as specified in [CSS21]. [rp-25]

Servers **MAY** use dynamic resizing for small and large previews in addition to `oslc:Preview` `oslc:hintHeight` and `oslc:hintWidth` properties. [rp-26]

Servers that support dynamic sizing **MUST** send resize messages to the parent window using the `Window.postMessage` method [whatwg-web-messaging] where the source of the event is the preview's window. [rp-27]

A resize message **MUST** consist of the characters `oslc-resize:` followed by a JSON object with at least one of the following properties:

- `oslc:hintHeight` or
- `oslc:hintWidth`.

[rp-28]

EXAMPLE 17: Example Resize Message

```
oslc-resize:{"oslc:hintHeight": "277px", "oslc:hintWidth": "400px"}
```

A resize message **MAY** also also consist of the characters `oslc-preview-height:` followed by the desired height in pixels. [rp-29]

EXAMPLE 18: Example Resize Height Message

```
oslc-preview-height:277
```

This `postMessage` format should be considered deprecated and is only provided for OSLC 2.0 compatibility.

The `oslc:hintHeight` and `oslc:hintWidth` property values in the resize message JSON **MUST** be length units as specified in [CSS21]. [rp-30]

Servers **MAY** send more than one dynamic resize request. [rp-31]

Clients **MAY** choose to ignore the size hints in `oslc:Preview` resources or in dynamic resize messages. [rp-32]

6. Resource Constraints

This document applies the following constraints to the OSLC Core vocabulary terms.

An OSLC server providing the Resource Preview capability **MUST** implement the vocabulary defined in this section. [rp-33]

6.1 Resource: Compact

- **Describes:** `http://open-services.net/ns/core#Compact`
- **Summary:** Describes how to display a resource preview.

Compact Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:title`	Zero-or-one	true	string	N/A	Unspecified	Title that may be used in the display of a link to the resource. The value should include only content that is valid inside an HTML element. Providers should include a dcterms:title property with an informative label for the resource. The title is typically shown to a user as a hyperlink. For a resource with no obvious title, Providers should omit the dcterms:title property. Providers must first HTML escape the contents of the dcterms:title before sending the response.
`oslc:icon`	Zero-or-one	true	Resource	Reference	Unspecified	URI of an image which may be used in the display of a link to the resource.
`oslc:iconAltLabel`	Zero-or-one	true	string	N/A	Unspecified	Alternative label used in association with the oslc:icon, such as HTML img tag's alt attribute.
`oslc:iconSrcSet`	Zero-or-one	true	string	N/A	Unspecified	Specification of a set of images of different sizes based on HTML img element srcset attribute.
`oslc:iconTitle`	Zero-or-one	true	string	N/A	Unspecified	Title used in association with the oslc:icon, such as HTML img tag's title attribute.
`oslc:largePreview`	Zero-or-one	true	AnyResource	Either	`oslc:Preview`	URI and sizing properties for an HTML document to be used for a large preview.
`oslc:shortTitle`	Zero-or-one	true	string	N/A	Unspecified	Abbreviated title which may be used in the display of a link to the resource. The value should include only content that is valid inside an HTML element. Providers should include an abbreviated title for the resource when possible. The abbreviated title is typically shown to a user as a hyperlink in presentations where visual space is limited. As a general guideline, the length of the abbreviated title should be 5 characters or less. A user-visible identifier that ordinarily appears in the dcterms:title, such as a defect number, makes for a good oslc:shortTitle value. When a resource has no obvious identifier or handle, Providers should omit the oslc:shortTitle property. Providers must first HTML escape the contents of the oslc:shortTitle before sending the response.
`oslc:smallPreview`	Zero-or-one	true	AnyResource	Either	`oslc:Preview`	URI and sizing properties for an HTML document to be used for a small preview.

6.2 Resource: Preview

- **Describes:** `http://open-services.net/ns/core#Preview`
- **Summary:** An HTML representation of a resource that can be embedded in another user interface.

Preview Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:document	Exactly-one	true	Resource	Reference	Unspecified	The URI of an HTML document to be used for the preview.
oslc:hintHeight	Zero-or-one	true	string	N/A	Unspecified	Recommended height of the preview. Values are expressed using length units as specified in [CSS21].
oslc:hintWidth	Zero-or-one	true	string	N/A	Unspecified	Recommended width of the preview. Values are expressed using length units as specified in [CSS21].

7. Conformance

Implementations of this specification need to satisfy the following conformance clauses.

Clause Number	Requirement
rp-1	Servers **MAY** choose to provide Compact resources for some resources and not others.
rp-2	Resources with previews **MUST** support the HTTP OPTIONS method.
rp-3	In responses to HTTP GET requests targeting resources that have a Compact resource, servers **SHOULD** either include a `Vary` response header with at least `Accept` and `Prefer` field values or a `Cache-Control` header value `no-store`.
rp-4	Servers **MAY** consult the `Accept-Language` header on requests for a Compact resource (and on requests for preview documents) to return a resource for the requested natural language.
rp-5	Clients **SHOULD** gracefully handle unsuccessful attempts to use previews. Previews may not be supported for all resources or may not work due to security or service availability issues.
rp-6	In response to OPTIONS, HEAD or GET on resource, servers **MAY** include an `Accept: application/x-oslc-compact+xml` header to indicate the resource provides a compact representation.
rp-7	Servers **MAY** support the `application/x-oslc-compact+xml` MIME type extension to allow clients to use content negotiation to access the Compact resource. However, this usage should be considered deprecated and is only provided for OSLC 2.0 compatibility.
rp-8	Servers that do support the `application/x-oslc-compact+xml` MIME type extension **MUST** return the Compact resource using `Content-Type: application/x-oslc-compact+xml` as specified in [OSLCUIPreview20].
rp-9	In responses to successful HTTP requests for a target resource that has a Compact resource, servers **MUST** include a `Link` header [RFC5988] where: • The context URI is the effective request URI, • The link relation is `http://open-services.net/ns/core#Compact`, and • The target URI is the URI of the Compact resource.
rp-10	If a newly-created resource has a Compact resource, servers **MAY** return a `Link` header in response to the creation request where: • The context URI is the URI of the newly-created resource, • The link relation is `http://open-services.net/ns/core#Compact`, and • The target URI is the URI of the associated Compact resource.
rp-11	Clients **MAY** request that the Compact resource is returned inline with the target resource using the `Prefer` request header [RFC7240] and • Preference `return`, value `representation` • Parameter `include` [LDP], value `http://open-services.net/ns/core#PreferCompact`.
rp-12	Servers **MUST** honor a client's request to in-line the Compact resource in JSON and RDF representations if the target resource has a Compact resource and the request is successful.
rp-13	If the target resource does not exist or is not accessible to the requesting client, servers **MUST** return the same status code that it would have returned had the client not included the `Prefer` header in the request.
rp-14	When servers in-line the Compact resource with the target resource in an `application/json` [RFC4627] response, the response entity **MUST** be a JSON object with a `compact` property where the value is the Compact resource JSON as described in Appendix A. JSON Representation Format.
rp-15	Servers **MAY** choose to return only a subset of the target resource properties when a client requests that the Compact resource is returned in-line.
rp-16	Clients **SHOULD** inspect the response body for the Compact resource even if the server omits the `Preference-Applied` header [RFC7240].
rp-17	Servers **MUST** support the `application/json` [RFC4627] and `text/turtle` [turtle] media types for the Compact resource.
rp-18	Servers **SHOULD** support the `application/ld+json` media type [JSON-LD] for the Compact resource.
rp-19	The `application/json` Compact resource representation **MUST** be in the JSON format described in Appendix A. JSON Representation Format.

Clause Number	Requirement
rp-20	Servers **MAY** respond with JSON-LD in compacted document form when clients request the `application/json` representation of the Compact resource as long as the response meets the requirements for the JSON representation.
rp-21	RDF representations of the Compact resource **MUST** conform to the shape specified in 6.1 Resource: Compact.
rp-22	Content-Type `application/x-oslc-compact+xml` representations of the Compact resource **MUST** conform to the shape specified in Appendix B. XML Representation Format.
rp-23	The Compact resource **MAY** have additional server-specific properties.
rp-24	When displaying a preview inside another HTML presentation, clients **MUST** use an `iframe` element, setting its `src` attribute to the preview's document URI.
rp-25	Servers **MUST** express the `oslc:hintWidth` and `oslc:hintHeight` properties of an `oslc:Preview` in length units as specified in [CSS21].
rp-26	Servers **MAY** use dynamic resizing for small and large previews in addition to `oslc:Preview` `oslc:hintHeight` and `oslc:hintWidth` properties.
rp-27	Servers that support dynamic sizing **MUST** send resize messages to the parent window using the `Window.postMessage` method [whatwg-web-messaging] where the source of the event is the preview's window.
rp-28	A resize message **MUST** consist of the characters `oslc-resize:` followed by a JSON object with at least one of the following properties: • `oslc:hintHeight` or • `oslc:hintWidth`.
rp-29	A resize message **MAY** also also consist of the characters `oslc-preview-height:` followed by the desired height in pixels.
rp-30	The `oslc:hintHeight` and `oslc:hintWidth` property values in the resize message JSON **MUST** be length units as specified in [CSS21].
rp-31	Servers **MAY** send more than one dynamic resize request.
rp-32	Clients **MAY** choose to ignore the size hints in `oslc:Preview` resources or in dynamic resize messages.
rp-33	An OSLC server providing the Resource Preview capability **MUST** implement the vocabulary defined in this section.

Appendix A. JSON Representation Format

A Compact resource JSON representation might look like the following for the target resource `http://example.com/bugs/324`:

EXAMPLE 19: Compact Resource JSON

```
{
    "title": "324: Need a fix <em>NOW</em>",
    "shortTitle": "324",
    "icon": "http://example.com/icons/defect.jpg",
    "iconSrcSet": "http://example.com/icons/smallIcon.png 16w, http://example.com/icons/largeIcon.png 64w",
    "iconTitle": "Defect",
    "iconAltLabel": "Defect",
    "largePreview": {
        "document": "http://example.com/bugs/324?preview=large",
        "hintHeight": "250px",
        "hintWidth": "400px"
    },
    "smallPreview": {
        "document": "http://example.com/bugs/324?preview=small"
    }
}
```

The JSON representation has the following constraints:

- The value of the `title` or `shortTitle` properties **MUST**, if present, only contain markup valid in an HTML `span` element. To afford clients the greatest flexibility to style the text to match the context in which it is being displayed, the titles **SHOULD** be plain text with HTML markup used only to emphasize words or phrases or to convey additional information about the target resource.
- `icon`, if present, **MUST** be the URI of an image. The image **SHOULD** be square. A client **MAY** scale the image as needed.
- `iconSrcSet`, if present, **MUST** provide a comma-separated list of image URIs along with image size as specified for the HTML img srcset attribute.
- `iconAltLabel` and `iconTitle`, if present, **MUST** only have string content with no markup.
- `smallPreview` and `largePreview`, if present, **MAY** have any number of properties, including none.
  - The value of `smallPreview` and `largePreview`, if present, **MUST** be the JSON representation of an `oslc:Preview`.
    - A `Preview` **MUST** include a document property. It **MAY** have any number of other properties.
    - `document` **MUST** be the URI of a document containing a preview of the resource.
    - `hintWidth` and `hintHeight`, if present, **MUST** be expressed using length units as specified in [CSS21].

A.1 JSON-LD Context

This section is non-normative.

The following JSON-LD Context may be used with JSON Compact representations.

EXAMPLE 20: JSON-LD Context for a Compact Resource

```
{
    "_comment": "JSON-LD context for OSLC Resource Preview",
    "@context": {
        "@base": "@https://tools.oasis-open.org/version-control/svn/oslc-core/trunk/specs/contexts/preview.jsonld",
        "oslc" : "http://open-services.net/ns/core#",
        "rdf" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
        "dcterms" : "http://purl.org/dc/terms/",
        "@vocab": "http://open-services.net/ns/core#",

        "dcterms:title": {
            "@id": "dcterms:title",
            "@type": "xsd:string"
        },
        "compact": {
            "@id": "oslc:compact",
            "@type": "@id"
        },
        "icon": {
            "@id": "oslc:icon",
            "@type": "@id"
        },
        "iconSrcSet": {
            "@id": "oslc:iconSrcSet",
            "@type": "@id"
```



```
        },
        "iconAltLabel": {
            "@id": "oslc:iconAltLabel",
            "@type": "xsd:string"
        },
        "hintHeight": {
            "@id": "oslc:hintHeight",
            "@type": "xsd:integer"
        },
        "hintWidth": {
            "@id": "oslc:hintWidth",
            "@type": "xsd:integer"
        },
        "shortTitle": {
            "@id": "oslc:iconTitle",
            "@type": "xsd:string"
        },
        "smallPreview": {
            "@id": "oslc:smallPreview",
            "@type": "oslc:Preview"
        },
        "largePreview": {
            "@id": "oslc:largePreview",
            "@type": "oslc:Preview"
        }
    }
}
```

A.2 JSON Schema

This section is non-normative.

The following JSON Schema [JSONSchema] describes the Compact JSON representation.

EXAMPLE 21: JSON Schema for a Compact Resource

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "OSLC 3.0 Compact JSON",
    "type": "object",
    "properties": {
        "title": {
            "id": "title",
            "type": "string"
        },
        "shortTitle": {
            "id": "shortTitle",
            "type": "string"
        },
        "icon": {
            "id": "icon",
            "type": "string"
        },
        "iconTitle": {
            "id": "iconTitle",
            "type": "string"
        },
        "iconAltLabel": {
            "id": "iconAltLabel",
            "type": "string"
        },
        "largePreview": {
            "id": "largePreview",
            "type": "object",
            "properties": {
                "document": {
                    "id": "document",
                    "type": "string"
                },
                "hintHeight": {
                    "id": "hintHeight",
                    "type": "string",
                    "pattern": "^[0-9]+(\\.[0-9]+)?(em|ex|in|cm|mm|pt|pc|px)$"
                },
                "hintWidth": {
                    "id": "hintWidth",
                    "type": "string",
                    "pattern": "^[0-9]+(\\.[0-9]+)?(em|ex|in|cm|mm|pt|pc|px)$"
                }
```



```
            },
            "required": ["document"]
        },
        "smallPreview": {
            "id": "smallPreview",
            "type": "object",
            "properties": {
                "document": {
                    "id": "document",
                    "type": "string"
                },
                "hintHeight": {
                    "id": "hintHeight",
                    "type": "string",
                    "pattern": "^[0-9]+(\\.[0-9]+)?(em|ex|in|cm|mm|pt|pc|px)$"
                },
                "hintWidth": {
                    "id": "hintWidth",
                    "type": "string",
                    "pattern": "^[0-9]+(\\.[0-9]+)?(em|ex|in|cm|mm|pt|pc|px)$"
                }
            },
            "required": ["document"]
        }
    }
}
```


Appendix B. XML Representation Format

A Compact resource XML representation woould look like the following for the target resource `http://example.com/bugs/12345`:

EXAMPLE 22: Compact Resource XML

```
<?xml version="1.0" encoding="UTF-8"?>

<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:oslc="http://open-services.net/ns/core#">

 <oslc:Compact
   rdf:about="http://example.com/bugs/12345">

   <dcterms:title> 12345: Need a &quot;fix&quot; &lt;em&gt;NOW&lt;/em&gt; </dcterms:title>
   <oslc:shortTitle> 12345 </oslc:shortTitle>
   <oslc:icon rdf:resource="http://example.com/icons/defect.jpg" />
   <oslc:iconAltLabel>Defect</oslc:iconAltLabel>
   <oslc:iconTitle>Defect</oslc:iconTitle>

   <oslc:smallPreview>
      <oslc:Preview>
         <oslc:document rdf:resource="http://example.com/bugs/12345?hover=small" />
      </oslc:Preview>
   </oslc:smallPreview>


   <oslc:largePreview>
      <oslc:Preview>
         <oslc:document rdf:resource="http://example.com/bugs/12345?hover=large" />
         <oslc:hintWidth> 60em </oslc:hintWidth>
         <oslc:hintHeight> 20em </oslc:hintHeight>
      </oslc:Preview>
   </oslc:largePreview>

 </oslc:Compact>

</rdf:RDF>
```

The XML representation has the following syntactic and semantic constraints:

- The root element **MUST** be `rdf:RDF` and **SHOULD** identify the namespaces for RDF (`http://www.w3.org/1999/02/22-rdf-syntax-ns#`), Dublin Core (`http://purl.org/dc/terms/`), and OSLC (`http://open-services.net/ns/core#`), as well as namespaces for any additional Provider supplied properties.
- The `oslc:Compact` element **MUST** be a child of the root element and **MUST** specify an `rdf:about` attribute whose value is the URI of the target resource.
  - The `oslc:Compact` element **MAY** have any number of child elements, including none. Each child element specifies one property of the target resource. The order of child elements is not significant.
  - Each of the `dcterms:title`, `oslc:shortTitle`, `oslc:icon`, `oslc:smallPreview`, and `oslc:largePreview` child elements **MAY** occur at most once.
  - The content of a `dcterms:title` or `oslc:shortTitle` element **MUST** be limited to any XHTML element. To afford clients the greatest flexibility to style the text to match the context in which it is being displayed, the titles **SHOULD** be plain text with XHTML markup used only to emphasize words or phrases or to convey additional information about the target resource.
  - An `oslc:icon` element **MUST** have an `rdf:resource` attribute whose value is a URI of an image. The image **SHOULD** be 16x16 pixels in size.
  - The `oslc:iconAltLabel` and `oslc:iconTitle` elements **MUST** only have string content.
  - The `oslc:smallPreview` and `oslc:largePreview` elements **MAY** have any number of child elements, including none.
    - The `oslc:Preview` element **MUST** be a child of an `oslc:smallPreview` or `oslc:largePreview` element and **MAY** occur at most once.
      - An `oslc:Preview` element **MAY** have any number of child elements, including none. Each child element specifies one property of the Preview. The order of child elements is not significant.
      - Each of the `oslc:document`, `oslc:hintWidth`, `oslc:hintHeight`, and `oslc:initialHeight` child elements **MAY** occur at most once.
      - The `oslc:document` element **MUST** have an `rdf:resource` attribute whose value is a URI of a document containing a preview of the resource.
      - The content of an `oslc:hintWidth`, `oslc:hintHeight`, or `oslc:initialHeight` element **MUST** be a valid relative length value.

- All elements **MAY** have additional atttributes not specified here. All elements, other than `dcterms:title` and `oslc:shortTitle`, **MAY** include additional child elements not specified here. Consumers **MUST** quietly ignore unknown elements and attributes encountered in a Compact representation.

OASIS OPEN PROJECTS

OSLC Core Version 3.0. Part 4: Delegated Dialogs

OASIS Standard
26 August 2021

This stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/dialogs.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/dialogs.pdf

Previous stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/dialogs.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/dialogs.pdf
(published as Project Specification)

Latest stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/dialogs.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/dialogs.pdf

Latest version:

https://open-services.net/spec/core/latest

Latest editor's draft:

https://open-services.net/spec/core/latest-draft

Open Project:

OASIS Open Services for Lifecycle Collaboration (OSLC) OP

Project Chairs:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Editors:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Additional components:

This specification is one component of a Work Product that also includes:

- *OSLC Core Version 3.0. Part 1: Overview.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/oslc-core.html
- *OSLC Core Version 3.0. Part 2: Discovery*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/discovery.html
- *OSLC Core Version 3.0. Part 3: Resource Preview.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-preview.html
- *OSLC Core Version 3.0. Part 4: Delegated Dialogs (this document).* https://docs.oasis-open-projects.org/oslc-

op/core/v3.0/os/dialogs.html

- *OSLC Core Version 3.0. Part 5: Attachments*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/attachments.html
- *OSLC Core Version 3.0. Part 6: Resource Shape*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-shape.html
- *OSLC Core Version 3.0. Part 7: Vocabulary*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.html
- *OSLC Core Version 3.0. Part 8: Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.html
- *OSLC Core Version 3.0. Machine Readable Vocabulary Terms*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.ttl
- *OSLC Core Version 3.0. Machine Readable Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.ttl

Related work:

This specification is related to:

- *OSLC Core Version 3.0: Link Guidance*. https://oslc-op.github.io/oslc-specs/notes/link-guidance.html

RDF Namespaces:

http://open-services.net/ns/core#

Abstract:

Delegated dialogs allow one application to embed a creation or selection UI into another using HTML `iframe` elements and JavaScript code. The embedded dialog notifies the parent page of resize or submit events using HTML5 `postMessage`.

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the “Latest stage” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Open Project are listed at https://github.com/oslc-op/oslc-specs.

Comments on this work can be provided by opening issues in the project repository or by sending email to the project’s public comment list oslc-op@lists.oasis-open-projects.org.

The English version of this specification is the only normative version. Non-normative translations may also be available. Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:
[OSLC-Dialogs-3.0]
OSLC Core Version 3.0. Part 4: Delegated Dialogs. Edited by Jim Amsden and Andrii Berezovskyi. 26 August 2021. OASIS Standard. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/dialogs.html. Latest stage: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/dialogs.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

Table of Contents

1. Introduction

This section is non-normative.

OSLC specifications target specific integration scenarios. In some cases, allowing one application to delegate to a user interface defined in another application is a more effective way to support a use case than an HTTP interface that can only be accessed programmatically. There are two cases where this is especially true:

- *Resource creation*: when a user of a web application needs to create a new resource in another application. In this case, the web application asks the other server to provide a UI for resource creation, and the server notifies the application when the creation has been completed or canceled by the user.
- *Resource selection*: when a user of a web application needs to pick a resource managed by another application. In this case, the web application asks the other server to provide a UI for resource selection, and the server notifies the application when a resource or resources has been selected or if the selection was canceled.

Delegated dialogs support these two cases. They allow one application to embed a creation or selection UI into another using HTML `iframe` elements and JavaScript code.

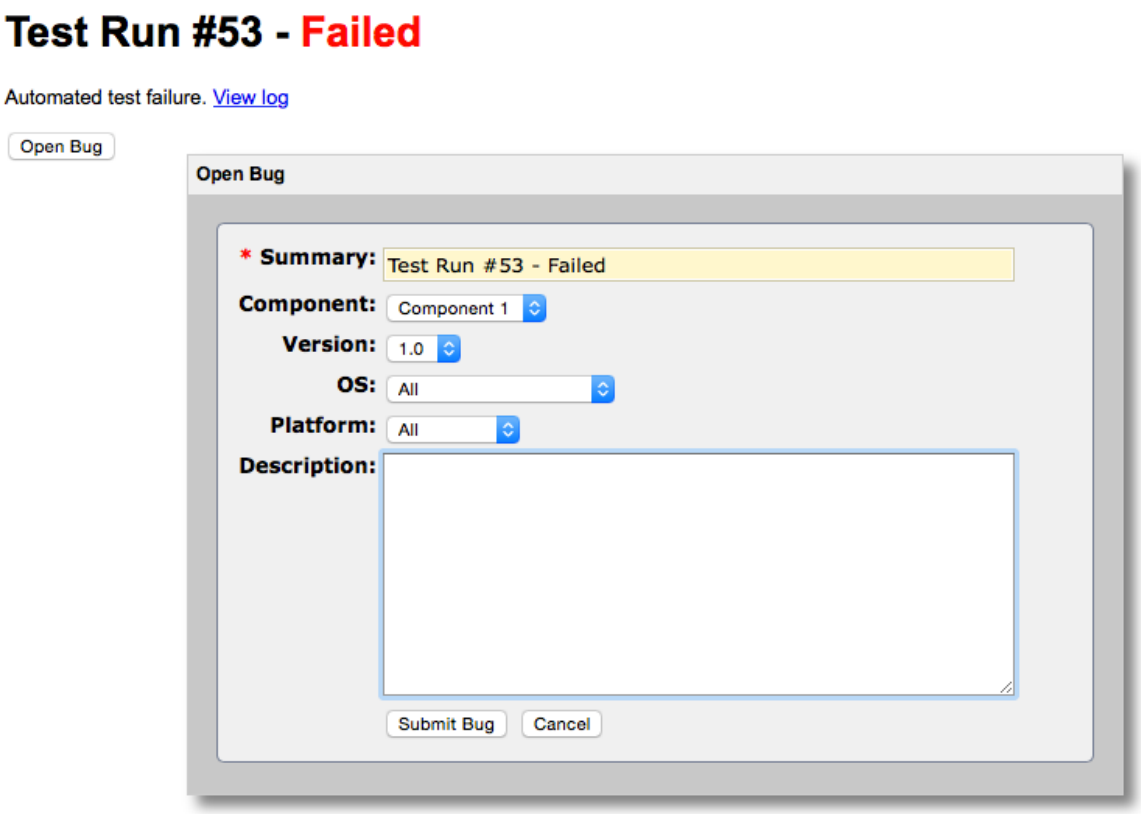

Fig. 1 Creation Dialog

Fig. 1 Creation Dialog depicts what a defect creation dialog might look like inside a quality management tool, displayed when the user clicks the *Open Bug* button. The dialog content itself comes from the change management tool, running on another server. The dialog displays seamlessly inside the quality management application, however, while retaining the change management tool's look and feel and capabilities.

Delegated dialogs provide a simple way for an end user to create or select resources from another application, thus eliminating the need to rebuild the other application's dialog and its application logic.

1.1 Terminology

Terminology uses and extends the terminology and capabilities of OSLC Core Overview, W3C Linked Data Platform [LDP], W3C's Architecture of the World Wide Web [WEBARCH], and Hyper-text Transfer Protocol [HTTP11].

The following terms are used in discussions of dialogs:

Dialog
: A web page for creating or selecting resources to be embedded inside another application.

Dialog Descriptor
: A resource that describes information about a dialog such as its title and dimensions. In RDF representations, the dialog descriptor has RDF type `oslc:Dialog`. See 5. Resource Constraints.

Prefill
: A method of setting initial field values in a creation dialog.

1.2 References

1.2.1 Normative references

[CSP]

M. West; A. Barth; D. Veditz. *Content Security Policy Level 2*. W3C. W3C Recommendation. URL: https://www.w3.org/TR/CSP2/

[CSS21]

Bert Bos; Tantek Çelik; Ian Hickson; Håkon Wium Lie. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C, 7 June 2011. W3C Recommendation. URL: https://www.w3.org/TR/CSS21/

[HTTP11]

R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. IETF, June 2014. Proposed Standard. URL: https://httpwg.org/specs/rfc7230.html

[LDP]

Steve Speicher; John Arwe; Ashok Malhotra. *Linked Data Platform 1.0*. W3C, 26 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/ldp/

[OSLCCore2]

S. Speicher; D. Johnson. *OSLC Core 2.0*. http://open-services.net. Finalized. URL: http://open-services.net/bin/view/Main/OslcCoreSpecification

[RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF, March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

[RFC3986]

T. Berners-Lee; R. Fielding; L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. IETF, January 2005. Internet Standard. URL: https://www.rfc-editor.org/rfc/rfc3986

[RFC5988]

M. Nottingham. *Web Linking*. IETF, October 2010. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc5988

[RFC7240]

J. Snell. *Prefer Header for HTTP*. IETF, June 2014. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc7240

[RFC8174]

B. Leiba. *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*. IETF, May 2017. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc8174

[RFC8259]

T. Bray, Ed.. *The JavaScript Object Notation (JSON) Data Interchange Format*. IETF, December 2017. Internet Standard. URL: https://www.rfc-editor.org/rfc/rfc8259

[RFC8288]

M. Nottingham. *Web Linking*. IETF, October 2017. Proposed Standard. URL: https://httpwg.org/specs/rfc8288.html

[whatwg-web-messaging]

WHATWG contributors. *HTML. Part 9.4: Cross-document messaging*. WHATWG. Living Standard. URL: https://html.spec.whatwg.org/multipage/web-messaging.html#web-messaging

1.2.2 Informative references

[WEBARCH]

Ian Jacobs; Norman Walsh. *Architecture of the World Wide Web, Volume One*. W3C, 15 December 2004. W3C Recommendation. URL: https://www.w3.org/TR/webarch/

1.3 Typographical Conventions and Use of RFC Terms

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this specification are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Discovering Dialogs

This section is non-normative.

Clients can discover dialogs in three ways:

- 2.1 Discovering Dialogs Using the Link Header
- 2.2 Discovering Dialogs Using the Prefer Header
- 2.3 Discovering Dialogs Using the Service resource

2.1 Discovering Dialogs Using the Link Header

This section is non-normative.

LDP containers [LDP] advertise their support for dialogs using the HTTP `Link` header [RFC8288]. Each dialog type, creation or selection, has its own link relation.

Dialog Type	Link Relation
Creation	`http://open-services.net/ns/core#creationDialog`
Selection	`http://open-services.net/ns/core#selectionDialog`

The client discovers the dialogs by making an HTTP OPTIONS request on the container.

EXAMPLE 1: An OPTIONS Request on the Container

```
OPTIONS /bugs/ HTTP/1.1
Host: example.com
```

The server response contains a `Link` header with URLs to the dialog descriptors.

EXAMPLE 2: Response with Dialog Link Headers

```
HTTP/1.1 204 No Content
Date: Thu, 12 Jun 2014 18:26:59 GMT
Allow: GET,POST,OPTIONS,HEAD
Accept-Post: text/turtle,application/ld+json
Link: <http://www.w3.org/ns/ldp#BasicContainer>; rel="type",
      <http://www.w3.org/ns/ldp#Resource>; rel="type",
      <http://example.com/dialogs/createBug>; rel="http://open-services.net/ns/core#creationDialog",
      <http://example.com/dialogs/selectBug>; rel="http://open-services.net/ns/core#selectionDialog"
```

The client then requests the dialog descriptor which contains an `oslc:dialog` property whose value is a link to the HTML dialog to be displayed.

EXAMPLE 3

```
GET /dialogs/selectBug HTTP/1.1
Host: example.com
```

EXAMPLE 4

```
HTTP/1.1 200 OK
Content-Type: text/turtle
Transfer-Encoding: chunked
```



```
@prefix oslc:  <http://open-services.net/ns/core#> .
@prefix dcterms: <http://purl.org/dc/terms/> .

<>      a                 oslc:Dialog ;
        oslc:dialog       <http://example.com/dialogs/selectBug/form> ;
        oslc:hintHeight   "600px" ;
        oslc:hintWidth    "400px" ;
        oslc:label        "Select Bug" ;
        oslc:resourceType <http://open-services.net/ns/cm#Bug> ;
        dcterms:title     "Select Bug from Product Z" .
```

2.2 Discovering Dialogs Using the Prefer Header

This section is non-normative.

Clients can also use the HTTP **`Prefer`** header to find the dialogs for a container. The client makes an HTTP GET request on the resource URI using the **`return=representation`** preference [RFC7240] and **`include`** parameter [LDP] value **`http://open-services.net/ns/core#PreferDialog`**. The server responds with the dialog descriptors in the response body. Clients should also use **`include`** parameter value **`http://www.w3.org/ns/ldp#PreferMinimalContainer`** so that the response doesn't include unnecessary data.

The following example shows a container that supports creation and selection dialogs:

EXAMPLE 5

```
GET /bugs/ HTTP/1.1
Host: example.com
Accept: text/turtle
Prefer: return=representation;
        include="http://open-services.net/ns/core#PreferDialog http://www.w3.org/ns/ldp#PreferMinimalContainer"
```

EXAMPLE 6

```
HTTP/1.1 200 OK
Content-Type: text/turtle
ETag: "_87e52ce291112"
Link: <http://www.w3.org/ns/ldp#BasicContainer>; rel="type",
      <http://www.w3.org/ns/ldp#Resource>; rel="type"
Accept-Post: text/turtle, application/ld+json
Allow: POST,GET,OPTIONS,HEAD
Preference-Applied: return=representation
Vary: Accept,Prefer
Transfer-Encoding: chunked

@prefix ex:    <http://example.com/vocab#> .
@prefix oslc:  <http://open-services.net/ns/core#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix ldp:   <http://www.w3.org/ns/ldp#> .

<http://example.com/bugs/>
        a                     ldp:BasicContainer ;
        oslc:creationDialog   <http://example.com/dialogs/createBug> ;
        oslc:selectionDialog  <http://example.com/dialogs/selectBug> ;
        dcterms:title         "Bugs Records for Product Z" .

<http://example.com/dialogs/createBug>
        a                 oslc:Dialog ;
        oslc:dialog       <http://example.com/dialogs/createBug/form> ;
        oslc:hintHeight   "600px" ;
        oslc:hintWidth    "400px" ;
        oslc:label        "New Bug" ;
        oslc:resourceType <http://open-services.net/ns/cm#Bug> ;
        dcterms:title     "Report Bug (Product Z)" .

<http://example.com/dialogs/selectBug>
```



```
        a                   oslc:Dialog ;
        oslc:dialog         <http://example.com/dialogs/selectBug/form> ;
        oslc:hintHeight     "600px" ;
        oslc:hintWidth      "400px" ;
        oslc:label          "Select Bug" ;
        oslc:resourceType   <http://open-services.net/ns/cm#Bug> ;
        dcterms:title       "Select Bug (Product Z)" .
```

2.3 Discovering Dialogs Using the Service resource

This section is non-normative.

Servers may also advertise their support for dialogs using the `oslc:creationDialog` and `oslc:selectionDialog` properties of the Service discovery resource.

The client discovers the dialogs by making a GET request on the Service resource and accessing these properties.

EXAMPLE 7: An OPTIONS Request on the Container

```
GET /serviceproviders/bugs/services.xml HTTP/1.1
Host: example.com
Accept: application/rdf+xml
```

The server response contains a Service resource which contains properties with URLs to the dialog descriptors.

EXAMPLE 8: Service Resource

```
HTTP/1.1 200 OK
Date: Thu, 12 Jun 2014 18:26:59 GMT
Content-Type: application/rdf+xml

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:oslc="http://open-services.net/ns/core#">
<oslc:ServiceProvider rdf:about="https://example.com/serviceproviders/bugs/services.xml">
<oslc:creationDialog>
  <oslc:Dialog>
    <dcterms:title rdf:parseType="Literal">New Bug</dcterms:title>
    <oslc:label>Bug Change Request</oslc:label>
    <oslc:usage rdf:resource="http://open-services.net/ns/cm#requirementsChangeRequest"/>
    <oslc:resourceType rdf:resource="http://open-services.net/ns/cm#ChangeRequest"/>
    <oslc:dialog rdf:resource="http://example.com/dialogs/createBug/form"/>
    <oslc:hintWidth>680px</oslc:hintWidth>
    <oslc:hintHeight>505px</oslc:hintHeight>
  </oslc:Dialog>
</oslc:creationDialog>
<oslc:selectionDialog>
  <oslc:Dialog>
    <dcterms:title rdf:parseType="Literal">Select Bug</dcterms:title>
    <oslc:label>Bug</oslc:label>
    <oslc:usage rdf:resource="http://open-services.net/ns/core#default"/>
    <oslc:resourceType rdf:resource="http://open-services.net/ns/cm#ChangeRequest"/>
    <oslc:dialog rdf:resource="http://example.com/dialogs/selectBug/form"/>
    <oslc:hintWidth>550px</oslc:hintWidth>
    <oslc:hintHeight>460px</oslc:hintHeight>
  </oslc:Dialog>
</oslc:selectionDialog>
</oslc:ServiceProvider>
</rdf:RDF>
```

The client then requests the dialog to display using the discovered URI in the `oslc:dialog` property in order to display the desired form.

EXAMPLE 9

```
GET /dialogs/selectBug/form HTTP/1.1
Host: example.com
```


3. Using Dialogs

This section is non-normative.

Servers can offer two kinds of dialogs, selection dialogs and creation dialogs. Clients use selection dialogs when they want a user to pick a resource from another application. They use creation dialogs when they want a user to create a new resource in another application.

A client can open the dialog in a new browser window, or it can embed the dialog in another page by creating an `iframe` element and setting the `src` attribute to the URI of the dialog to be included.

EXAMPLE 10: Display a Dialog

```
var iframe =
       document.createElement("iframe");
   iframe.style.border = 0;
   iframe.style.width = "600px";  // or preferred dialog width
   iframe.style.height = "400px"; // or preferred dialog height
   iframe.src = "http://example.com/dialogs/createBug/form";
   document.getElementById("dialogContainer").appendChild(iframe);
```

Clients might choose to place dialogs inside page elements styled to look like a dialog window.

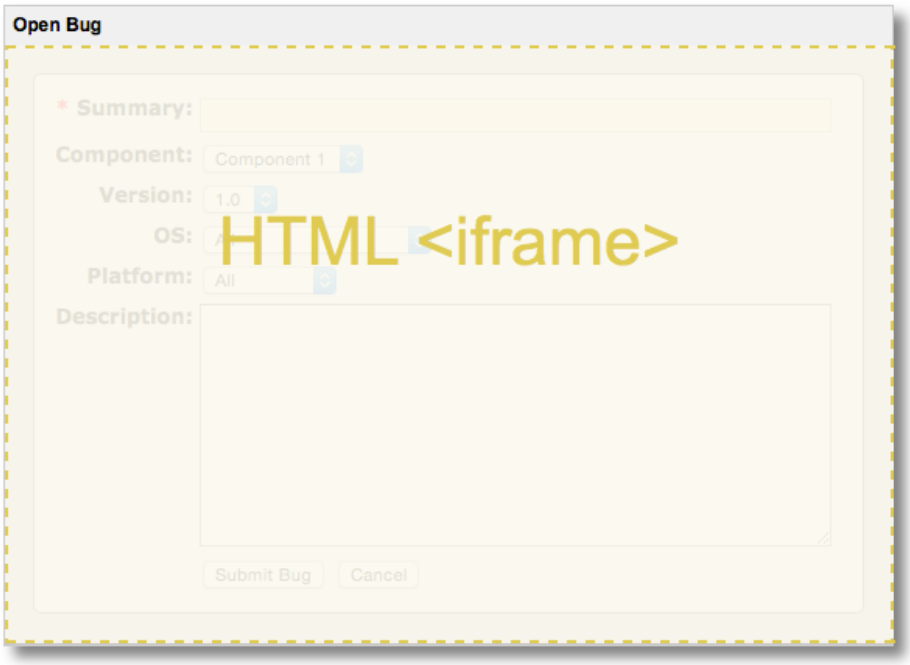

Fig. 2 Dialog iframe

The `iframe` itself, outlined in Fig. 2 Dialog iframe, contains the dialog content from the server.

Dialogs send results back to the client using the HTML5 function `Window.postMessage` [whatwg-web-messaging]. `postMessage` is called on `window.opener` if set. Otherwise, `postMessage` is called on `window.parent`.

When embedded in an `iframe`, dialogs may also request to be resized dynamically. These requests are the same as specified in OSLC Resource Preview and also use `postMessage`. Clients can distinguish between results and resize messages from the message prefix. Messages have the following prefixes:

Message Prefix	Meaning
`oslc-response:`	The user has created or selected resources or canceled the dialog.
`oslc-resize:`	The dialog is asking to be resized.

A stringified JSON object will be concatenated to the message prefix. For dialog responses, the JSON is described by Appendix A. Dialog Results JSON. It is a JSON object with an `oslc:results` array of results. Each result is a JSON object with an `rdf:resource` property whose value is the resource URI. The result may also have an `oslc:label` that can be useful for client to display a label for the delegated dialog.

EXAMPLE 11: Dialog Results Message

```
oslc-response:{"oslc:results":
  [{ "oslc:label": "bug 123: server crash", "rdf:resource": "http://example.com/bug123" }]}
```

An empty array indicates the dialog was canceled.

For resize requests, the JSON is an object with an `oslc:hintHeight` property, an `oslc:hintWidth` property, or both.

EXAMPLE 12: Dyanmic Resize Message

```
oslc-resize:{"oslc:hintHeight": "277px", "oslc:hintWidth": "400px"}
```

3.1 The Client's Responsibilities

This section is non-normative.

1. Open the dialog in a new window using the `Window.open()` method or embed the dialog in an `iframe`, setting `iframe src` to the URI of the dialog.
2. Add a `message` listener to receive messages from the dialog.
3. Listen for `message` events, ignoring unrecognized events from other sources or those not prefixed with `oslc-response:`
4. When message from the dialog indicates a completed action, free resources and handle the action.

EXAMPLE 13: Listen for Dialog Results

```
window.addEventListener("message", function(event) {
    // Make sure the message originated from the same origin as the dialog.
    if (event.origin !== "http://example.com") {
        return;
    }

    // Make sure the message starts with oslc-response:
    var message = event.data;
    if (message.indexOf("oslc-response:") !== 0) {
        return;
    }

    // Handle each result.
    var response = JSON.parse(message.substr("oslc-response:".length));
    var results = response["oslc:results"];
    for (var i = 0; i < results.length; i++) {
        var label = results[i]["oslc:label"];
        var uri = results[i]["rdf:resource"];
        // Handle resource...
    }

    // Remove the dialog from the page.
    var dialogContainer = document.getElementById('dialogContainer');
    dialogContainer.removeChild(dialogContainer.firstChild);
}, false);
```


3.2 The Server's Responsibilities

This section is non-normative.

1. Provide the dialog, an HTML page for resource creation or selection.
2. Allow the user to perform resource creation or selection.
3. Once the user has created or selected a resource or canceled the dialog, send notification using `postMessage` to the page's opener or parent window prefixed with `"oslc-response:"`.

The following JavaScript code example shows how a dialog would send a response using `postMessage`, taking into account pages that the dialog might be in its own window or an `iframe`.

EXAMPLE 14

```
function respondWithSelection(label, uri) {
    var response = {
        "oslc:results": [ {
            "oslc:label": label,
            "rdf:resource": uri
        } ]
    };
    (window.opener || window.parent).postMessage("oslc-response:" + JSON.stringify(response), "*");
}
```

3.3 Prefill

This section is non-normative.

Servers may support setting the initial values in a dialog. A client can test if a dialog supports prefill by making an HTTP OPTIONS request to the dialog descriptor URI. Prefill can be used with creation dialogs to provide a template of initial values. Clients can use prefill with selection dialogs to inform servers about the users desired content in the selection dialog.

EXAMPLE 15

```
OPTIONS /dialogs/createBug HTTP/1.1
Host: example.com
```

If the server supports prefill for this dialog, it includes POST among the methods in the `Allow` response header.

EXAMPLE 16

```
HTTP/1.1 204 No Content
Allow: GET,POST,HEAD,OPTIONS
```

To prefill content, clients POST the resource representation with the desired initial values to the delegated dialog descriptor URI.

EXAMPLE 17: Prefilling a Bug Creation Dialog

```
POST /dialogs/createBug HTTP/1.1
Host: example.com
Content-Type: text/turtle
Transfer-Encoding: chunked

@prefix oslc_cm: <http://open-services.net/ns/cm#> .
@prefix dcterms: <http://purl.org/dc/terms/> .

<>
   a oslc_cm:Bug ;
   dcterms:title "Build 23 failed" ;
   oslc_cm:severity <http://example.com/enums#S1> .
```


The content of the request depends on the type of dialog and the server. Servers may describe constraints on POST content using ResourceShapes. The dialog descriptor uses property `oslc:resourceShape` for the shape constraints.

The server's response contains a `Location` header with the prefilled dialog's URI. The client uses this as the `iframe src`.

EXAMPLE 18: The Server Responds with the Dialog Location

```
HTTP/1.1 201 Created
Location: http://example.com/dialogs/createBug/form/2zFy45
```

The client then uses the dialog URI from the `Location` like any other dialog. The dialog URI from a prefill request is often temporary. After a reasonable time, the server might respond with 404 Not Found or 410 Gone.

3.4 Security Issue: Clickjacking (UI redress attack)

This section is non-normative.

Clickjacking is a vulnerability that can occur when a malicious (or compromised) web application embeds an OSLC delegated dialog in the application in such a way that the user is tricked into using their authenticated session with the dialog, believing that they are performing an operation other than the one indicated on the delegated dialog. The clickjacking application embeds it own web page in a way that means the dialog is not clearly visible to the user, either almost invisible, or hidden behind other components). When the user attempts to click on the visible components, the browser interprets this as a click on the obscured delegated dialog, and the components within it. The malicious web page encourages the user to click one of the visible components, which the user might believe will not have side-effects (such as a link saying "If you are not redirected within 2 seconds, click here"), but places that component over a component in the delegated dialog that the malicious page wishes the user to click unknowingly. The user attempts to click on the visible component, but the browser interprets that as a click on the hidden component, which will perform some action that the user is authorised to do (but the malicious web page is not) but that the user did not intend to perform.

This vulnerability requires that the user loads a malicious web page in their browser. This can happen in a number of ways, including:

- The user is tricked into loading the page directly, believing it to be trustworthy.
- A trusted page is compromised and "infected" to perform this attack.
- An OSLC server administrator is tricked into configuring their server to trust a malicious server. Their server then loads the malicious page as a delegated dialog or rich UI preview, which then embeds the page under attack in a further iframe.

There are always two pages involved in this attack; the "UI consumer" page (which has the iframe embedded within it), and the "UI provider" page (which gets loaded within the iframe). Consider the UI provider's server the "server under attack".

To protect against this attack:

1. When requesting the URL for the delegated dialog (from the ServiceProvider) or the UI preview (from the compact resource), the server under attack must require requests for the ServiceProvider to be authenticated. This authentication must be for the end-user, not for the client app itself. That is, access to the dialog URIs should be protected, as well as access to the dialogs themselves.
2. When returning a URL for the dialog or preview, the server under attack must return a URL that is specific to the authenticated user, and that cannot be guessed by merely knowing the user's username or ID.
3. When the server under attack receives a request for the dialog or preview URL itself (the request that is generated by the iframe to load the page to display within it), the server under attack must check that the URL that was requested is one that was generated for the same user that is requesting this page (i.e. the user that is identified by a session ID in the cookies sent with the request). This makes sure that the client can't get a URL for the dialog from one user that it is authenticated for, and use that to perform a clickjacking attack on another user that has not authorized that client.
4. If the users (for the URL and the cookie) do not match, then the server can either return with an HTTP error, or include an `X-Frame-Options` header with value SAMEORIGIN or DENY.

This is only required on dialogs or resource previews that contain controls that can be invoked with one or more clicks, and where

those controls have side effects on the server. That is, if there are no controls on a preview, then there is no target for the malicious page to tick the user to click on.

By following these steps the server under attack knows that only clients that are able to authenticate against the server (e.g. for OAuth authentication they must have a valid client ID and - if required by the interaction pattern they're using - they must know the client secret), and that have also been authorized by the user to whom the the dialog is displayed, can display that dialog.

There are still some potential attack vectors:

- The server administrator could authorize a malicious client app.
- The user could authorize a malicious app to authenticate with the server under attack on their behalf.

But the exposure has been reduced as the attack cannot occur simply by loading a malicious web page in the user's browser. See the OWASP Clickjacking Defense Cheat Sheet for further information.

4. Implementation Conformance

4.1 Discovery

In responses to successful HTTP requests for an LDP container that has a selection dialog, server **MUST** include a `Link` header [RFC5988] where:

- The context URI is the effective request URI,
- The link relation is `http://open-services.net/ns/core#selectionDialog`, and
- The target URI is the URI of the selection dialog descriptor.

[dd-1]

> EXAMPLE 19
>
> ```
> Link: <http://example.com/dialogs/selectBug>; rel="http://open-services.net/ns/core#selectionDialog"
> ```

In responses to successful HTTP requests for an LDP container that has a creation dialog, server **MUST** include a `Link` header [RFC5988] where:

- The context URI is the effective request URI,
- The link relation is `http://open-services.net/ns/core#creationDialog`, and
- The target URI is the URI of the creation dialog descriptor.

[dd-2]

> EXAMPLE 20
>
> ```
> Link: <http://example.com/dialogs/createBug>; rel="http://open-services.net/ns/core#creationDialog"
> ```

Clients **MAY** request that the dialog descriptors for an LDP container are returned inline using the `Prefer` request header [RFC7240] with:

- Preference `return`, value `representation`
- Parameter `include` [LDP], value `http://open-services.net/ns/core#PreferDialog`.

[dd-3]

> EXAMPLE 21: Example Prefer Header
>
> ```
> Prefer: return=representation; include="http://open-services.net/ns/core#PreferDialog"
> ```

Servers **MUST** honor a client's request to inline dialog descriptors if the target resource has any dialog descriptors and the request is successful. [dd-4]

Servers **MUST** express the `oslc:hintWidth` and `oslc:hintHeight` properties of an `oslc:Dialog` in length units as specified in [CSS21]. [dd-5]

In responses to HTTP GET requests targeting resources that have dialogs, servers **SHOULD** either include a `Vary` response header with at least `Accept` and `Prefer` field values or a `Cache-Control` header value `no-store`. [dd-6]

Servers **MAY** also provide delegated dialog discovery using the ServiceProvider and Service resource and the `oslc:creationDialog` and `oslc:selectionDialog` properties. [dd-7]

4.2 Display

When embedding a dialog inside another web page, the client **MUST** use an `iframe` element and set the its `src` attribute to the URI of the dialog. [dd-8]

In order to allow the dialogs to be loaded on another webpage, the server **SHOULD** allow embedding dialogs on trusted domains by following the [CSP] specification. [dd-9]

4.3 Messaging

Servers **MUST** support the `Window.postMessage` method [whatwg-web-messaging] for dialog responses. [dd-10]

Servers **MAY** support the Window Name Protocol defined in [OSLCCore2] for backward compatibility with OSLC 2.0 clients that use that protocol. [dd-11]

Servers **MAY** support other protocols for dialog responses not specified in this document, which clients request by appending a fragment identifier [RFC3986] describing the protocol to the end of the dialog URI. [dd-12]

Servers **MUST** use `postMessage` for dialog responses if a client has not added a fragment identifier to the dialog URI, or the fragment identifier is `#oslc-core-postMessage-1.0`. [dd-13]

Messages describing dialog results **MUST** start with the characters `oslc-response:` followed by JSON as specified in Appendix A. Dialog Results JSON. [dd-14]

EXAMPLE 22

```
oslc-response:{"oslc:results":
  [{ "oslc:label": "bug 123: server crash", "rdf:resource": "http://example.com/bug123" }]}
```

Servers **MAY** include additional server-specific properties in the results JSON. [dd-15]

If the user cancels a dialog, the dialog **MUST** still send a message with an empty `oslc:results` array. [dd-16]

EXAMPLE 23

```
oslc-response:{"oslc:results": []}
```

Calls to `postMessage` from within a dialog **MUST** be made on `window.opener` unless `null` or `undefined`. [dd-17]

If `window.opener` is `null` or `undefined`, calls to `postMessage` **MUST** be made on `window.parent`. [dd-18]

EXAMPLE 24

```
(window.opener || window.parent).postMessage("oslc-response:" + response, "*");
```

Clients handling `message` events from dialogs **MUST** verify that the event `origin` is the same as the dialog. [dd-19]

For example, if the dialog is from example.com,

EXAMPLE 25

```
function handleMessage(event) {
    if (event.origin !== "http://example.com") {
        return;
    }

    // Otherwise, process message...
}
```


Dialogs **MAY** support dynamic resizing as specified in OSLC Resource Preview. [dd-20]

Clients **MUST** anticipate other, unrelated uses of `postMessage` and ignore unrecognized messages not conforming to the protocol. [dd-21]

4.4 Prefill

Servers **MAY** support prefill for creation and/or selection dialogs. Client provided prefill information, either in a request entity body or query parameters is intended to inform servers of possible initial values for creation dialogs, or the preferred content of selection dialogs. Clients **SHOULD NOT** assume any specific content or URL format. [dd-22]

Servers **MAY** allow clients to prefill dialogs by accepting HTTP POST requests where the Request-URI is the dialog descriptor and the entity body is an entity describing the initial values. [dd-23]

Servers that support prefill **MUST** include the value `POST` in the set of `Allow` header values returned in response to HTTP OPTIONS requests for the dialog descriptor URI. [dd-24]

> EXAMPLE 26
>
> ```
> Allow: GET,POST,HEAD,OPTIONS
> ```

Servers **MAY** describe prefill constraints by adding an `oslc:resourceShape` property to the dialog descriptor whose value is a resource shape URI. [dd-25]

Servers **MUST NOT** reject prefill requests on creation dialogs solely because they are missing required fields for the resource. Users can fill in the missing values in the dialog. [dd-26]

On successful prefill requests, servers **MUST** respond with status code 201 (Created) and a `Location` header whose value is the URI of the prefilled dialog. [dd-27]

After some elapsed time, servers **MAY** respond with a 404 (Not Found) or 410 (Gone) to an HTTP GET request for a prefilled dialog URI. [dd-28]

Servers **MAY** support prefill by adding initial values as query parameters to the delegated dialog URI. Clients **SHOULD NOT** assume any specific URL format, however. [dd-29]

5. Resource Constraints

This document applies the following constraints to the Core vocabulary terms.

An OSLC server providing Dialog capability **MUST** implement the vocabulary defined in this section. [dd-30]

- **Describes:** `http://open-services.net/ns/core#Dialog`
- **Summary:** Describes information about a dialog such as its title and dimensions.

Dialog Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:title`	Exactly-one	true	XMLLiteral	N/A	Unspecified	Title string that could be used for display.
`oslc:dialog`	Exactly-one	true	unspecified	Either	Unspecified	The URI of the dialog.
`oslc:hintHeight`	Zero-or-one	true	unspecified	Either	Unspecified	Recommended height of the dialog. Values are expressed using length units as specified in [CSS21].
`oslc:hintWidth`	Zero-or-one	true	unspecified	Either	Unspecified	Recommended width of the dialog. Values are expressed using length units as specified in [CSS21].
`oslc:label`	Zero-or-one	true	string	N/A	Unspecified	Very short label for use in menu items.
`oslc:resourceShape`	Zero-or-many	true	Resource	Reference	`oslc:ResourceShape`	Describes constraints on dialog prefill requests.
`oslc:resourceType`	Zero-or-many	true	Resource	Reference	`rdfs:Class`	The expected resource type URI for the resources that will be returned when using this dialog. These would be the URIs found in the result resource's `rdf:type` property.
`oslc:usage`	Zero-or-many	true	Resource	Reference	Unspecified	An identifier URI for the domain specified usage of this dialog. If a resource has multiple uses, it may designate the primary or default one that should be used with a property value of oslc:default.

6. Conformance

Implementations of this specification need to satisfy the following conformance clauses.

Clause Number	Requirement
dd-1	In responses to successful HTTP requests for an LDP container that has a selection dialog, server **MUST** include a `Link` header [RFC5988] where: • The context URI is the effective request URI, • The link relation is `http://open-services.net/ns/core#selectionDialog`, and • The target URI is the URI of the selection dialog descriptor.
dd-2	In responses to successful HTTP requests for an LDP container that has a creation dialog, server **MUST** include a `Link` header [RFC5988] where: • The context URI is the effective request URI, • The link relation is `http://open-services.net/ns/core#creationDialog`, and • The target URI is the URI of the creation dialog descriptor.
dd-3	Clients **MAY** request that the dialog descriptors for an LDP container are returned inline using the `Prefer` request header [RFC7240] with: • Preference `return`, value `representation` • Parameter `include` [LDP], value `http://open-services.net/ns/core#PreferDialog`.
dd-4	Servers **MUST** honor a client's request to inline dialog descriptors if the target resource has any dialog descriptors and the request is successful.
dd-5	Servers **MUST** express the `oslc:hintWidth` and `oslc:hintHeight` properties of an `oslc:Dialog` in length units as specified in [CSS21].
dd-6	In responses to HTTP GET requests targeting resources that have dialogs, servers **SHOULD** either include a `Vary` response header with at least `Accept` and `Prefer` field values or a `Cache-Control` header value `no-store`.
dd-7	Servers **MAY** also provide delegated dialog discovery using the ServiceProvider and Service resource and the `oslc:creationDialog` and `oslc:selectionDialog` properties.
dd-8	When embedding a dialog inside another web page, the client **MUST** use an `iframe` element and set the its `src` attribute to the URI of the dialog.
dd-9	In order to allow the dialogs to be loaded on another webpage, the server **SHOULD** allow embedding dialogs on trusted domains by following the [CSP] specification.
dd-10	Servers **MUST** support the `Window.postMessage` method [whatwg-web-messaging] for dialog responses.
dd-11	Servers **MAY** support the Window Name Protocol defined in [OSLCCore2] for backward compatibility with OSLC 2.0 clients that use that protocol.
dd-12	Servers **MAY** support other protocols for dialog responses not specified in this document, which clients request by appending a fragment identifier [RFC3986] describing the protocol to the end of the dialog URI.
dd-13	Servers **MUST** use `postMessage` for dialog responses if a client has not added a fragment identifier to the dialog URI, or the fragment identifier is `#oslc-core-postMessage-1.0`.
dd-14	Messages describing dialog results **MUST** start with the characters `oslc-response:` followed by JSON as specified in Appendix A. Dialog Results JSON.

Clause Number	Requirement
dd-15	Servers **MAY** include additional server-specific properties in the results JSON.
dd-16	If the user cancels a dialog, the dialog **MUST** still send a message with an empty `oslc:results` array.
dd-17	Calls to `postMessage` from within a dialog **MUST** be made on `window.opener` unless `null` or `undefined`.
dd-18	If `window.opener` is `null` or `undefined`, calls to `postMessage` **MUST** be made on `window.parent`.
dd-19	Clients handling `message` events from dialogs **MUST** verify that the event `origin` is the same as the dialog.
dd-20	Dialogs **MAY** support dynamic resizing as specified in OSLC Resource Preview.
dd-21	Clients **MUST** anticipate other, unrelated uses of `postMessage` and ignore unrecognized messages not conforming to the protocol.
dd-22	Servers **MAY** support prefill for creation and/or selection dialogs. Client provided prefill information, either in a request entity body or query parameters is intended to inform servers of possible initial values for creation dialogs, or the preferred content of selection dialogs. Clients **SHOULD NOT** assume any specific content or URL format.
dd-23	Servers **MAY** allow clients to prefill dialogs by accepting HTTP POST requests where the Request-URI is the dialog descriptor and the entity body is an entity describing the initial values.
dd-24	Servers that support prefill **MUST** include the value `POST` in the set of `Allow` header values returned in response to HTTP OPTIONS requests for the dialog descriptor URI.
dd-25	Servers **MAY** describe prefill constraints by adding an `oslc:resourceShape` property to the dialog descriptor whose value is a resource shape URI.
dd-26	Servers **MUST NOT** reject prefill requests on creation dialogs solely because they are missing required fields for the resource. Users can fill in the missing values in the dialog.
dd-27	On successful prefill requests, servers **MUST** respond with status code 201 (Created) and a `Location` header whose value is the URI of the prefilled dialog.
dd-28	After some elapsed time, servers **MAY** respond with a 404 (Not Found) or 410 (Gone) to an HTTP GET request for a prefilled dialog URI.
dd-29	Servers **MAY** support prefill by adding initial values as query parameters to the delegated dialog URI. Clients **SHOULD NOT** assume any specific URL format, however.
dd-30	An OSLC server providing Dialog capability **MUST** implement the vocabulary defined in this section.

Appendix A. Dialog Results JSON

Dialog results are represented as JSON [RFC8259]. The top-level JSON object has an `oslc:results` property.

Property	Type	Occurs	Description
`oslc:results`	JSON Array	Exactly-one	An array of results, each result a JSON object. An empty array means the user canceled the dialog or didn't select a resource.

Each result is a JSON object with the following properties.

Property	Type	Occurs	Description
`rdf:resource`	String	Exactly-one	URI of the resource selected or created
`oslc:label`	String	Zero-or-one	Short label describing the resource selected

Here is an example response that contains two resources.

EXAMPLE 27

```
{
    "oslc:results": [
        {
            "oslc:label": "Bug 123: Server crash",
            "rdf:resource": "http://example.com/bug123"
        },
        {
            "oslc:label": "Bug 456: Client hangs on startup",
            "rdf:resource": "http://example.com/bug456"
        }
    ]
}
```

When sent as a message in calls to `postMessage`, the JSON string is prefixed with the characters `oslc-response:`.

OASIS OPEN PROJECTS

OSLC Core Version 3.0. Part 5: Attachments

OASIS Standard
26 August 2021

This stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/attachments.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/attachments.pdf

Previous stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/attachments.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/attachments.pdf
(published as Project Specification)

Latest stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/attachments.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/attachments.pdf

Latest version:

https://open-services.net/spec/core/latest

Latest editor's draft:

https://open-services.net/spec/core/latest-draft

Open Project:

OASIS Open Services for Lifecycle Collaboration (OSLC) OP

Project Chairs:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Editor:

Jim Amsden (jamsden@us.ibm.com), IBM

Additional components:

This specification is one component of a Work Product that also includes:

- *OSLC Core Version 3.0. Part 1: Overview*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/oslc-core.html
- *OSLC Core Version 3.0. Part 2: Discovery*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/discovery.html
- *OSLC Core Version 3.0. Part 3: Resource Preview*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-preview.html
- *OSLC Core Version 3.0. Part 4: Delegated Dialogs*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/dialogs.html
- *OSLC Core Version 3.0. Part 5: Attachments (this document)*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/attachments.html
- *OSLC Core Version 3.0. Part 6: Resource Shape*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-shape.html
- *OSLC Core Version 3.0. Part 7: Vocabulary*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.html
- *OSLC Core Version 3.0. Part 8: Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.html
- *OSLC Core Version 3.0. Machine Readable Vocabulary Terms*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.ttl
- *OSLC Core Version 3.0. Machine Readable Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.ttl

Related work:

This specification is related to:

- *OSLC Core Version 3.0: Link Guidance*. https://oslc-op.github.io/oslc-specs/notes/link-guidance.html

RDF Namespaces:

http://open-services.net/ns/core#

Abstract:

Binary or text documents may be considered attachments to other resources. This specification describes a minimal way to manage attachments related to web resources using LDP-Containers and Non-RDF Source [LDP].

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the "Latest stage" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Open Project are listed at https://open-services.net/about/.

Comments on this work can be provided by opening issues in the project repository or by sending email to the project's public comment list oslc-op@lists.oasis-open-projects.org.

The English version of this specification is the only normative version. Non-normative translations may also be available. Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:
[OSLC-Attachments-3.0]
OSLC Core Version 3.0. Part 5: Attachments. Edited by Jim Amsden. 26 August 2021. OASIS Standard. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/attachments.html. Latest stage: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/attachments.html.

Notices

Table of Contents

1. Introduction

This section is non-normative.

Various tools handle the association and creation of related resources in conceptually similar ways, but often differ in details on how it is accomplished. The Linked Data Platform (LDP) already defines a model by which it is possible to relate resources to another, even if they are not RDF-based. This specification defines the method to create associated attachments to a given resource and understand if that resource supports the attaching of attachments.

As an example of how to create an attachment, simply HTTP POST the attachment content to the attachment container for the resource. The request should have a `Content-Type` header describing the attachment's media type. The optional `Slug` header is used to give the attachment a name.

EXAMPLE 1

```
POST /bugs/2314/attachments HTTP/1.1
Slug: design
Content-Type: application/vnd.oasis.opendocument.text
Content-Length: 18124

[binary content]
```

The response contains a Link to the new attachment in the `Location` header. This server has also included a Link to the `oslc:AttachmentDescriptor` for the attachment in the HTTP response, which contains metadata about the attachment.

EXAMPLE 2

```
HTTP/1.1 201 Created
Allow: GET,HEAD,OPTIONS,POST
Location: http://example.com/bugs/2314/attachments/3
Link: <http://example.com/bugs/2314/attachments/meta/3>; rel="describedby"; anchor="http://example.com/bugs/2314/attachments/3",
      <http://www.w3.org/ns/ldp#Resource>; rel="type"
Content-Length: 0
```

The following sections detail how to leverage LDP to accomplish the ways in which to discovery, get, create, update or delete attachments and associate with a web resource.

1.1 Terminology

Terminology uses and extends the terminology and capabilities of OSLC Core Overview, W3C Linked Data Platform [LDP], W3C's Architecture of the World Wide Web [WEBARCH], Hyper-text Transfer Protocol [HTTP11].

Attachment

A LDP-NR whose lifecycle is coupled with the attaching resource.

Attachment Container

A LDPC that contains Attachments for a resource.

Attachment Descriptor

A LDP-RS that contains additional data about an Attachment.

1.2 References

1.2.1 Normative references

[HTTP11]

R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. IETF, June 2014. Proposed Standard. URL: https://httpwg.org/specs/rfc7230.html

[LDP]

Steve Speicher; John Arwe; Ashok Malhotra. *Linked Data Platform 1.0*. W3C, 26 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/ldp/

[PURLMediaTypes]

N. Freed; M. Kucherawy; M. Baker; B. Hoehrmann. *Media Types*. IANA. URL: http://www.iana.org/assignments/media-types/media-types.xhtml

[RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF, March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

[RFC2183]

R. Troost; S. Dorner; K. Moore, Ed.. *Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field*. IETF, August 1997. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc2183

[RFC5023]

J. Gregorio, Ed.; B. de hOra, Ed.. *The Atom Publishing Protocol*. IETF, October 2007. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc5023

[RFC8174]

B. Leiba. *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*. IETF, May 2017. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc8174

[RFC8288]

M. Nottingham. *Web Linking*. IETF, October 2017. Proposed Standard. URL: https://httpwg.org/specs/rfc8288.html

[turtle]

Eric Prud'hommeaux; Gavin Carothers. *RDF 1.1 Turtle*. W3C, 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/turtle/

1.2.2 Informative references

[WEBARCH]

Ian Jacobs; Norman Walsh. *Architecture of the World Wide Web, Volume One*. W3C, 15 December 2004. W3C Recommendation. URL: https://www.w3.org/TR/webarch/

1.3 Typographical Conventions and Use of RFC Terms

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this specification are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The namespace for OSLC Core is `http://open-services.net/ns/core#`.

Sample resource representations are provided in `text/turtle` format [turtle].

Commonly used namespace prefixes:

```
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix ldp:     <http://www.w3.org/ns/ldp#>.
@prefix oslc:    <http://open-services.net/ns/core#>.
@prefix oslc_cm: <http://open-services.net/ns/cm#>.
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix wdrs:    <http://www.w3.org/2007/05/powder-s#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#>.
```


2. Motivation

This section is non-normative.

Most users of lifecycle tools have the need to easily create attachments across a variety of integrated tools and associate them to some lifecycle resource in context to some scenario. Some specific scenarios where this touches cross tool integration:

- **Running application scanning** : automatically creating a defect or task to track a problem, attaching a log file that outlines the details of the problem.
- **Publishing build results** : as part of an automated software build, publish successful build artifacts to an asset management repository
- **Mockups of app design** : share screenshots and designs to a given user story (requirement)

3. Basic Concepts

This section is non-normative.

Attachments are added to a resource via a simple POST request to the appropriate LDP-Container resource. The entity body becomes the content of the attachment resource. The attachment may automatically be associated with the resource via some membership relationship, which may use the `oslc:attachment` membership predicate. Statements are also automatically added to the `oslc:AttachmentDescriptor` resource. The property values are assigned by the server or can be determined from standard headers of the POST. The following table maps the HTTP request headers from the POST request to create the attachment resource, to what can be used to derive the initial values in the indicated `oslc:AttachmentDescriptor` resource:

HTTP Request Header	Prefixed Name
`Slug`	`dcterms:title`
`Content-Type`	`dcterms:format`
`Content-Length`	`oslc:attachmentSize`

4. Working with Attachments

This section is non-normative.

The following examples illustrate how a client can work with attachments.

4.1 Find the Attachments for a Resource

Clients get the attachments for a resource by:

1. Finding the attachment container for a resource using an HTTP OPTIONS method and Link header
2. Getting the container for the list of attachments

Each resource that supports attachments has an attachment container, which is an LDP container. Clients discover the attachment container through an HTTP Link header. A client can use GET or HEAD to get the Link header, but OPTIONS is often more efficient because the server does not have to calculate the ETag or content length of the response. LDP resources must support HTTP OPTIONS, and responses to all HTTP requests for resources that support attachments must have the Link header.

EXAMPLE 3

```
OPTIONS /bugs/2314 HTTP/1.1
Host: example.com
```

The response contains a Link to the attachment container with Link relation `http://open-services.net/ns/core#AttachmentContainer`. Note that other Link headers are in the response. In fact, LDP requires additional Link headers, which is why the response has a Link with relation `type` and target URI `http://www.w3.org/ns/ldp#Resource`.

EXAMPLE 4

```
HTTP/1.1 200 OK
Allow: GET,HEAD,OPTIONS,PUT,DELETE
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type",
      <http://example.com/bugs/2314/attachments>; rel="http://open-services.net/ns/core#AttachmentContainer"
```

Now the client requests the attachment container to see the attachments for this resource. It's a good practice to include an HTTP `Prefer` header to explicitly ask the server for the LDP containment triples.

EXAMPLE 5

```
GET /bugs/2314/attachments HTTP/1.1
Host: example.com
Accept: text/turtle
Prefer: return=representation; include="http://www.w3.org/ns/ldp#PreferContainment"
```

The response is an LDP container for the attachments. It can be any LDP container such as an `ldp:BasicContainer` or an `ldp:DirectContainer`. This example uses an `ldp:BasicContainer`. The attachment container only contains attachments for a single resource.

EXAMPLE 6

```
HTTP/1.1 200 OK
Allow: GET,HEAD,OPTIONS,POST
Content-Length: 323
Content-Type: text/turtle
ETag: W/"2773fef2237e91273bde782a43925458"
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type",
      <http://www.w3.org/ns/ldp#Container>; rel="type",
Preference-Applied: return=representation
Vary: Accept,Prefer

@prefix oslc: <http://open-services.net/ns/core#> .
@prefix ldp:   <http://w3.org/ns/ldp#> .

<http://example.com/bugs/2314/attachments>
        a             oslc:AttachmentContainer , ldp:BasicContainer ;
        ldp:contains  <http://example.com/bugs/2314/attachments/2> , <http://example.com/bugs/2314/attachments/1> .
```


Clients can look at the `ldp:contains` property on the container for the attachments.

4.2 Get the Attachment Content

Once clients have the attachment URI, they can get the attachment by simply making an HTTP GET request to the attachment URI.

A `Slug` header can be included by a server in the response to a `GET` on an attachment resource. If a client wishes to store the content as a file, this value provides a hint as to the file name to use (subject, of course, to any file system restrictions). In the absence of an `Slug` header, the client may use the last segment of the resource's URI as a hint, or just choose an arbitrary file name.

EXAMPLE 7

```
GET /bugs/2314/attachments/1 HTTP/1.1
Host: example.com
```

The response body is the attachment content. Servers should set the response `Content-Type` to describe the media type of the attachment. The response may have a `Content-Disposition` header with a filename parameter, although this isn't required. This example also contains a Link with relation `describedby`, which links to the `oslc:AttachmentDescriptor` for the attachment.

EXAMPLE 8

```
HTTP/1.1 200 OK
Allow: GET,HEAD,OPTIONS,PUT,DELETE
Content-Disposition: attachment; filename="screenshot.png"
Content-Length: 53622
Content-Type: image/png
ETag: W/"678609cdee68e0fb8aea5f252b84a511"
Link: <http://example.com/bugs/2314/attachments/meta/1>; rel="describedby",
      <http://www.w3.org/ns/ldp#Resource>; rel="type",
      <http://www.w3.org/ns/ldp#NonRDFSource>; rel="type"

[binary content]
```

4.3 Create an Attachment

To create an attachment, POST the attachment content to the attachment container for the resource. The request should have a `Content-Type` header describing the attachment's media type and subtype as specified in Media Types. The optional `Slug` header is used to give the attachment a name. The `Content-Length` header is used to initialize the attachment size.

A client can set a `Slug` header in the attachment-creating POST to specify a hint for a name for the resource as part of that single request. This can be important as some systems require a name at the time the attachment is created. Different systems may have different requirements for valid attachment names, so the value of the `Slug` header should be interpreted as a hint in this context. If the given name can not be used as specified, it is transformed into a valid name. If that is not possible or the header is not specified, an arbitrary value is assigned. Failure due to an invalid name is undesirable because of the potentially large size of an attachment resource.

The client can provide the attachment size in the `Content-Length` header and this can be used to initialize the `oslc:AttachmentDescriptor` `oslc:attachmentSize` property. The server may compute a different attachment size than that provided by the client if the client specified value is incorrect or not provided.

EXAMPLE 9

```
POST /bugs/2314/attachments HTTP/1.1
Slug: design
Content-Type: application/vnd.oasis.opendocument.text
Content-Length: 18124

[binary content]
```

The response contains a Link to the new attachment in the `Location` header. This server has also included a Link to the `oslc:AttachmentDescriptor` for the attachment in the HTTP response, which contains metadata about the attachment.

When a server successfully creates an attachment resource, it responds with an HTTP status code of 201 (created) with the URI of the newly created attachment resource in the HTTP response `Location` header. Additionally, if the server created an associated `oslc:AttachmentDescriptor` resource, the URI for this resource should be listed in the HTTP response `Link` header [RFC8288] with `rel="describedby"` [LDP].

Properties for the AttachmentDescriptor that are not readOnly, such as its title and description, can be updated using the usual HTTP `PUT` method.

EXAMPLE 10


```
HTTP/1.1 201 Created
Allow: GET,HEAD,OPTIONS,POST
Location: http://example.com/bugs/2314/attachments/3
Link: <http://example.com/bugs/2314/attachments/meta/3>; rel="describedby"; anchor="http://example.com/bugs/2314/attachments/3",
      <http://www.w3.org/ns/ldp#Resource>; rel="type"
Content-Length: 0
```

4.4 Update an Attachment

To update an attachment, PUT the attachment content to the attachment resource.

EXAMPLE 11

```
PUT /bugs/2314/attachments/3 HTTP/1.1
Content-Type: application/vnd.oasis.opendocument.text
Content-Length: 19377

[binary content]
```

The server typically responds with a 204 No Content status if the request succeeds. It also updates an associated attachment metadata in the **`oslc:AttachmentDescriptor`** in the **`describedby`** link. For example, the client could have included a **`Slug`** header on the update request in order to rename the attachment.

EXAMPLE 12

```
HTTP/1.1 204 No Content
Link: <http://example.com/bugs/2314/attachments/meta/3>; rel="describedby"; anchor="http://example.com/bugs/2314/attachments/3",
      <http://www.w3.org/ns/ldp#Resource>; rel="type"
Content-Length: 0
```

4.5 Remove an Attachment

To remove an attachment, make a DELETE request on the attachment URI. This removes the attachment from the container and deletes the content and attachment metadata from the server.

EXAMPLE 13

```
DELETE /bugs/2314/attachments/3 HTTP/1.1
Host: example.com
```

The server typically responds with 204 No Content status if the request was successful.

EXAMPLE 14

```
HTTP/1.1 204 No Content
Content-Length: 0
```

4.6 Include Attachment Information Inline with a Resource

Servers can choose to include the attachment information directly in the HTTP response for a resource although this isn't required. Here is an example defect resource that contains attachments. The attachment container is an **`ldp:DirectContainer`** where the membership resource is the defect itself. The membership predicate is **`oslc:attachment`**, although this predicate is not required. The following example shows the results of an **`HTTP GET`** on **`http://example.com/bugs/2314`**.

EXAMPLE 15

```
@prefix oslc:     <http://open-services.net/ns/core#> .
@prefix oslc_cm:  <http://open-services.net/ns/cm#> .
@prefix dcterms:  <http://purl.org/dc/terms/> .
@prefix ldp:      <http://w3.org/ns/ldp#> .
@prefix wdrs:  <http://www.w3.org/2007/05/powder-s#> .

<http://example.com/bugs/2314>
        a                   oslc_cm:Defect ;
        oslc:attachment     <http://example.com/bugs/2314/attachments/2> , <http://example.com/bugs/2314/attachments/1> ;
        dcterms:title       "A serious bug!" ;
        dcterms:identifier  "2314" .
```



```
<http://example.com/bugs/2314/attachments>
        a                       ldp:DirectContainer , oslc:AttachmentContainer ;
        ldp:contains            <http://example.com/bugs/2314/attachments/2> , <http://example.com/bugs/2314/attachments/1> ;
        ldp:hasMemberRelation   oslc:attachment ;
        ldp:membershipResource  <http://example.com/bugs/2314> .

<http://example.com/bugs/2314/attachments/1>
        wdrs:describedBy  <http://example.com/bugs/2314/attachments/meta/1> .

<http://example.com/bugs/2314/attachments/meta/1>
        a                       oslc:AttachmentDescriptor ;
        oslc:attachmentSize     "53622"^^<http://www.w3.org/2001/XMLSchema#integer> ;
        dcterms:created         "2011-07-18T13:22:30.45-05:00"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
        dcterms:creator         <http://example.com/users/steve> ;
        dcterms:format          <http://purl.org/NET/mediatypes/image/png> ;
        dcterms:identifier      "1" ;
        dcterms:title           "screenshot.png" .

<http://example.com/bugs/2314/attachments/2>
        wdrs:describedBy  <http://example.com/bugs/2314/attachments/meta/2> .

<http://example.com/bugs/2314/attachments/meta/2>
        a                       oslc:AttachmentDescriptor ;
        oslc:attachmentSize     "9196"^^<http://www.w3.org/2001/XMLSchema#int> ;
        dcterms:created         "2011-07-19T15:03:54.00-05:00"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
        dcterms:creator         <http://example.com/users/dave> ;
        dcterms:format          <http://purl.org/NET/mediatypes/text/x-diff> ;
        dcterms:identifier      "2" ;
        dcterms:title           "fix.patch" .
```


5. Implementation Conformance

5.1 General

Servers that support OSLC attachments **MUST** be Linked Data Platform 1.0 conformant servers [LDP]. [at-1]

5.2 Resources with Attachments

Each resource that supports attachments **MUST** have at least one `oslc:AttachmentContainer` that holds attachments for that resource. [at-2]

Responses to HTTP requests for resources that support attachments **MUST** contain at least one Link header [RFC8288] where the context URI is the resource URI, the Link relation is `http://open-services.net/ns/core#AttachmentContainer`, and the target URI is the URI of an `oslc:AttachmentContainer` resource. [at-3]

5.3 Attachments

An attachment **MUST** be a conformant Linked Data Platform Non-RDF Source (LDP-NR). [at-4]

Successful responses to HTTP GET requests for an attachment URI **SHOULD** include a `Content-Disposition` header [RFC2183] with disposition type `attachment` and a filename parameter. The filename is often the `Slug` header value used to create the attachment with an appropriate file extension added for the attachment's media type. [at-5]

If an attachment has an associated `oslc:AttachmentDescriptor`, responses to HTTP requests for the attachment URI **MUST** include a Link header [RFC8288] where the context URI is the attachment URI, the Link relation is `describedby`, and the target URI is the URI of the `oslc:AttachmentDescriptor`. [at-6]

When servers update an attachment, they **MUST** also update any affected `oslc:AttachmentDescriptor` properties of the associated attachment descriptor. [at-7]

When deleting attachments, servers **MUST** also delete any associated `oslc:AttachmentDescriptor` resources. [at-8]

5.4 Attachment Containers

Each `oslc:AttachmentContainer` **MUST** be a conformant Linked Data Platform Container (LDPC). [at-9]

Clients **MAY** use the HTTP `Slug` request header [RFC5023] to suggest a name when creating an attachment. If present, the `Slug` header **SHOULD NOT** include a file extension. [at-10]

Servers **SHOULD NOT** reject an HTTP POST request to an `oslc:AttachmentContainer` solely because it does not contain a `Slug` header. [at-11]

Servers **SHOULD NOT** reject an an HTTP POST request to an `oslc:AttachmentContainer` solely because they cannot use the `Slug` value unchanged. Servers **SHOULD** instead modify the `Slug` value as needed or assign a different name. [at-12]

In response to a successful HTTP POST request that creates an attachment with an associated `oslc:AttachmentDescriptor`, the server **MUST** include an HTTP Link header in the response where the context URI is the newly-created attachment URI, the link relation is `describedby`, and the link target is the `oslc:AttachmentDescriptor` URI. [at-13]

Clients **MAY** specify an LDP-NR interaction model when POSTing RDF content to an `oslc:AttachmentContainer` by including an HTTP Link header where the target URI is `http://www.w3.org/ns/ldp#NonRDFSource` and the link relation is `type`. In this case, Servers **MUST** honor the client's requested interaction model and treat the resource as an LDP-NR. [at-14]

Servers **MUST** reject an HTTP DELETE request to an `oslc:AttachmentContainer`. [at-15]

5.5 Attachment Descriptors

Servers **MAY** create an associated `oslc:AttachmentDescriptor` to describe properties of the attachment such as its name, media type, and size. [at-16]

An `oslc:AttachmentDescriptor` **MUST** have an explicit `rdf:type` set to `http://open-services.net/ns/core#AttachmentDescriptor` in its RDF representations. It **MAY** have additional `rdf:type` values. [at-17]

An `oslc:AttachmentDescriptor` **MUST** be a conforming Linked Data Platform RDF Source (LDPR). [at-18]

The `dcterms:title` of the `oslc:AttachmentDescriptor` **SHOULD** be the value of the client-supplied HTTP `Slug` header. [at-19]

Servers **SHOULD** use the `Content-Type` header value from an attachment creation request to determine the `dcterms:format` property value in the newly-created attachment's `oslc:AttachmentDescriptor`. The `dcterms:format` value **SHOULD** be a [PURLMediaTypes] media-type resource. [at-20]

An `oslc:AttachmentDescriptor` **MUST** conform to the shape defined in 6.1 Resource: AttachmentDescriptor. [at-21]

6. Resource Constraints

6.1 Resource: AttachmentDescriptor

This document applies the following constraints to the OSLCCoreVocab vocabulary terms.

An OSLC server providing the Attachments capability **MUST** implement the vocabulary defined in this section. [at-22]

The `oslc:AttachmentDescriptor` resource type is used to describe the binary resource (or non-RDF Resource) associated with a particular resource. When a client POSTs an attachment content to a server, the server stores the attachment content and assigns a URI just like any other type of resource creation but it may also create an `oslc:AttachmentDescriptor` resource to contain data about the attachment.

There is no restriction on the content of each attachment resource. For example, it could be a photo of a kitten, an installation manual, a log file, or a source code patch. Since the attachment cannot be expected to contain additional client or server supplied data, a typical set of properties for each attachment is included with the `oslc:AttachmentDescriptor` resource itself. Thus, the object of each `oslc:attachment` statement is the binary attachment. Issuing an HTTP HEAD or GET operation on that binary attachment resource URL should produce an HTTP response with a header value of `Link: rel='describedBy'` to indicate the URL of the `oslc:AttachmentDescriptor` resource. The properties for the `oslc:AttachmentDescriptor` resource are indicated in the table below.

- **Describes:** `http://open-services.net/ns/core#AttachmentDescriptor`
- **Summary:** LDP-RS to contain data about a LDP-NR(Attachment)

AttachmentDescriptor Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:created`	Zero-or-one	true	dateTime	N/A	Unspecified	Timestamp of attachment creation.
`dcterms:creator`	Zero-or-many	true	AnyResource	Either	Unspecified	Creator or creators of the attachment. Likely a `foaf:Person`, but not necessarily so.
`dcterms:description`	Zero-or-one	false	XMLLiteral	N/A	Unspecified	Descriptive text about the attachment.
`dcterms:format`	Zero-or-one	true	unspecified	Either	Unspecified	MIME type of the attachment content; **SHOULD** be a PURL media-type resource [at-23].
`dcterms:identifier`	Zero-or-one	true	string	N/A	Unspecified	System-assigned identifier.
`dcterms:title`	Zero-or-one	false	string	N/A	Unspecified	Client-specified file name or title.
`oslc:attachmentSize`	Zero-or-one	true	integer	N/A	Unspecified	Size in bytes of the attachment content.

7. Conformance

Implementations of this specification need to satisfy the following conformance clauses.

Clause Number	Requirement
at-1	Servers that support OSLC attachments **MUST** be Linked Data Platform 1.0 conformant servers [LDP].
at-2	Each resource that supports attachments **MUST** have at least one `oslc:AttachmentContainer` that holds attachments for that resource.
at-3	Responses to HTTP requests for resources that support attachments **MUST** contain at least one Link header [RFC8288] where the context URI is the resource URI, the Link relation is `http://open-services.net/ns/core#AttachmentContainer`, and the target URI is the URI of an `oslc:AttachmentContainer` resource.
at-4	An attachment **MUST** be a conformant Linked Data Platform Non-RDF Source (LDP-NR).
at-5	Successful responses to HTTP GET requests for an attachment URI **SHOULD** include a `Content-Disposition` header [RFC2183] with disposition type `attachment` and a filename parameter. The filename is often the `Slug` header value used to create the attachment with an appropriate file extension added for the attachment's media type.
at-6	If an attachment has an associated `oslc:AttachmentDescriptor`, responses to HTTP requests for the attachment URI **MUST** include a Link header [RFC8288] where the context URI is the attachment URI, the Link relation is `describedby`, and the target URI is the URI of the `oslc:AttachmentDescriptor`.
at-7	When servers update an attachment, they **MUST** also update any affected `oslc:AttachmentDescriptor` properties of the associated attachment descriptor.
at-8	When deleting attachments, servers **MUST** also delete any associated `oslc:AttachmentDescriptor` resources.
at-9	Each `oslc:AttachmentContainer` **MUST** be a conformant Linked Data Platform Container (LDPC).
at-10	Clients **MAY** use the HTTP `Slug` request header [RFC5023] to suggest a name when creating an attachment. If present, the `Slug` header **SHOULD NOT** include a file extension.
at-11	Servers **SHOULD NOT** reject an HTTP POST request to an `oslc:AttachmentContainer` solely because it does not contain a `Slug` header.
at-12	Servers **SHOULD NOT** reject an an HTTP POST request to an `oslc:AttachmentContainer` solely because they cannot use the `Slug` value unchanged. Servers **SHOULD** instead modify the `Slug` value as needed or assign a different name.
at-13	In response to a successful HTTP POST request that creates an attachment with an associated `oslc:AttachmentDescriptor`, the server **MUST** include an HTTP Link header in the response where the context URI is the newly-created attachment URI, the link relation is `describedby`, and the link target is the `oslc:AttachmentDescriptor` URI.
at-14	Clients **MAY** specify an LDP-NR interaction model when POSTing RDF content to an `oslc:AttachmentContainer` by including an HTTP Link header where the target URI is `http://www.w3.org/ns/ldp#NonRDFSource` and the link relation is `type`. In this case, Servers **MUST** honor the client's requested interaction model and treat the resource as an LDP-NR.
at-15	Servers **MUST** reject an HTTP DELETE request to an `oslc:AttachmentContainer`.
at-16	Servers **MAY** create an associated `oslc:AttachmentDescriptor` to describe properties of the attachment such as its name, media type, and size.
at-17	An `oslc:AttachmentDescriptor` **MUST** have an explicit `rdf:type` set to `http://open-services.net/ns/core#AttachmentDescriptor` in its RDF representations. It **MAY** have additional `rdf:type` values.
at-18	An `oslc:AttachmentDescriptor` **MUST** be a conforming Linked Data Platform RDF Source (LDPR).
at-19	The `dcterms:title` of the `oslc:AttachmentDescriptor` **SHOULD** be the value of the client-supplied HTTP `Slug` header.
at-20	Servers **SHOULD** use the `Content-Type` header value from an attachment creation request to determine the `dcterms:format` property value in the newly-created attachment's `oslc:AttachmentDescriptor`. The `dcterms:format` value **SHOULD** be a [PURLMediaTypes] media-type resource.
at-21	An `oslc:AttachmentDescriptor` **MUST** conform to the shape defined in 6.1 Resource: AttachmentDescriptor.
at-22	An OSLC server providing the Attachments capability **MUST** implement the vocabulary defined in this section.
at-23	MIME type of the attachment content; **SHOULD** be a PURL media-type resource

OASIS OPEN PROJECTS

OSLC Core Version 3.0. Part 6: Resource Shape

OASIS Standard
26 August 2021

This stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-shape.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-shape.pdf

Previous stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/resource-shape.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/resource-shape.pdf
(published as Project Specification)

Latest stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/resource-shape.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/resource-shape.pdf

Latest version:

https://open-services.net/spec/core/latest

Latest editor's draft:

https://open-services.net/spec/core/latest-draft

Open Project:

OASIS Open Services for Lifecycle Collaboration (OSLC) OP

Project Chairs:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Editors:

Jim Amsden (jamsden@us.ibm.com), IBM
Arthur Ryman (arthur.ryman@gmail.com), Ryerson University

Additional components:

This specification is one component of a Work Product that also includes:

- *OSLC Core Version 3.0. Part 1: Overview.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/oslc-core.html
- *OSLC Core Version 3.0. Part 2: Discovery*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/discovery.html
- *OSLC Core Version 3.0. Part 3: Resource Preview.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-preview.html

- *OSLC Core Version 3.0. Part 4: Delegated Dialogs*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/dialogs.html
- *OSLC Core Version 3.0. Part 5: Attachments*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/attachments.html
- *OSLC Core Version 3.0. Part 6: Resource Shape (this document)*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-shape.html
- *OSLC Core Version 3.0. Part 7: Vocabulary*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.html
- *OSLC Core Version 3.0. Part 8: Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.html
- *OSLC Core Version 3.0. Machine Readable Vocabulary Terms*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.ttl
- *OSLC Core Version 3.0. Machine Readable Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.ttl

Related work:

This specification is related to:

- *OSLC Core Version 3.0: Link Guidance*. https://oslc-op.github.io/oslc-specs/notes/link-guidance.html

RDF Namespaces:

http://open-services.net/ns/core#

Abstract:

This document defines a high-level RDF vocabulary for specifying the *shape* of RDF resources. The shape of an RDF resource is a description of the set of triples it is expected to contain and the integrity constraints those triples are required to satisfy. Applications of shapes include validating RDF data, documenting RDF APIs, and providing metadata to tools, such as form and query builders, that handle RDF data.

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the “Latest stage” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Open Project are listed at https://open-services.net/about/.

Comments on this work can be provided by opening issues in the project repository or by sending email to the project’s public comment list oslc-op@lists.oasis-open-projects.org.

The English version of this specification is the only normative version. Non-normative translations may also be available. Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:
[OSLC-Shapes-3.0]
OSLC Core Version 3.0. Part 6: Resource Shape. Edited by Jim Amsden and Arthur Ryman. 26 August 2021. OASIS Standard. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-shape.html. Latest stage: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/resource-shape.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

Table of Contents

1. Introduction

This section is non-normative.

Linked Data has proved to be an effective way to look up and navigate information at web scale. It is also emerging as a compelling architectural basis for web applications [RWLD]. The principles for designing Linked Data web applications are standardized by the W3C Linked Data Platform Specification 1.0 [LDP].

RDF is at the core of Linked Data. The design principles for Linked Data [LDDI] include the following rule:

> When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)

This emergence of RDF as an important application development technology for Linked Data and other domains has brought to light a serious gap in the associated technology stack, namely the absence of a way to describe the integrity constraints imposed by an application on the RDF documents it processes. Here the term "integrity constraint" denotes any characteristic of data required by or enforced by an application. Common examples of integrity constraints include the mandatory presence of properties, the cardinality of relations, and restrictions on the allowed values of properties.

Well-established programming technologies such as relational databases, object-oriented programming languages, and XML utilize closed-world assumptions to provide schemas or type definitions that encode integrity constraints. Applications utilizing RDF and open-world assumptions may need a similar mechanism to constrain the open-world for a particular context, and frequently start to use RDFS or OWL for this purpose. However, this is a misapplication of those technologies.

RDFS and/or OWL are used to define OSLC vocabularies which specify the terms (classes and properties) and their semantic meaning. Vocabularies don't define constraints on what can be asserted, rather they define the meaning of what was asserted, often through the use of reasoners that introduce inferred assertions. However, there are also situations where a vocabulary needs to be constrained in order to be used in a specific context for a specific purpose. RDFS and OWL are often not useful for this purpose as unless the vocabularies are carefully designed, reasoners can introduce unintended assertions that are not consistent with the specified purpose.

This document describes the Resource Shape specification, a high-level RDF vocabulary for describing the *shape* of RDF resources. Here the term "RDF resource" is used to denote an LDPR resource that has an RDF document among its set of available representations. The term "shape" is used because it is often useful to visualize an RDF document as a topological object, namely as a graph consisting of nodes interconnected by arcs. Throughout this document the terms "RDF resource", "RDF document", and "RDF graph" are used more or less interchangeably albeit somewhat imprecisely.

The shape of an RDF graph includes both a description of its expected contents (properties, types) within some operational context (e.g. GET, POST) and the integrity constraints that must be satisfied by the contents in that context.

Resource shapes specify a standard way of describing resources and constraints on resources that may be used in defining among other things, OSLC domain resources. There are other means that OSLC domains may use in addition to or instead of resource shapes such as W3C [SHACL] or [JSONSchema]. OSLC servers should describe their resources using resource shapes if they wish to integrate with OSLC Core 3.0 clients or servers that are expecting resource shapes. OSLC domain specifications use shapes to specify the constraints on their domain vocabulary elements that must be supported by servers.

This specification begins with a brief discussion of the use cases and requirements that the OSLC Resource Shape Specification addresses. It then describes all aspects of the current specification in detail. Finally, it concludes with some recommendations for extensions based on implementation experience.

1.1 Terminology

Terminology uses and extends the terminology and capabilities of OSLC Core Overview, W3C Linked Data Platform [LDP], W3C's Architecture of the World Wide Web [WEBARCH], Hyper-text Transfer Protocol [HTTP11].

This specification introduces the following terminology:

applicability

Shapes associated with a resource apply to that resource if the shape's `oslc:describes` property matches the type of the associated resource, or the shape has no `oslc:describes` property in which case it applies to all resources associated with that shape. For all shapes that "apply to" or describe an associated resource, that resource should satisfy the constraints defined by those shapes. See Associating and Applying Shapes for the definition of how shapes are associated with resources, what associated shapes apply to a resource, and what it means if multiple shapes apply to a single resource.

association

This specification defines three contexts in which a shape may become associated with a described resource. The described resource itself may link to a shape using the property oslc:instanceShape, the described resource may be the entity request to or response from a REST service whose service description links to a shape using the property oslc:resourceShape, or the resource may be the object value of an rdf:Property whose applicable oslc:valueShape links to the constraining shape. More than one shape may become associated with a described resource in a given context. The shapes associated with a resource are the set of shapes to be checked for applicability. Not all the associated shapes are necessarily applicable. Refer to the definition of applicability above.

datatype property

A datatype property is a defined property that takes literal values.

defined property

A defined property is a property, such as dcterm:description or oslc_cm:status, of a described resource that is defined by some shape applicable to the described resource in a given context. A shape is linked to its defined properties using the property oslc:property. A described resource type might have a given defined property in some contexts but not in others. For example, the body of a POST request might have fewer defined properties than the created resource.

described resource

A described resource is a resource described by some shape.

document

A representation of a resource. For example, an RDF document may represent a resource.

graph

An RDF document may be regarded as a directed, labeled multi-graph, or, simply, a graph, in which each triple of the document corresponds to an arc of the graph. For each arc in the graph, the source is the subject of its triple, the target is the object of its triple, and the label is the predicate of its triple.

object property

An object property is a defined property that links to a resource, referred to as the object resource.

object resource

The object resource is the resource that is linked to by an object property.

shape

A resource of type oslc:ResourceShape that describes the contents of and constraints on some set of described resources.

1.2 References

1.2.1 Normative references

[HTTP11]

R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. IETF, June 2014. Proposed Standard. URL: https://httpwg.org/specs/rfc7230.html

[LDP]

Steve Speicher; John Arwe; Ashok Malhotra. *Linked Data Platform 1.0*. W3C, 26 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/ldp/

[RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF, March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

[RFC8174]

B. Leiba. *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*. IETF, May 2017. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc8174

[rdf11-concepts]

Richard Cyganiak; David Wood; Markus Lanthaler. *RDF 1.1 Concepts and Abstract Syntax*. W3C, 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/rdf11-concepts/

[xmlschema11-1]

Sandy Gao; Michael Sperberg-McQueen; Henry Thompson; Noah Mendelsohn; David Beech; Murray Maloney. *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. W3C, 5 April 2012. W3C Recommendation. URL: https://www.w3.org/TR/xmlschema11-1/

1.2.2 Informative references

[JSONSchema]

JSON Schema. http://json-schema.org/. Draft. URL: http://json-schema.org/

[LDDI]

Tim Berners-Lee. *Linked Data Design Issues*. W3C. URL: http://www.w3.org/DesignIssues/LinkedData.html

[LDOW2013]

Arthur Ryman; Arnaud Le Hors; Steve Speicher. *OSLC Resource Shape: A language for defining constraints on Linked Data*. URL: http://events.linkeddata.org/ldow2013/papers/ldow2013-paper-02.pdf

[RDFVAL]

W3C RDF Validation Workshop. W3C. URL: http://www.w3.org/2012/12/rdf-val/

[REST]

Roy Thomas Fielding. *Representational State Transfer (REST)*. University of California. URL: http://www.ics.uci.edu/%7Efielding/pubs/dissertation/rest_arch_style.htm

[RWLD]

Tim Berners-Lee. *Read-Write Linked Data*. W3C. URL: http://www.w3.org/DesignIssues/ReadWriteLinkedData.html

[SHACL]

Holger Knublauch; Arthur Ryman. *Shapes Constraint Language (SHACL)*. http://www.w3.org/. Draft. URL:

https://w3c.github.io/data-shapes/shacl/

[ShapesRDFVAL]

Achille Fokoue; Arthur Ryman. *OSLC Resource Shape: A Linked Data Constraint Language*. W3C. URL: http://www.w3.org/2001/sw/wiki/images/b/b7/RDFVal_Fokuoe_Ryman.pdf

[TURTLE]

Tim Berners-Lee; David Beckett; Eric Prud'hommeaux; Gavin Carothers. *Turtle - Terse RDF Triple Language*. W3C. URL: http://www.w3.org/TR/turtle/

[WEBARCH]

Ian Jacobs; Norman Walsh. *Architecture of the World Wide Web, Volume One*. W3C, 15 December 2004. W3C Recommendation. URL: https://www.w3.org/TR/webarch/

1.3 Typographical Conventions and Use of RFC Terms

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this specification are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Sample resource representations are provided in `text/turtle` format [TURTLE].

The following common URI prefixes are used throughout this specification:

```
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix ex:      <http://example.org/>.
@prefix ext:     <http://example.org/extension#>.
@prefix ldp:     <http://www.w3.org/ns/ldp#>.
@prefix oslc:    <http://open-services.net/ns/core#>.
@prefix oslc_cm: <http://open-services.net/ns/cm#>.
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#>.
```

Fig. 1 Commonly used URI prefixes.

2. Motivation

This section is non-normative.

This section briefly describes the main use cases that motivate an RDF graph shape language. For a more complete discussion of this topic, refer to the W3C RDF Validation Workshop [RDFVAL].

2.1 Describing RDF APIs

This section is non-normative.

Any application that provides programmatic services should also provide application programming interface (API) documentation to programmers so that they can consume those services. Applications that process RDF, including Linked Data web applications, are no different than other applications in this respect.

Although API documentation is normally directed towards human programmers, there are also important use cases where other programs need to understand the API. For example, consider a generic form-building tool that can generate a user interface for creating or updating resources. Such a tool needs to understand the expected contents of the resource so it can generate a user interface. It also needs to understand the applicable integrity constraints so it can validate user input before sending it to the service provider or server. Furthermore, the server may apply different constraints for creation versus update operations.

2.2 Describing RDF Datasets

This section is non-normative.

Users must understand the contents of an RDF dataset in order to write SPARQL queries. Understanding the integrity constraints can help users write better SPARQL queries. For example, if a property is known to always be present, then there is no need to wrap it in an **OPTIONAL** clause.

Query building tools can take advantage of shape information. For example, consider a query builder that allows the user to define a query that filters results by selecting allowed values from a list. In principle, the query builder could dynamically query the dataset to determine the list of values present. However, such a query could be slow if the dataset was large. In addition, only those values present at the time of the query would appear in the list. In contrast, if the query builder had access to shape information, then it could avoid the potentially expensive query and present the user with the complete list of allowed values, whether or not they were actually present in the dataset at that time.

2.3 Describing Indexable Data

This section is non-normative.

Resource indexers may be looking for certain types of structured data. For example, a web crawler might be indexing product descriptions and pricing for a marketplace application. The web crawler could provide a shape to describe the data it is looking for. Web application developers would then be able to provide that information in the web pages of their commerce site and thereby have their sites included in the index.

3. Requirements

This section is non-normative.

This section describes some high-level requirements for an RDF resource shape language. For simplicity, members of a shape language are referred to as shape resources. It is to be understood that the validity and meaning of a shape resource depends on any other shape resources it is linked to.

1. A shape language must be able to express the most commonly occurring aspects of RDF resources in high-level terms, using familiar concepts. Interested parties must be able to both understand existing shapes and author new ones.
2. A shape resource must be processable using readily available RDF technologies. These include RDF parsers and SPARQL processors.
3. The constraints defined by a shape resource must be verifiable using commonly available technologies, with acceptable performance. The performance criteria depend on the use case. For example, a tool checking user input must respond quickly. A tool validating an entire dataset may run as a batch job, but should complete within a reasonable amount of time (e.g. hours, not years).
4. A shape language must be extensible so that application-specific aspects of resources can be expressed. A trade-off of expressiveness at the expense of ease-of-understanding is acceptable.
5. It must be possible to apply different shapes to the same resources in different contexts in order to use the resources for different purposes.

4. Shape Resources

This section describes the Resource Shape specification (aka the *Shapes* specification) which is part of OSLC Core 3.0. The Shapes specification defines:

- the contents and meaning of shape resources, and
- how to associate shape resources with resources and services

4.1 Overview

This section is non-normative.

A resource shape is a resource that describes the contents of, and constraints on, the RDF representation of other resources. These constraints are intended to be applicable to OSLC core and domain vocabularies in order to express the domains sufficiently to support interoperability between clients and servers that use and support them. Resource shapes specify the minimum an implementation needs to do be considered compliant. Specifically, shape constraints say how OSLC clients and servers must behave if the resources satisfy the applicable shape constraints, but they do not say what clients and servers may do if the applicable constraints are not satisfied.

A shape resource itself has an RDF representation which uses the terms defined by the oslc: vocabulary. The term "shape resource" or simply "shape" is sometimes used as shorthand for the more verbose phrase "the RDF representation of a shape resource" where this can lead to no confusion. The following sections describe all of the RDF vocabulary terms used in shape resources.

The Shapes specification is based on a simple conceptual model of resources that works well in practice, but is somewhat biased towards the view that the RDF representation of a resource looks like a set of property-value pairs on that resource. The Shapes specification works well when the resource being described appears as a subject node in its RDF graph and all other nodes are connected to the resource node by a path consisting of one or more properties. Each property-value pair is represented by a triple in which the subject is the resource, the predicate is the property, and the object is the value. The value may be either a literal or a resource. When the object is a resource, that resource may itself be described by another shape. Thus the Shapes specification is powerful enough to describe complex graphs. Although the Shapes specification works well in practice, it cannot describe arbitrary RDF graphs.

The following diagram summarizes the main concepts and relations used in this specification:

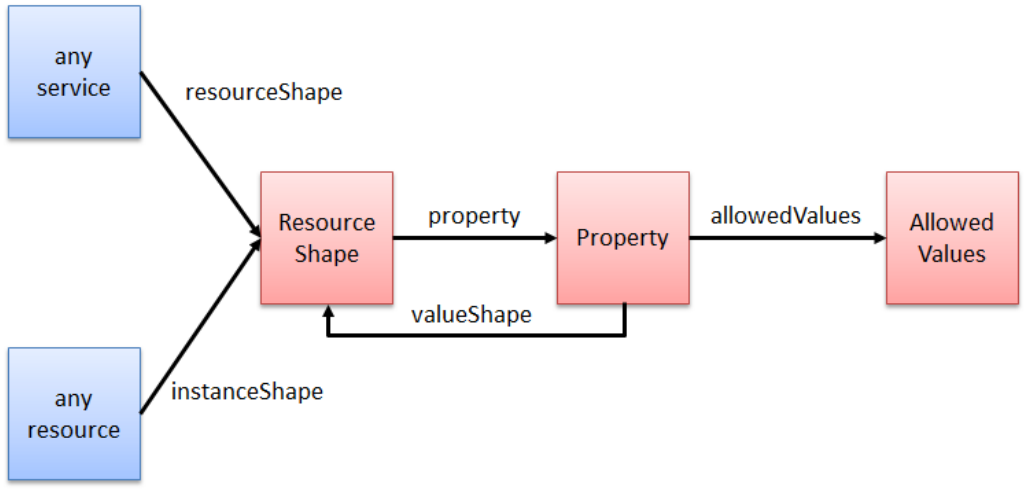

Fig. 2 Diagram of main concepts and relations.

In this diagram the boxes represent resource types and the arrows represent the relations between them that are defined by the Shapes specification. The two boxes on the left represent external types of resources that use shapes. The other three

boxes represent the resource types that are defined by the Shapes specification.

The box labeled "Resource Shape" represents a shape. A shape is a resource of `rdf:type` oslc:ResourceShape. A shape describes a set of resources. A shape is basically a set of one or more defined properties that any resource described by that shape is expected to contain.

The box labeled "Property" represents a defined property. A defined property is a resource of rdf:type oslc:Property. The arrow labeled "property" represents the aggregation relation between a shape and its defined properties. The predicate of this relation is oslc:property.

Each oslc:Property resource has a set of properties that describe the defined property and the constraints on its use within any resource that the given shape applies to. These include a description of the values of the defined property and the number of its occurrences or cardinality within the resource. The value of a defined property may be a literal, a resource, or either. If the value of a defined property is a resource, then defined property may refer to another oslc:ResourceShape resource that describes the value resource. This relation is depicted by the arrow labeled "valueShape". The predicate of this relation is oslc:valueShape.

The value of a defined property may be constrained to take one of an allowed set of values. In some cases, the allowed set of values may be large and be used in many shapes. In this case it is useful to put the allowed values in a separate resource so they can be easily reused. The box labeled "Allowed Values" represents a resource of rdf:type oslc:AllowedValues. The arrow labeled "allowedValues" represents the relation between a defined property and its set of allowed values. The predicate of this relation is oslc:allowedValues.

A [REST] service may describe aspects of its interface contract using shapes. For example, a REST service may provide a URI where new resources can be created via HTTP POST. This service could describe the expected contents of POST request bodies using shapes. Similarly, a REST service may provide a URI that represents a container of resources and could describe those resources using shapes. The box labeled "any service" represents any REST service description. The arrow labeled "resourceShape" represents the predicate oslc:resourceShape which is a property of the service description resource.

Similarly, any resource can describe its own contents by linking to a shape resource. The box labeled "any resource" represents any resource. The arrow labeled "instanceShape" represents the predicate oslc:instanceShape which is a property of the resource.

4.2 Associating and Applying Shapes

In general, the relation between shapes and resources is many-to-many. Given a resource R there **MAY** be zero or more shapes S associated with it. This specification defines three ways to associate shapes with a resource, namely using oslc:instanceShape, oslc:resourceShape, and oslc:valueShape. Other specifications **MAY** define additional mechanisms. [rs-1]

1. Resource `oslc:instanceShape` ResourceShape - directly associates a constraining shape with a resource.
2. Service `oslc:resourceShape` ResourceShape - associates a constraining shape with the entity request or response resource of a service (e.g., a creation or query service).
3. Property `oslc:valueShape` ResourceShape - associates a constraining shape with the resource that is the object value of a property of a resource.

Not all shapes associated with a resource are necessarily applicable to it. Let S be associated with R. S is said to *apply* to R in the following two cases:

Case 1: Generic Shape

S applies to R when S has no oslc:describes properties, in which case it is a generic shape and applies to any associated resource.

Case 2: Typed Shape

S applies to R when S is linked via oslc:describes to an rdf:type T and R has rdf:type T.

If no shapes are associated with a resource then there are no implied constraints on that resource.

If one or more shapes are associated with a resource then at least one of those **SHOULD** be applicable to that resource. If no associated shape applies to a resource then this **SHOULD** be interpreted as an error condition. [rs-2]

If exactly one shape applies to a resource then that resource **SHOULD** satisfy all the constraints defined by that shape. [rs-3]

If more than one shape applies to a resource then that resource **SHOULD** satisfy all the constraints defined by all the shapes (AND semantics). However, the specification for a service description **MAY** define alternate semantics. For example, a service **MAY** require that the resource satisfy the constraints defined by at least one of the shapes (OR semantics). [rs-4]

If a resource satisfies its applicable shapes, client and server implementations **MUST** behave as described in the defining OSLC specifications. If a resource does not satisfy its applicable shapes, implementations **SHOULD** attempt to complete the operation with the given data if possible. Otherwise implementations **MAY** reject the operation. [rs-5]

4.3 A Running Example of Shape Resources

This section is non-normative.

This section presents a simple running example to illustrate the Shapes specification. For more examples, refer to [LDOW2013], and [ShapesRDFVAL].

Consider a simple bug tracking service in which each bug has just two properties: a summary and status. The summary is required, but the status is optional. The summary is given by the property dcterms:title, and the status by oslc_cm:status. The oslc_cm:status is constrained to take one of the values "Submitted", "InProgress", or "Done". The RDF type of a bug is oslc_cm:ChangeRequest.

4.3.1 Example of a Valid Bug

This section is non-normative.

The following listing shows the RDF representation of the bug `http://example.com/bugs/1` which satisfies the constraints defined by the bug tracking service:

EXAMPLE 1

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix oslc: <http://open-services.net/ns/core#> .
@prefix oslc_cm: <http://open-services.net/ns/cm#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<http://example.com/bugs/1> a oslc_cm:ChangeRequest ;
    dcterms:title "Null pointer exception in web ui"^^rdf:XMLLiteral ;
    oslc_cm:status "Submitted" ;
    oslc:instanceShape <http://example.com/shape/oslc-change-request> .
```

4.3.2 Example of an Invalid Bug

This section is non-normative.

The following listing shows the RDF representation of the bug `http://example.com/bugs/2` which violates the constraints since its oslc_cm:status property has two values:

EXAMPLE 2

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix oslc: <http://open-services.net/ns/core#> .
@prefix oslc_cm: <http://open-services.net/ns/cm#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<http://example.com/bugs/2> a oslc_cm:ChangeRequest ;
```



```
    dcterms:title "Wrong arguments"^^rdf:XMLLiteral ;
    oslc_cm:status "Submitted", "InProgress" ;
    oslc:instanceShape <http://example.com/shape/oslc-change-request> .
```

4.3.3 Example of a Shape for Bugs

This section is non-normative.

We can represent the constraints defined by the bug tracking service using the shape resource **`http://example.com/shape/oslc-change-request`** which has the following RDF representation:

EXAMPLE 3

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix oslc: <http://open-services.net/ns/core#> .
@prefix oslc_cm: <http://open-services.net/ns/cm#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@base <http://example.com/shape/> .

<oslc-change-request> a oslc:ResourceShape ;
    dcterms:title "Creation shape of OSLC Change Request"^^rdf:XMLLiteral ;
    oslc:describes oslc_cm:ChangeRequest ;
    oslc:property
        <oslc-change-request#dcterms-title> ,
        <oslc-change-request#oslc_cm-status> .

<oslc-change-request#dcterms-title> a oslc:Property ;
    oslc:propertyDefinition dcterms:title ;
    oslc:name "title" ;
    oslc:occurs oslc:Exactly-one .

<oslc-change-request#oslc_cm-status> a oslc:Property ;
    oslc:propertyDefinition oslc_cm:status ;
    oslc:name "status" ;
    oslc:occurs oslc:Zero-or-one ;
    oslc:allowedValues <status-allowed-values> .
```

The RDF representation contains one oslc:ResourceShape resource and two oslc:Property resources. The oslc:ResourceShape resource contains a short description of the shape using dcterms:title, gives the RDF type of the resource being described using oslc:describes, and lists the two properties contained in the resource being described using oslc:property.

The contained properties, dcterms:title and oslc_cm:status are described in oslc:Property resources. Each oslc:Property resource specifies the URI of the RDF property it is defining using oslc:propertyDefinition, and the occurrence of that property using oslc:occurs. The occurrence of each property is specified using the terms oslc:Exactly-one and oslc:Zero-or-one which indicate that the properties are both single-valued. dcterms:title is required and oslc_cm:status is optional.

In this example, only the occurrence and allowed values constraints are used. The Shapes specification provides for properties to be constrained by value type, range, and several other aspects.

4.3.4 Example of Allowed Values for Status

This section is non-normative.

The definition of oslc_cm:status refers to **`http://example.com/shape/status-allowed-values`** using oslc:allowedValues. That resource has the rdf:type oslc:AllowedValues. It specifies the allowed values of the oslc_cm:status property. It has the following RDF representation:

EXAMPLE 4


```
@prefix oslc: <http://open-services.net/ns/core#> .

<http://example.com/shape/status-allowed-values> a oslc:AllowedValues ;
    oslc:allowedValue "Done" , "InProgress" , "Submitted" .
```


5. Constraints

Property tables are used below to describe the resources defined by the Shapes specification, namely oslc:ResourceShape, oslc:Property, and oslc:AllowedValues. A property table is a tabular depiction of a subset of the information that can be specified using shapes. Each table describes one type of resource. Each row describes one property of the resource. Each column describes some aspect of the properties. The columns have the following meanings:

Prefixed Name

The compact IRI [IRI] of the defined property being described in the current row of the table. This corresponds to the oslc:propertyDefinition property.

Occurs

The number of times the defined property may occur. This corresponds to the oslc:occurs property.

Read-Only

`true` if the property is read-only. If omitted, or set to false, then the property is writable.

Value-Type

The type of the values of the defined property. This corresponds to the oslc:valueType property. A defined property that takes literal values is called a *datatype property*. A defined property that links to another resource is called an *object property*. The resource linked to is called the *object resource*. For datatype properties, this property gives the datatype URI of the literal values. For object properties, this property specifies whether the object resource has a URI (Resource), is a blank node (LocalResource), or is any resource (AnyResource). These values correspond to oslc:Resource, oslc:LocalResource, and oslc:AnyResource.

Representation

For object properties, the location of the representation of the object resource. This corresponds to the oslc:representation property. The representation of the object resource may be contained in the same document as the described resource (Inline), or it may be contained in a remote document (Reference), or it may be either (Either). These values correspond to oslc:Inline, oslc:Reference, and oslc:Either. Domain vocabulary designers should carefully consider restrictions on representations, such as requiring oslc:Referenceinstead of oslc:Either, in order to provide maximum flexibility.

Range

For object properties, the allowed rdf:type(s) of the object resources. This corresponds to the oslc:range property. This value may be Any if any type is allowed. The value Any corresponds to oslc:Any.

Description

A description of the defined property. This corresponds to the dcterms:title property.

oslc:instanceShape Property

The `oslc:instanceShape` property is used to link any described resource with a shape resource that describes its contents. A resource **MAY** be associated with zero or more shapes. [rs-6]

oslc:resourceShape Property

The `oslc:resourceShape` property is used to link an application service description with a shape resource that describes some aspect of the service's API contract. A service description **MAY** be linked with zero or more shapes. [rs-7]

For example, in OSLC a resource that accepts POST requests to and LDPC in order to create new resources is referred to as a *creation factory*. The service description for a creation factory may link to one or more shape resources that describe the bodies of POST requests.

5.1 ResourceShape Constraints

- **Describes:** `http://open-services.net/ns/core#ResourceShape`
- **Summary:** A shape resource describes the contents of and constraints on some set of described resources.
- **Description:** A resource should satisfy all the constraints defined by its applicable shapes.

ResourceShape Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:description`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	The description of the defined constraint.
`dcterms:title`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	The summary of this shape.
`oslc:describes`	Zero-or-many	true	Resource	Reference	`rdfs:Class`	The described resource types that this shape applies to.
`oslc:hidden`	Zero-or-one	true	boolean	N/A	Unspecified	Indicates the resource or property should not be displayed to users.
`oslc:property`	Zero-or-many	true	Resource	Inline	`oslc:Property`	Indicates an expected property of the described resources.
`rdf:type`	Zero-or-many	true	Resource	Reference	Unspecified	An OSLC resource shape **SHOULD** have an RDF type of oslc:ResourceShape.

dcterms:description Property

dcterms:title Property

`dcterms:title` is used to provide a summary of oslc:ResourceShape and oslc:Property resources. Its value **SHOULD** be a literal of type rdf:XMLLiteral that is valid content for an XHTML `<span>` element. If the value contains no XML markup then it **MAY** be represented as a plain text literal or xsd:string. [rs-8]

`dcterms:description` is used to provide a description of oslc:Property resources. Its value **SHOULD** be a literal of type rdf:XMLLiteral that is valid content for an XHTML `<div>` element. If the value contains no XML markup then it **MAY** be represented as a plain text literal or xsd:string. [rs-9]

oslc:describes Property

`oslc:describes` is used to list the types of the described resources associated with this shape. Suppose that shape S is associated with described resource R, e.g. via an oslc:resourceShape or oslc:instanceShape link. If shape S describes type T and described resource R has type T then S describes the contents of and constraints on R.

For example, a creation factory may be able to create many different types of resources. The constraints on a given type of resource is specified by the associated shape resources that contain an oslc:describes link to that type.

oslc:property Property

`oslc:property` is used to list the defined properties that are expected to be contained in described resources associated with this shape. The object of this property **MUST** be an oslc:Property resource whose representation is contained in the shape document. [rs-10]

If a described resource contains a property described by some oslc:Property resource, then a REST service is expected to process that property in some useful way as defined by the service's API contract. If there is no matching oslc:Property resource then the behavior of the service is undefined.

5.2 Property Constraints

- **Describes:** `http://open-services.net/ns/core#Property`
- **Summary:** Specifies the name, description, summary, occurrence, value type, allowed values, and several other aspects of the defined property.

Property Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:description`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	The description of the defined constraint.
`dcterms:title`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	The summary of the defined property.
`oslc:allowedValue`	Zero-or-many	true	unspecified	Either	Unspecified	Specifies the allowed values of a property.
`oslc:allowedValues`	Zero-or-one	true	Resource	Reference	`oslc:AllowedValues`	The resource containing a set of allowed values of the defined property.
`oslc:defaultValue`	Zero-or-one	true	unspecified	Either	Unspecified	The default value of the defined property.
`oslc:hidden`	Zero-or-one	true	boolean	N/A	Unspecified	Indicates the resource or property should not be displayed to users.
`oslc:isMemberProperty`	Zero-or-one	true	boolean	N/A	Unspecified	If true then the described resource is a container and the defined property is used for container membership.

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:maxSize	Zero-or-one	true	integer	N/A	Unspecified	For string datatype properties, the maximum number of characters.
oslc:name	Exactly-one	true	string	N/A	Unspecified	The local name of the defined property.
oslc:occurs	Exactly-one	true	Resource	Reference	oslc:Cardinality	The number of times the defined property may occur.
oslc:propertyDefinition	Exactly-one	true	Resource	Reference	rdf:Property	The URI of the defined or constrained property.
oslc:queryable	Zero-or-one	true	boolean	N/A	Unspecified	Indicates whether a property is queryable (can appear in oslc.where and olsc.select clause) or not.
oslc:range	One-or-many	true	Resource	Reference	rdfs:Class	For object properties, specifies what the target resource type is expected to be, but that is not necessarily the case.
oslc:readOnly	Zero-or-one	true	boolean	N/A	Unspecified	If true then the defined property cannot be directly writen by clients, but may be updated indirectly by servers.

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:representation	Zero-or-one	true	Resource	Reference	oslc:Representation	For object properties, how the object resource is represented in the representation of the described resource.
oslc:valueShape	Zero-or-one	true	Resource	Reference	oslc:ResourceShape	For object properties, the URI of a shape resource that describes the object resource.
oslc:valueType	Zero-or-many	true	Resource	Reference	oslc:ResourceValueType	The type of values of the defined property.
rdf:type	Zero-or-many	true	Resource	Reference	Unspecified	An OSLC property SHOULD have an RDF type of oslc:Property.

oslc:allowedValue Property

oslc:allowedValue is used to specify an allowed value of the defined property. The object of this property SHOULD be compatible with the type specified by the oslc:valueType property if present. An oslc:Property resource MAY contain one or more oslc:allowedValue properties and an optional oslc:allowedValues property which links to an oslc:AllowedValues resource. The complete set of allowed values is the union of all the values specified directly in the oslc:Property resource and the linked oslc:AllowedValues resource. [rs-11]

oslc:allowedValues Property

oslc:allowedValues specifies a link to an oslc:AllowedValues resource which defines a set of allowed values for the defined property. See oslc:allowedValue for a description of how the complete set of allowed values is defined.

oslc:defaultValue Property

oslc:defaultValue specifies the default value for the defined property. The object of this property SHOULD be compatible with the type specified by the oslc:valueType property if present. [rs-12]

A default value is normally used when creating resources. A service SHOULD use the default value to provide a value for a property if none is provided in the creation request. [rs-13]

A default value MAY be used by clients of a described resource if the defined property is not present in the representation of the described resource. This mechanism is useful if a service introduces a new defined property but does not update all pre-existing described resources. [rs-14]

oslc:hidden Property

If present and true, `oslc:hidden` is used to indicate that the defined property is not normally presented to users. A client of the described resource **SHOULD NOT** display hidden defined properties to normal users. It **MAY** display hidden defined properties to administrative users. [rs-15]

oslc:isMemberProperty Property

If the `oslc:isMemberProperty` is present and true then the defined property is a container membership property similar to rdfs:member. The described resource is the container and the object resources are its members.

For example, OSLC Query Capabilities are query services that behave like containers for other resources. A defined property for which oslc:isMemberProperty is true links the container to its member resources.

The recent Linked Data Platform specification [LDP] elaborates the concept of resource container. It is therefore desirable to evolve the Shapes specification to align with the LDP concept of container membership.

oslc:name Property

`oslc:name` is used to specify the local name of the defined property. This is normally the portion of the defined property URI (see oslc:propertyDefinition) that follows the last hash (#) or slash (/).

oslc:maxSize Property

For datatype properties whose type is `xsd:string`, `oslc:maxSize` specifies the maximum number of characters in the defined property value. The absence of oslc:maxSize indicates that either there is no maximum size or that the maximum size is specified some other way.

oslc:occurs Property

`oslc:occurs` is used to specify the number of times that the defined property may occur. The value of this property **MUST** be one of the following individuals:

oslc:Exactly-one
: The defined property **MUST** occur exactly once. It is required and single-valued.

oslc:One-or-many
: The defined property **MUST** occur at least once. It is required and multi-valued.

oslc:Zero-or-many
: The defined property **MAY** occur any number of times. It is optional and multi-valued.

oslc:Zero-or-one
: The defined property **MUST** occur no more than once. It is optional and single-valued.

For strings and language-tagged strings, single-valued means there is at most one value for any given language tag, and at most one untagged value.

[rs-16]

oslc:propertyDefinition Property

`oslc:propertyDefinition` is used to specify the URI of the the defined property.

oslc:range Property

`oslc:range` **MUST NOT** be used with datatype properties. It **MAY** be used with object properties. For object properties, `oslc:range` is used to specify the allowed `rdf:type`s of the object resource. The target resource **SHOULD** be any of the specified `oslc:range` types, but no inferencing is intended if the actual target resource is or is not one of these types. This is very different semantics than `rdfs:range` which does have inferencing implications.

The values of this property **MAY** be any rdf:type URI or the following individual:

oslc:Any
: This value specifies that there is no constraint on the type of the object resource.

[rs-17]

oslc:readOnly Property

If present and true, `oslc:readOnly` is used to specify that the value of defined property is managed by the service, i.e. that it is read-only. It cannot be directly modified by clients of the service. Services **SHOULD** ignore attempts to modify read-only properties, but **MAY** fail such requests. If a service ignores an attempt to modify a read-only property then it **SHOULD NOT** do so silently. A service **MAY** use the HTTP Warning header or some other means to indicate that the attempt to modify a read-only property has been ignored. [rs-18]

In this context, modification means a change in any object of the triples associated with the defined property. For example, a GET request followed by a PUT request would not modify the triples. A service **MUST NOT** interpret a PUT request that does not modify the triples associated with the defined property as a violation of the oslc:readOnly constraint. [rs-19]

Examples of read-only properties include creation and modification timestamps, the identity of who created or modified the described resource, and properties computed from the values of other properties.

When modifying a resource, it is natural for a client to first retrieve its current representation using a GET request. This request will return read-only properties along with read-write properties. If a service fails PUT requests that contain read-only properties then clients will have to remove all read-only properties before submitting PUT requests. The behavior of ignoring read-only properties in PUT requests is therefore more convenient for clients. Similarly, when copying a resource, a client would GET it first. It is therefore more convenient for clients if the service ignores read-only properties also in POST requests.

oslc:representation Property

For object properties, `oslc:representation` is used to specify how the object resource is represented. The value of `oslc:representation` **MUST** be one of the following individuals:

oslc:Either
: There is no constraint on the representation of the object resource.

oslc:Inline
: The representation of the object resource **MUST** be present in the representation of the described resource.

oslc:Reference
: The representation of the object resource **MUST NOT** be present in the representation of the described resource.

[rs-20]

oslc:valueShape Property

For object properties, `oslc:valueShape` is used to specify a link to resource shape that describes the object resource.

oslc:valueType Property

Literal Value Types

For datatype properties, `oslc:valueType` specifies the literal value type. OSLC datatype properties are a subset of the base or primitive types defined by [rdf11-concepts] and [xmlschema11-1]. A datatype **MUST** be one of the following individuals:

rdf:XMLLiteral
: An XML fragment.

xsd:boolean
: A boolean.

xsd:dateTime
: A date-time.

xsd:decimal
: A decimal number.

xsd:double
: A double precision floating point number.

xsd:float
: A single precision floating point number.

xsd:integer
: An integer.

xsd:string
: A string.

rdf:langString
: A string with a language tag. Unless otherwise specified, anywhere OSLC uses xsd:string, rdf:langString may also be used.

[rs-21]

Resource Value Types

For object properties, `oslc:valueType` specifies how the object resource is identified. It **MUST** be one of the following individuals:

oslc:AnyResource
: The object resource **MUST** be identified with either a URI or a blank node.

oslc:LocalResource
: The object resource **MUST** be identified with a blank node. The term "local resource" is used because the scope of identifier is local to the representation.

oslc:Resource
: The object resource **MUST** be identified with a URI.

[rs-22]

5.3 AllowedValues Constraints

- **Describes:** `http://open-services.net/ns/core#AllowedValues`
- **Summary:** Defines a set of allowed values for a defined property.

AllowedValues Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`oslc:allowedValue`	One-or-many	true	unspecified	Either	Unspecified	Specifies the allow values in an AllowedValue constraint.

6. Conformance

Implementations of this specification need to satisfy the following conformance clauses.

Clause Number	Requirement
rs-1	Given a resource R there **MAY** be zero or more shapes S associated with it. This specification defines three ways to associate shapes with a resource, namely using oslc:instanceShape, oslc:resourceShape, and oslc:valueShape. Other specifications **MAY** define additional mechanisms.
rs-2	If one or more shapes are associated with a resource then at least one of those **SHOULD** be applicable to that resource. If no associated shape applies to a resource then this **SHOULD** be interpreted as an error condition.
rs-3	If exactly one shape applies to a resource then that resource **SHOULD** satisfy all the constraints defined by that shape.
rs-4	If more than one shape applies to a resource then that resource **SHOULD** satisfy all the constraints defined by all the shapes (AND semantics). However, the specification for a service description **MAY** define alternate semantics. For example, a service **MAY** require that the resource satisfy the constraints defined by at least one of the shapes (OR semantics).
rs-5	If a resource satisfies its applicable shapes, client and server implementations **MUST** behave as described in the defining OSLC specifications. If a resource does not satisfy its applicable shapes, implementations **SHOULD** attempt to complete the operation with the given data if possible. Otherwise implementations **MAY** reject the operation.
rs-6	The `oslc:instanceShape` property is used to link any described resource with a shape resource that describes its contents. A resource **MAY** be associated with zero or more shapes.
rs-7	The `oslc:resourceShape` property is used to link an application service description with a shape resource that describes some aspect of the service's API contract. A service description **MAY** be linked with zero or more shapes.
rs-8	`dcterms:title` is used to provide a summary of oslc:ResourceShape and oslc:Property resources. Its value **SHOULD** be a literal of type rdf:XMLLiteral that is valid content for an XHTML `<span>` element. If the value contains no XML markup then it **MAY** be represented as a plain text literal or xsd:string.
rs-9	`dcterms:description` is used to provide a description of oslc:Property resources. Its value **SHOULD** be a literal of type rdf:XMLLiteral that is valid content for an XHTML `<div>` element. If the value contains no XML markup then it **MAY** be represented as a plain text literal or xsd:string.
rs-10	`oslc:property` is used to list the defined properties that are expected to be contained in described resources associated with this shape. The object of this property **MUST** be an oslc:Property resource whose representation is contained in the shape document.
rs-11	`oslc:allowedValue` is used to specify an allowed value of the defined property. The object of this property **SHOULD** be compatible with the type specified by the oslc:valueType property if present. An oslc:Property resource **MAY** contain one or more oslc:allowedValue properties and an optional oslc:allowedValues property which links to an oslc:AllowedValues resource. The complete set of allowed values is the union of all the values specified directly in the oslc:Property resource and the linked oslc:AllowedValues resource.
rs-12	`oslc:defaultValue` specifies the default value for the defined property. The object of this property **SHOULD** be compatible with the type specified by the oslc:valueType property if present.
rs-13	A default value is normally used when creating resources. A service **SHOULD** use the default value to provide a value for a property if none is provided in the creation request.
rs-14	A default value **MAY** be used by clients of a described resource if the defined property is not present in the representation of the described resource. This mechanism is useful if a service introduces a new defined property but does not update all pre-existing described resources.

Clause Number	Requirement
rs-15	If present and true, `oslc:hidden` is used to indicate that the defined property is not normally presented to users. A client of the described resource **SHOULD NOT** display hidden defined properties to normal users. It **MAY** display hidden defined properties to administrative users.
rs-16	`oslc:occurs` is used to specify the number of times that the defined property may occur. The value of this property **MUST** be one of the following individuals: oslc:Exactly-one The defined property **MUST** occur exactly once. It is required and single-valued. oslc:One-or-many The defined property **MUST** occur at least once. It is required and multi-valued. oslc:Zero-or-many The defined property **MAY** occur any number of times. It is optional and multi-valued. oslc:Zero-or-one The defined property **MUST** occur no more than once. It is optional and single-valued. For strings and language-tagged strings, single-valued means there is at most one value for any given language tag, and at most one untagged value.
rs-17	`oslc:range` **MUST NOT** be used with datatype properties. It **MAY** be used with object properties. For object properties, `oslc:range` is used to specify the allowed `rdf:type`s of the object resource. The target resource **SHOULD** be any of the specified `oslc:range` types, but no inferencing is intended if the actual target resource is or is not one of these types. This is very different semantics than `rdfs:range` which does have inferencing implications. The values of this property **MAY** be any rdf:type URI or the following individual: oslc:Any This value specifies that there is no constraint on the type of the object resource.
rs-18	Services **SHOULD** ignore attempts to modify read-only properties, but **MAY** fail such requests. If a service ignores an attempt to modify a read-only property then it **SHOULD NOT** do so silently. A service **MAY** use the HTTP Warning header or some other means to indicate that the attempt to modify a read-only property has been ignored.
rs-19	A service **MUST NOT** interpret a PUT request that does not modify the triples associated with the defined property as a violation of the oslc:readOnly constraint.

Clause Number	Requirement
rs-20	For object properties, `oslc:representation` is used to specify how the object resource is represented. The value of `oslc:representation` **MUST** be one of the following individuals: oslc:Either There is no constraint on the representation of the object resource. oslc:Inline The representation of the object resource **MUST** be present in the representation of the described resource. oslc:Reference The representation of the object resource **MUST NOT** be present in the representation of the described resource.
rs-21	For datatype properties, `oslc:valueType` specifies the literal value type. OSLC datatype properties are a subset of the base or primitive types defined by [rdf11-concepts] and [xmlschema11-1]. A datatype **MUST** be one of the following individuals: rdf:XMLLiteral An XML fragment. xsd:boolean A boolean. xsd:dateTime A date-time. xsd:decimal A decimal number. xsd:double A double precision floating point number. xsd:float A single precision floating point number. xsd:integer An integer. xsd:string A string. rdf:langString A string with a language tag. Unless otherwise specified, anywhere OSLC uses xsd:string, rdf:langString may also be used.

Clause Number	Requirement
rs-22	For object properties, `oslc:valueType` specifies how the object resource is identified. It **MUST** be one of the following individuals: oslc:AnyResource The object resource **MUST** be identified with either a URI or a blank node. oslc:LocalResource The object resource **MUST** be identified with a blank node. The term "local resource" is used because the scope of identifier is local to the representation. oslc:Resource The object resource **MUST** be identified with a URI.

Appendix A. Acknowledgements

This section is non-normative.

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

John Arwe (IBM)
Nick Crossley (IBM)
Miguel Esteban Gutiérrez (Universidad Politécnica de Madrid)
Arnaud Le Hors (IBM)
Martin Nally (IBM)
Eric Prud'hommeaux (W3C)
Arthur Ryman (IBM)
Steve Speicher (IBM)
Tack Tong (IBM)

In addition, the OSLC Open Project would like to call out special recognition of the significant contribution by the originators of the Resource Shape concepts. The OSLC Resource Shape specification was initially developed by the OSLC Reporting Workgroup under the leadership of Tack Tong (IBM), with major contributions from Arthur Ryman (IBM) and Martin Nally (IBM). Members of OSLC workgroups found shapes to be applicable to other aspects of OSLC and so the specification was subsequently integrated into the OSLC Core specification by Dave Johnson (IBM).

Anamitra Bhattacharyya (IBM) provided valuable feedback based on implementation experience and input on datatype facets. Steve Speicher (IBM) and John Arwe (IBM) contributed proposals for extending shapes which has informed the [SHACL] work.

OASIS OPEN PROJECTS

OSLC Core Version 3.0. Part 7: Vocabulary

OASIS Standard
26 August 2021

This stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.pdf

Previous stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/core-vocab.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/core-vocab.pdf
(published as Project Specification)

Latest stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/core-vocab.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/core-vocab.pdf

Latest version:

https://open-services.net/spec/core/latest

Latest editor's draft:

https://open-services.net/spec/core/latest-draft

Open Project:

OASIS Open Services for Lifecycle Collaboration (OSLC) OP

Project Chairs:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Editors:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Additional components:

This specification is one component of a Work Product that also includes:

- *OSLC Core Version 3.0. Part 1: Overview.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/oslc-core.html
- *OSLC Core Version 3.0. Part 2: Discovery*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/discovery.html
- *OSLC Core Version 3.0. Part 3: Resource Preview.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-preview.html

- *OSLC Core Version 3.0. Part 4: Delegated Dialogs*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/dialogs.html
- *OSLC Core Version 3.0. Part 5: Attachments*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/attachments.html
- *OSLC Core Version 3.0. Part 6: Resource Shape*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-shape.html
- *OSLC Core Version 3.0. Part 7: Vocabulary (this document)*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.html
- *OSLC Core Version 3.0. Part 8: Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.html
- *OSLC Core Version 3.0. Machine Readable Vocabulary Terms*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.ttl
- *OSLC Core Version 3.0. Machine Readable Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.ttl

Related work:

This specification is related to:

- *OSLC Core Version 3.0: Link Guidance*. https://oslc-op.github.io/oslc-specs/notes/link-guidance.html

RDF Namespaces:

http://open-services.net/ns/core#

Abstract:

Core Vocabulary defines the OSLC Core RDF vocabulary terms and resources, that have broad applicability across various domains.

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the “Latest stage” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Open Project are listed at https://github.com/oslc-op/oslc-specs.

Comments on this work can be provided by opening issues in the project repository or by sending email to the project’s public comment list oslc-op.

The English version of this specification is the only normative version. Non-normative translations may also be available. Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:
[OSLC-CoreVocab-3.0]
OSLC Core Version 3.0. Part 7: Vocabulary. Edited by Jim Amsden and Andrii Berezovskyi. 26 August 2021. OASIS Standard. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.html. Latest stage: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/core-vocab.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

Table of Contents

1. Introduction

This section is non-normative.

Various resources and properties may be so commonly used or apply so broadly that it makes sense to define them in one place so they can be easily reused. Some common examples are short names or labels, error messages, discussion threads, traceability/impacts relationship behavior or annotating other vocabulary terms.

See OSLC Core Version 3.0. Part 8: Constraints for the standard OSLC constraints defined on this vocabulary.

1.1 Terminology

Terminology uses and extends the terminology and capabilities of OSLC Core Overview, W3C Linked Data Platform [LDP], W3C's Architecture of the World Wide Web [WEBARCH], Hyper-text Transfer Protocol [HTTP11].

Archived Resource

A resource in which an explicit action has been performed to mark the resource as no longer active.

1.2 References

1.2.1 Normative references

[HTTP11]

R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. IETF, June 2014. Proposed Standard. URL: https://httpwg.org/specs/rfc7230.html

[LDP]

Steve Speicher; John Arwe; Ashok Malhotra. *Linked Data Platform 1.0*. W3C, 26 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/ldp/

[RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF, March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

[RFC8174]

B. Leiba. *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*. IETF, May 2017. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc8174

1.2.2 Informative references

[LinkGuidance]

Steve Speicher; Jim Amsden. *OSLC Link Guidance 3.0*. OASIS. URL: https://tools.oasis-open.org/version-control/svn/oslc-core/trunk/supporting-docs/link-guidance.html

[WEBARCH]

Ian Jacobs; Norman Walsh. *Architecture of the World Wide Web, Volume One*. W3C, 15 December 2004. W3C Recommendation. URL: https://www.w3.org/TR/webarch/

1.3 Typographical Conventions and Use of RFC Terms

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this specification are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Motivation

This section is non-normative.

Most OSLC vocabularies and resource shape constraints on usages of those vocabularies are given in the various OSLC domain specifications. The motivation for these domain specifications is to define agreed upon, formalized vocabulary terms for key elements in the domain. Domain vocabularies are not intended to restrict what vocabularies servers actually use for those domains, or what domains they support. Rather vocabularies establish a common core of domain terms that reduce accidental variability and foster greater interchange and interoperability between tools that support and users that make use of domains. Servers are free to extend the domains and integrate across domains as required to realize their provided capabilities.

OSLC Core takes a similar approach for common terms that are used across most domains. The intent is to provide a foundation for building domains that again reduces unnecessary variability, and eliminates the need for the various domain specifications to redundantly define similar terms. The follow paragraphs describe the kinds of common terms defined by OSLC core in order to achieve the stated intent.

3. Consideration for select vocabulary terms

3.1 Archived resources

Archived Resources are typically found in large systems in which an immutable copy of the state of a resource at a given time is captured. The purpose may vary in that it could be simply a way to facilitate access to a backup or snapshot of a resource at a particular point in time. Another use may be to indicate that a resource has been deleted, but is saved by the system for historical or legal reasons. Having a consistent way to indicate that a resource, or a set of them, has been archived helps when defining certain views of the resources or queries.
Archived Resources **MAY** be identified by having a property `oslc:archived`, with value `true`. [cc-1]
Archived Resources **MAY** be removed from typical user interactions. [cc-2]
Archived Resources **SHOULD** be considered immutable. [cc-3]

3.2 Comments

Many different kinds of applications have a way to provide comments or notes related to a given resource. These take the form of a discussion, with a sequence of comments. OSLC Core provide a common way for applications to easily add to a comment to a discussion thread or navigate a discussion thread.

3.3 Errors

Error responses from HTTP request often take the form of HTML pages intended for a human to read, even though these requests are often initiated from applications that don't have a human actively monitoring it. OSLC Core defines a consistent way to request error responses of a certain format, and a prescribed interaction model that helps clients better handle errors automatically.

3.4 Impact of resource changes on links

Some RDF properties express relations or links between subject and object artifacts. If a change in state of subject and/or object of a triple may result in the assertion becoming invalid, the link may be seen to represent a dependency. OSLC Core provides property `oslc:impactType` as a means of defining the dependency represented by an RDF property.

3.5 Inverse properties

Consider a user interface for a query builder that allows users to build queries about test cases. It is natural for the query builder to present the user with a list of the properties that apply to test cases that could be used in the query. Suppose the user wants to build a query that returns all the requirements that are validated by a test case. The query builder should describe the available properties from the point of view of the test case. This implies that the query builder should describe the inverse relation asserted by any triple that has the test case as an object. In this example, the query builder should describe assertions of the form {requirement `oslc_rm:validatedBy` test case} as {test case `validates` requirement}.

4. Defining Enumerations

This section is non-normative.

Some property values are are characterized by a limited set of enumerated values. The type for these property values is called an enumeration in many modeling and programming languages, while the values are called enumeration literals. RDF does not define a specific way of defining enumerated types and enumeration literals. As a result, different vocabularies may take different, but equally valid approaches. In order to foster interoperability and integration, OSLC Core provides a recommended approach for defining enumerated types and enumeration literals. This approach is used in defining the OSLC Core vocabulary terms.

Enumerations in an OSLC vocabulary should be defined as an RDF class. Enumeration literals are the URIs of individuals of that class. For example, consider an enumeration called "Color" that has enumeration literals `{red, yellow, green, blue}` (using Java notation). Color would be defined as an RDF class and the enumeration literals would be individuals of that class. A `color` property is defined and then used to assert that the color of myCar is blue.

EXAMPLE 1

```
# Color enumeration
Color
    a rdfs:Class ;
    rdfs:label "Color" ;
    rdfs:comment "The class of possible color values." .

# Color enumeration literals
red
    a Color ;
    rdfs:label "red" .

yellow
    a Color ;
    rdfs:label "yellow" .

green
    a Color ;
    rdfs:label "green" .

blue
    a Color ;
    rdfs:label "blue" .

# A Color property
color
    a rdf:Property ;
    rdfs:label "color" ;
    rdfs:comment "Used to specify the color of a resource".

# Asserting the color of a resource
myCar color blue.
```

Enumerations can be open or closed. Open enumerations allow additional enumeration literals to be added as needed. Closed enumerations have a fixed set of enumeration literals that is not intended to be extended. Resource shapes can be used to constrain enumerations to a specific set of values. Notice in the example above that the color property did not specify its rdfs:range. This keeps the enumeration completely open to any set of individuals. OSLC prefers to use resource shapes to constrain resources for particular usages, leaving them open for extension for other, possibly unanticipated usages.

A shape can be used to constrain the Color enumeration for a specific purpose. For example, the color of lights in a traffic light should be constrained to exactly red, yellow and green.

EXAMPLE 2

```
# Create a constraint on Color for traffic lights
TrafficLightConstraint
    a oslc:ResourceShape ;
    oslc:describes fhwa:TrafficLight ;
    dcterms:title "Establish constraints for traffic light colors" ;
    oslc:property colorConstraint .

colorConstraint
    a oslc:Property ;
    oslc:name "color" ;
    dcterms:description "The colors for a traffic light as specified by FHWA."
    oslc:propertyDefinition color ;
    oslc:occurs oslc:Exactly-one ;
    oslc:range Color ;
    oslc:allowValue red, yellow, green ;
    oslc:readOnly false ;
    oslc:representation oslc:Reference ;
    oslc:valueType oslc:Resource .
```

TrafficLightConstraint defines a constraint associated with the vocabulary term **`fhwa:TrafficLight`**. The constraint has one property, colorConstraint whose **`oslc:propertyDefinition`** is the color RDF property. The **`oslc:range`** for the colorConstraint is set to Color, meaning the value of the applicable property is constrained to be of **`rdf:type`** Color. The **`oslc:allowedValue`** property further constrains the values to be red, yelow, or green. If the **`oslc:allowedValue`** were not specified, then the TrafficLightConstraint would allow the enumeration to be open.

A completely different shape constraint could be used for colors that represent the status of a risk mitigation in a software development project.

5. Terms for describing vocabularies

5.1 Inverse Labels

The W3C RDF Schema vocabulary defines the vocabulary annotation property `rdfs:label`. This property is intended to provide a human-readable description for a resource's name. It is often used to provide a label for RDF properties. [LinkGuidance] discourages the creation of inverse predicates. However, there is still a need for a property, like rdfs:label, to specify an inverse label for a predicate.

For example, consider the OSLC Requirements Management (RM) property oslc_rm:validatedBy. When used as the predicate of a triple, this property is used to assert that the subject resource, e.g. a Requirement, is validated by the object resource, e.g. a TestCase. The rdfs:label for this property is "validatedBy".

Now consider the user interface of a query builder that allows users to build queries about TestCases. It is natural for the query builder to present the user with a list of the properties that apply to TestCases. Suppose the user wants to build a query that returns all the Requirements that are validated by a TestCase. The query builder should describe the available properties from the point of view of the TestCase. This implies that the query builder should describe the inverse relation asserted by any triple that has the TestCase as an object. In our example, the query builder should describe oslc_rm:validatedBy as "validates".

The oslc:inverseLabel property provides a human-readable label for the inverse of the subject property.

For example, the following triple (in Turtle notation) would be added to the OSLC RM vocabulary:

EXAMPLE 3

```
oslc_rm:validatedBy oslc:inverseLabel "validates".
```

It should be noted that the use of inverse labels is independent of the existence of explicit RDF inverse properties. However, if an inverse property is defined by some vocabulary, then a consistent label should be used in order to avoid confusion. In general, it is good practice to avoid the creation of inverse properties since it creates redundant information and complicates SPARQL queries. Instead, a single property should be wherever possible and it should be given an inverse label in order to describe the property from the perspective of the object.

For example, the OSLC Quality Management (QM) vocabulary defines two properties that are approximately inverse to oslc_rm:validatedBy. These are oslc_qm:validatesRequirement and oslc_qm:validatesRequirementCollection. In this case the choice of inverse label "validates" for oslc_rm:validatedBy is consistent with the actual labels of the inverse properties, namely "validatesRequirement" and "validatesRequirementCollection".

5.2 Traceability and Impact type

Some RDF properties express dependency relations between artifacts, and it is often very valuable to trace the impact of a change in an artifact to those artifacts that depend on it directly or indirectly. The concept of dependency is very general. For example, the concept of trace relations is described in SysML: "A generic trace requirement relationship provides a general-purpose relationship between a requirement and any other model element. The semantics of trace include no real constraints and therefore are quite weak."

As a general guideline, if any assertion involving a given predicate may become invalid if the state of either its subject or object resources change, then we may legitimately regard that predicate as expressing a dependency relation, in which case it may be useful to explicitly describe the nature of the dependency.

An assertion describes a link between subject and object resources whose name is the property or predicate of the assertion. A dependency relationship may be in the same direction as the link, the opposite direction, both directions, or the link may not represent any dependency whose impact might need to be assessed.

For example, in assertions such as {requirement validatedBy testcase}, it may be important to assess the impact of a change

in the requirement or a change in the testcase. Typically test cases are updated to reflect changes in requirements in order perform the correct validation. So in this case, property validatedBy would introduce impact that follows the link, from the subject requirement to the object testcase. However, if a team is doing test-driven development, they may treat test cases as formal, executable specifications of requirements and the requirement is simply an informal description of the test case. In this case, the team might consider the impact to be opposite of the link, from the testcase to the requirement.

The property **`oslc:impactType`** asserts that the subject property is a dependency relation and gives the direction of impact. The resources **`oslc:FollowsLink`** and **`oslc:OppositeLink`** identify whether the impact follows the direction of the assertion (subject to object), or the opposite direction (object to subject). **`oslc:SymmetricImpact`** describes a symmetric dependency relation in which the property represents a dependency from both subject to object and object to subject. **`oslc:NoImpact`** indicates the predicate does not represent any dependency between the subject and object resources.

For example, the following triple (in Turtle notation) would be added a vocabulary to indicate test cases are dependent on requirements:

EXAMPLE 4

```
ex:validatedBy oslc:impactType oslc:FollowsLink .
```

The same dependency could also be described from the perspective of the test case. In this case, the dependency is opposite of the validatesRequirement predicate:

EXAMPLE 5

```
ex:validatesRequirement oslc:impactType oslc:OppositeImpact .
ex:validatesRequirementCollection oslc:impactType oslc:OppositeImpact.
```


6. Discovery

Vocabulary terms are discovered via published vocabulary documents at the OSLC Core namespace and shapes at advertised URLs.

7. Terms

7.1 Vocabulary Details

The namespace URI for this vocabulary is: `http://open-services.net/ns/core#`

All vocabulary URIs defined in the OSLC Core namespace.

7.1.1 Classes in this namespace (27)

AllowedValues, Any, AttachmentContainer, AttachmentDescriptor, Cardinality, Comment, Compact, CreationFactory, Dialog, Discussion, Error, ExtendedError, ImpactType, OAuthConfiguration, PrefixDefinition, Preview, Property, Publisher, QueryCapability, Representation, ResourceShape, ResourceShapeConstraints, ResourceValueType, ResponseInfo, Service, ServiceProvider, ServiceProviderCatalog

AllowedValues

http://open-services.net/ns/core#AllowedValues

AllowedValues is an RDFS class.

Provides a way to specify allowed values for one or more properties.

Any

http://open-services.net/ns/core#Any

Any is an RDFS class.

Any value type is allowed.

AttachmentContainer

http://open-services.net/ns/core#AttachmentContainer

AttachmentContainer is an RDFS class.

An LDP-C that contains attachments for a resource.

AttachmentDescriptor

http://open-services.net/ns/core#AttachmentDescriptor

AttachmentDescriptor is an RDFS class.

An LDP-RS that contains additional data about an attachment.

Cardinality

http://open-services.net/ns/core#Cardinality

Cardinality is an RDFS class.

The number of allowed values for a property.

Comment

http://open-services.net/ns/core#Comment

Comment is an RDFS class.

A Comment resource represents a single note, or comment, in a discussion thread.

Compact

http://open-services.net/ns/core#Compact

Compact is an RDFS class.

A resource describing how to display a link and Preview for another, associated resource.

CreationFactory

http://open-services.net/ns/core#CreationFactory

CreationFactory is an RDFS class.

The CreationFactory definition included in a ServiceProvider.

Dialog

http://open-services.net/ns/core#Dialog

Dialog is an RDFS class.

Describes information about a dialog such as its title and dimensions.

Discussion

http://open-services.net/ns/core#Discussion

Discussion is an RDFS class.

A Discussion resource is intended to represent a sequence of comments or notes regarding the associated resource.

Error

http://open-services.net/ns/core#Error

Error is an RDFS class.

Basis for forming an error response.

ExtendedError

http://open-services.net/ns/core#ExtendedError

ExtendedError is an RDFS class.

Extended error information.

ImpactType

http://open-services.net/ns/core#ImpactType

ImpactType is an RDFS class.

An enumeration of specifying different impact types or a property.

OAuthConfiguration

http://open-services.net/ns/core#OAuthConfiguration

OAuthConfiguration is an RDFS class.

The OAuthConfiguration definition included in ServiceProvider.

PrefixDefinition

http://open-services.net/ns/core#PrefixDefinition

PrefixDefinition is an RDFS class.

The PrefixDefinition definition included in ServiceProvider.

Preview

http://open-services.net/ns/core#Preview

*Preview*is an RDFS class.

An HTML representation of a resource that can be embedded in another user interface.

Property

http://open-services.net/ns/core#Property

Property is an RDFS class.

A Property resource describes one allowed or required property of a resource.

Publisher

http://open-services.net/ns/core#Publisher

Publisher is an RDFS class.

The Publisher definition included in ServiceProvider.

QueryCapability

http://open-services.net/ns/core#QueryCapability

QueryCapability is an RDFS class.

The QueryCapability definition included in a ServiceProvider.

Representation

http://open-services.net/ns/core#Representation

Representation is an RDFS class.

Specifies how a resource is represented in a document.

ResourceShape

http://open-services.net/ns/core#ResourceShape

ResourceShape is an RDFS class.

The Resource Shape used for creation, query and modify. Formally, a shape S applies to a resource R if there is a triple R rdf:type T and there is a triple S oslc:describes T, or if there is a triple R oslc:instanceShape S.

ResourceShapeConstraints

http://open-services.net/ns/core#ResourceShapeConstraints

ResourceShapeConstraints is an RDFS class.

Resource Shape Constraints metadata

ResourceValueType

http://open-services.net/ns/core#ResourceValueType

ResourceValueType is an RDFS class.

Specifies how an object reference is represented in a document.

ResponseInfo

http://open-services.net/ns/core#ResponseInfo

ResponseInfo is an RDFS class.

The ResponseInfo included in query results.

Service

http://open-services.net/ns/core#Service

Service is an RDFS class.

The Service definition included in a ServiceProvider.

ServiceProvider

http://open-services.net/ns/core#ServiceProvider

ServiceProvider is an RDFS class.

The Service Provider resource.

ServiceProviderCatalog

http://open-services.net/ns/core#ServiceProviderCatalog

ServiceProviderCatalog is an RDFS class.

The Service Provider Catalog resource.

7.1.2 Properties in this namespace (81)

allowedValue, allowedValues, archived, attachment, attachmentSize, authorizationURI, cause, comment, creation, creationDialog, creationFactory, default, defaultValue, describes, details, dialog, discussedBy, discussionAbout, document, domain, error, executes, extendedError, futureAction, hidden, hintHeight, hintWidth, icon, iconAltLabel, iconSrcSet, iconTitle, impactType, initialHeight, inReplyTo, instanceShape, inverseLabel, isMemberProperty, label, largePreview, maxSize, message, modifiedBy, moreInfo, name, nextPage, oauthAccessTokenURI, oauthConfiguration, oauthRequestTokenURI, occurs, order, partOfDiscussion, postBody, prefix, prefixBase, prefixDefinition, property, propertyDefinition, publisher, queryable, queryBase, queryCapability, range, readOnly, rel, representation, resourceShape, resourceType, results, score, selectionDialog, service, serviceProvider, serviceProviderCatalog, shortId, shortTitle, smallPreview, statusCode, totalCount, usage, valueShape, valueType

allowedValue

http://open-services.net/ns/core#allowedValue

allowedValue is an RDF property.

Specifies the allowed values for a property (may be more than one).

allowedValues

http://open-services.net/ns/core#allowedValues

allowedValues is an RDF property.

Reference to an AllowedValues resource that specifies the allowed values for the property.

archived

http://open-services.net/ns/core#archived

archived is an RDF property.

Indicates whether the subject has been marked as archived, no longer an actively updating resource.

attachment

http://open-services.net/ns/core#attachment

attachment is an RDF property.

An attachment associated with a resource. May be used as a membership predicate for an attachment container.

attachmentSize

http://open-services.net/ns/core#attachmentSize

attachmentSize is an RDF property.

Size in bytes of the attachment content.

authorizationURI

http://open-services.net/ns/core#authorizationURI

authorizationURI is an RDF property.

URI for obtaining OAuth authorization.

cause

http://open-services.net/ns/core#cause

cause is an RDF property.

An error that is a cause of this error.

comment

http://open-services.net/ns/core#comment

comment is an RDF property.

Comment about the resource.

creation

http://open-services.net/ns/core#creation

creation is an RDF property.

To create a new resource via the factory, post it to this URI.

creationDialog

http://open-services.net/ns/core#creationDialog

creationDialog is an RDF property.

Enables clients to create a resource via UI.

creationFactory

http://open-services.net/ns/core#creationFactory

creationFactory is an RDF property.

Enables clients to create new resources.

default

http://open-services.net/ns/core#default

default is an RDF property.

Used in conjunction with oslc:usage property used to identify which service is the default usage.

defaultValue

http://open-services.net/ns/core#defaultValue

defaultValue is an RDF property.

A default value for property, inlined into property definition.

describes

http://open-services.net/ns/core#describes

describes is an RDF property.

This shape describes resources that are of the RDF type given by the object of the oslc:describes predicate. Formally, a shape S applies to a resource R if there is a triple R rdf:type T and there is a triple S oslc:describes T.

details

http://open-services.net/ns/core#details

details is an RDF property.

A URL that may be used to retrieve a resource to determine additional details about the service provider.

dialog

http://open-services.net/ns/core#dialog

dialog is an RDF property.

The URI of the HTML dialog.

discussedBy

http://open-services.net/ns/core#discussedBy

discussedBy is an RDF property.

A series of notes and comments about this resource.

discussionAbout

http://open-services.net/ns/core#discussionAbout

discussionAbout is an RDF property.

Reference to associated resource.

document

http://open-services.net/ns/core#document

document is an RDF property.

The URI of an HTML document to be used for the preview.

domain

http://open-services.net/ns/core#domain

domain is an RDF property.

Namespace URI of the specification that is implemented by this service. In most cases this namespace URI will be for an OSLC domain, but other URIs **MAY** be used.

error

http://open-services.net/ns/core#error

error is an RDF property.

Error information that may be associated with a resource.

executes

http://open-services.net/ns/core#executes

executes is an RDF property.

Link from a currently available action to the future action it realizes.

extendedError

http://open-services.net/ns/core#extendedError

extendedError is an RDF property.

Extended (additional) error information.

futureAction

http://open-services.net/ns/core#futureAction

futureAction is an RDF property.

A predicate that links to an action that is not currently executable on the subject resource, but may be executable in the future and/or on other resources. For example, in OSLC Automation this is expected to link from an oslc_auto:AutomationPlan to an oslc:Action resource with zero bindings (as it is not executable), with the meaning that the executable form of the action may be available on oslc_auto:AutomationResult resources generated by executing that Automation Plan. Similarly, resource shapes can allow discovery of actions available on the output of a creation factory.

hidden

http://open-services.net/ns/core#hidden

hidden is an RDF property.

A hint that indicates that property **MAY** be hidden when presented in a user interface.

hintHeight

http://open-services.net/ns/core#hintHeight

hintHeight is an RDF property.

Preferred height of a delegated user interface. Values must be expressed using length units as specified in Cascading Style Sheets 2.1.

hintWidth

http://open-services.net/ns/core#hintWidth

hintWidth is an RDF property.

Preferred width of a delegated user interface. Values must be expressed using length units as specified in Cascading Style Sheets 2.1.

icon

http://open-services.net/ns/core#icon

icon is an RDF property.

URI of an image applicable to the resource.

iconAltLabel

http://open-services.net/ns/core#iconAltLabel

iconAltLabel is an RDF property.

Alternative label used in association with the oslc:icon, such as HTML img tag's alt attribute.

iconSrcSet

http://open-services.net/ns/core#iconSrcSet

iconSrcSet is an RDF property.

Specification of a set of images of different sizes based on HTML img element srcset attribute.

iconTitle

http://open-services.net/ns/core#iconTitle

iconTitle is an RDF property.

Title used in association with the oslc:icon, such as HTML img tag's title attribute.

impactType

http://open-services.net/ns/core#impactType

impactType is an RDF property.

Asserts that the subject property is a dependency relation and gives the direction of impact.

initialHeight ***(Archaic term)***

http://open-services.net/ns/core#initialHeight

initialHeight is an RDF property.

Recommended initial height of the preview. The presence of this property indicates that the preview supports dynamically computing its size. Values are expressed in relative length units as defined in the W3C Cascading Style Sheets Specification (CSS 2.1). Em and ex units are interpreted relative to the default system font (at 100% size).

inReplyTo

http://open-services.net/ns/core#inReplyTo

inReplyTo is an RDF property.

Reference to comment this comment is in reply to.

instanceShape

http://open-services.net/ns/core#instanceShape

instanceShape is an RDF property.

The URI of a Resource Shape that describes the possible properties.

inverseLabel

http://open-services.net/ns/core#inverseLabel

inverseLabel is an RDF property.

Provides a human-readable label for the inverse of the subject property.

isMemberProperty

http://open-services.net/ns/core#isMemberProperty

isMemberProperty is an RDF property.

Used to define when a property is a member of a container, useful for query.

label

http://open-services.net/ns/core#label

label is an RDF property.

Very short label for use in menu items.

largePreview

http://open-services.net/ns/core#largePreview

*largePreview*is an RDF property.

URI and sizing properties for an HTML document to be used for a large preview.

maxSize

http://open-services.net/ns/core#maxSize

maxSize is an RDF property.

For String properties only, specifies maximum characters allowed. If not set, then there is no maximum or maximum is specified elsewhere.

message

http://open-services.net/ns/core#message

message is an RDF property.

An informative message describing the error that occurred.

modifiedBy

http://open-services.net/ns/core#modifiedBy

modifiedBy is an RDF property.

The URI of a resource describing the entity that most recently modified this resource. The link target is usually a foaf:Person or foaf:Agent, but could be any type. This is modeled after dcterms:creator, but Dublin Core currently has no equivalent property.

moreInfo

http://open-services.net/ns/core#moreInfo

moreInfo is an RDF property.

{{A resource giving more information on the error, with an HTML content-type.

name

http://open-services.net/ns/core#name

name is an RDF property.

Name of property being defined, i.e. second part of property's Prefixed Name. For all other uses, consider dcterms:title, rdfs:label, oslc:shortTitle or oslc:label.

nextPage

http://open-services.net/ns/core#nextPage

nextPage is an RDF property.

Link to next page of response.

oauthAccessTokenURI

http://open-services.net/ns/core#oauthAccessTokenURI

oauthAccessTokenURI is an RDF property.

URI for obtaining OAuth access token.

oauthConfiguration

http://open-services.net/ns/core#oauthConfiguration

oauthConfiguration is an RDF property.

Defines the three OAuth URIs required for a client to act as an OAuth consumer.

oauthRequestTokenURI

http://open-services.net/ns/core#oauthRequestTokenURI

oauthRequestTokenURI is an RDF property.

URI for obtaining OAuth request token.

occurs

http://open-services.net/ns/core#occurs

occurs is an RDF property.

One of the values http://open-services.net/ns/core#Exactly-one, http://open-services.net/ns/core#Zero-or-one, http://open-services.net/ns/core#Zero-or-many or http://open-services.net/ns/core#One-or-many.

order

http://open-services.net/ns/core#order

order is an RDF property.

A computed property for each member resource of a query with an orderBy clause supporting sorting of the RDF results.

partOfDiscussion

http://open-services.net/ns/core#partOfDiscussion

partOfDiscussion is an RDF property.

Reference to owning Discussion resource .

postBody

http://open-services.net/ns/core#postBody

postBody is an RDF property.

The body of a POST request to return the next page if the response was to a POST request. Where a paged resource supports POST with an application/x-www-form-urlencoded body as an alternative to GET to avoid the request URI exceeding server limitations, the oslc:ResponseInfo in the response to the POST **SHOULD** contain this property so that a client knows what to POST to get the next page.

prefix

http://open-services.net/ns/core#prefix

prefix is an RDF property.

Namespace prefix to be used for this namespace.

prefixBase

http://open-services.net/ns/core#prefixBase

prefixBase is an RDF property.

The base URI of the namespace.

prefixDefinition

http://open-services.net/ns/core#prefixDefinition

prefixDefinition is an RDF property.

Defines a namespace prefix for use in JSON representations and in forming OSLC Query Syntax strings.

property

http://open-services.net/ns/core#property

property is an RDF property.

The properties that are allowed or required by this shape.

propertyDefinition

http://open-services.net/ns/core#propertyDefinition

propertyDefinition is an RDF property.

URI of the property whose usage is being described.

publisher ***(Archaic term)***

http://open-services.net/ns/core#publisher

publisher is an RDF property.

An entity responsible for making the resource available. Servers should use dcterms:publisher.

queryable

http://open-services.net/ns/core#queryable

queryable is an RDF property.

Indicates whether a property is queryable (can appear in oslc.where and olsc.select clause) or not.

queryBase

http://open-services.net/ns/core#queryBase

queryBase is an RDF property.

The base URI to use for queries. Queries may be invoked either by HTTP GET or HTTP POST. For HTTP GET, a query URI is formed by appending a key=value pair to the base URI. For HTTP POST, the query parameters are encoded as content with media type application/x-www-form-urlencoded and sent in the request body. The base URI **MAY** accept other query languages and media types in the request body, e.g. application/sparql-query for SPARQL queries.

queryCapability

http://open-services.net/ns/core#queryCapability

queryCapability is an RDF property.

Enables clients query across a collection of resources.

range

http://open-services.net/ns/core#range

range is an RDF property.

For properties with a resource value-type, Providers **MAY** also specify the range of possible resource types allowed, each specified by URI. The default range is http://open-services.net/ns/core#Any.

readOnly

http://open-services.net/ns/core#readOnly

readOnly is an RDF property.

true if the property is read-only. If omitted, or set to false, then the property is writable. Providers **SHOULD** declare a property read-only when changes to the value of that property will not be accepted after the resource has been created, e.g. on PUT/PATCH requests. Consumers should note that the converse does not apply: Providers **MAY** reject a change to the value of a writable property.

rel

http://open-services.net/ns/core#rel

rel is an RDF property.

If present and set to 'alternate' then indicates that work-around is provided, behavior for other values is undefined.

representation

http://open-services.net/ns/core#representation

representation is an RDF property.

Should be http://open-services.net/ns/core#Reference, http://open-services.net/ns/core#Inline or http://open-services.net/ns/core#Either.

resourceShape

http://open-services.net/ns/core#resourceShape

resourceShape is an RDF property.

A Creation Factory **MAY** provide Resource Shapes that describe shapes of resources that may be created.

resourceType

http://open-services.net/ns/core#resourceType

resourceType is an RDF property.

The expected resource type URI of the resource that will be created using this creation factory. These would be the URIs found in the result resource's rdf:type property.

results

http://open-services.net/ns/core#results

results is an RDF property.

Used to hold the results of dialog action or JSON query results (default). The JSON query result attribute 'oslc:results' is used whenever a provider doesn't have a suitable property already in its model for such purposes.

score

http://open-services.net/ns/core#score

score is an RDF property.

A computed property for each member resource of a full text search query indicating the quality of the match, and sort them in

descending order of score.

selectionDialog

http://open-services.net/ns/core#selectionDialog

selectionDialog is an RDF property.

Enables clients to select a resource via a UI.

service

http://open-services.net/ns/core#service

service is an RDF property.

Describes a service offered by the service provider.

serviceProvider

http://open-services.net/ns/core#serviceProvider

serviceProvider is an RDF property.

A link to the resource's OSLC Service Provider.

serviceProviderCatalog

http://open-services.net/ns/core#serviceProviderCatalog

serviceProviderCatalog is an RDF property.

Additional service provider catalog.

shortId

http://open-services.net/ns/core#shortId

shortId is an RDF property.

A short, human-readable, plain text value. This value should be unique in some context that is apparent to human users of a service.

shortTitle

http://open-services.net/ns/core#shortTitle

shortTitle is an RDF property.

Shorter form of dcterms:title for the resource.

smallPreview

http://open-services.net/ns/core#smallPreview

*smallPreview*is an RDF property.

URI and sizing properties for an HTML document to be used for a small preview.

statusCode

http://open-services.net/ns/core#statusCode

statusCode is an RDF property.

The HTTP status code reported with the error.

totalCount

http://open-services.net/ns/core#totalCount

totalCount is an RDF property.

This optional property indicates the total number of results across all pages, its value should be non-negative. In the context of a query resource, this value **SHOULD** be the total number of results, i.e. the number of resources that match the query. In the context of other resources, the value **SHOULD** be the total number of property values (i.e. RDF triples) of the resource. Unless Stable Paging is in effect, the total count **MAY** vary as a client retrieves subsequent pages.

usage

http://open-services.net/ns/core#usage

usage is an RDF property.

An identifier URI for the domain specified usage of this creation factory. If a service provides multiple creation factories, it may designate the primary or default one that should be used with a property value of http://open-services.net/ns/core#default.

valueShape

http://open-services.net/ns/core#valueShape

valueShape is an RDF property.

if the value-type is a resource type, then Property **MAY** provide a shape value to indicate the Resource Shape that applies to the resource.

valueType

http://open-services.net/ns/core#valueType

valueType is an RDF property.

A URI that indicates the value type, for example XML Schema or RDF URIs for literal value types, and OSLC-specified for others. If this property is omitted, then the value type is unconstrained.

7.1.3 Resources (Individuals) in this namespace (14)

AnyResource, Either, Exactly-one, ImpactFollowsLink, ImpactOppositeLink, Inline, LocalResource, NoImpact, One-or-many, Reference, Resource, SymmetricImpact, Zero-or-many, Zero-or-one

AnyResource

http://open-services.net/ns/core#AnyResource

AnyResource is an RDF individual.

The object resource can be identified with either a URI or a blank node.

Either

http://open-services.net/ns/core#Either

Either is an RDF individual.

Representation is either a URI reference or blank node.

Exactly-one

http://open-services.net/ns/core#Exactly-one

Exactly-one is an RDF individual.

Property with value is required.

ImpactFollowsLink

http://open-services.net/ns/core#ImpactFollowsLink

ImpactFollowsLink is an RDF individual.

The property represents a dependency from subject to object.

ImpactOppositeLink

http://open-services.net/ns/core#ImpactOppositeLink

ImpactOppositeLink is an RDF individual.

The property represents a dependency from object to subject.

Inline

http://open-services.net/ns/core#Inline

Inline is an RDF individual.

The representation of the object resource must be present in the representation of the described resource.

LocalResource

http://open-services.net/ns/core#LocalResource

LocalResource is an RDF individual.

The object resource must be identified with a blank node. The term 'local resource' is used because the scope of identifier is local to the representation. Clients and servers should use oslc:representation oslc:Inline instead to include resource representations in the same document.

NoImpact

http://open-services.net/ns/core#NoImpact

NoImpact is an RDF individual.

The property does not represent a dependency.

One-or-many

http://open-services.net/ns/core#One-or-many

One-or-many is an RDF individual.

Property is required and multi-valued.

Reference

http://open-services.net/ns/core#Reference

Reference is an RDF individual.

A URI Reference representation to a resource.

Resource

http://open-services.net/ns/core#Resource

Resource is an RDF individual.

The object resource must be identified with a URI.

SymmetricImpact

http://open-services.net/ns/core#SymmetricImpact

SymmetricImpact is an RDF individual.

The property represents a dependency from both subject to object and object to subject.

Zero-or-many

http://open-services.net/ns/core#Zero-or-many

Zero-or-many is an RDF individual.

Property is optional and multi-valued.

Zero-or-one

http://open-services.net/ns/core#Zero-or-one

Zero-or-one is an RDF individual.

Property is optional and single valued.

8. Conformance

OSLC servers **MUST** use the vocabulary terms defined here where required, and with the meanings defined here.

OSLC servers **MAY** augment this vocabulary with additional classes, properties, and individuals.

Clause Number	Requirement
cc-1	Archived Resources **MAY** be identified by having a property `oslc:archived`, with value `true`.
cc-2	Archived Resources **MAY** be removed from typical user interactions.
cc-3	Archived Resources **SHOULD** be considered immutable.

OASIS OPEN PROJECTS

OSLC Core Version 3.0. Part 8: Constraints

OASIS Standard
26 August 2021

This stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.pdf

Previous stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/core-shapes.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps02/core-shapes.pdf
(published as Project Specification)

Latest stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/core-shapes.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/core-shapes.pdf

Latest version:

https://open-services.net/spec/core/latest

Latest editor's draft:

https://open-services.net/spec/core/latest-draft

Open Project:

OASIS Open Services for Lifecycle Collaboration (OSLC) OP

Project Chairs:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Editors:

Jim Amsden (jamsden@us.ibm.com), IBM
Andrii Berezovskyi (andriib@kth.se), KTH

Additional components:

This specification is one component of a Work Product that also includes:

- *OSLC Core Version 3.0. Part 1: Overview.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/oslc-core.html
- *OSLC Core Version 3.0. Part 2: Discovery*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/discovery.html
- *OSLC Core Version 3.0. Part 3: Resource Preview.* https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-preview.html
- *OSLC Core Version 3.0. Part 4: Delegated Dialogs*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/dialogs.html
- *OSLC Core Version 3.0. Part 5: Attachments*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/attachments.html
- *OSLC Core Version 3.0. Part 6: Resource Shape*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/resource-shape.html
- *OSLC Core Version 3.0. Part 7: Vocabulary*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.html

- *OSLC Core Version 3.0. Part 8: Constraints (this document)*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.html
- *OSLC Core Version 3.0. Machine Readable Vocabulary Terms*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-vocab.ttl
- *OSLC Core Version 3.0. Machine Readable Constraints*. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.ttl

Related work:

This specification is related to:

- *OSLC Core Version 3.0: Link Guidance*. https://oslc-op.github.io/oslc-specs/notes/link-guidance.html

RDF Namespaces:

http://open-services.net/ns/core#

Abstract:

Core Vocabulary defines the OSLC Core RDF vocabulary terms and resources, that have broad applicability across various domains. This document specifies the standard constraints on those vocabulary terms using OSLC ResourceShapes.

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the “Latest stage” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Open Project are listed at https://github.com/oslc-op/oslc-specs.

Comments on this work can be provided by opening issues in the project repository or by sending email to the project’s public comment list oslc-op.

The English version of this specification is the only normative version. Non-normative translations may also be available. Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:
[OSLC-CoreShapes-3.0]
OSLC Core Version 3.0. Part 8: Constraints. Edited by Jim Amsden and Andrii Berezovskyi. 26 August 2021. OASIS Standard. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/os/core-shapes.html. Latest stage: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/core-shapes.html.

Notices

Table of Contents

1. Introduction

This section is non-normative.

RDF vocabularies define the terms and resources for a domain of interest, life-cycle management in the case of OSLC Core. These vocabularies are often specified in an open manner, without providing information such as property domain and range assertions, cardinalities, etc. This helps keep the vocabulary applicable for a wide range of uses and furthering integration with other vocabularies.

However, it is often desirable to closed down a vocabulary with specific constraints to facilitate using the vocabulary for a specific purpose. This document specifies the constraints for using the OSLC Core vocabulary in OSLC. Different sets of constraints **MAY** be applied to a vocabulary in order to tailor its use, without overly constraining the vocabulary for other usages. [cc-1]

These constraints apply to the core vocabulary defined in OSLC Core Version 3.0. Part 7: Vocabulary.

1.1 Terminology

Terminology uses and extends the terminology and capabilities of OSLC Core Overview, W3C Linked Data Platform [LDP], W3C's Architecture of the World Wide Web [WEBARCH], Hyper-text Transfer Protocol [HTTP11].

No newterms are defined in this part.

1.2 References

1.2.1 Normative references

[DC-TERMS]

DCMI Usage Board. *Dublin Core Metadata Terms, version 1.1*. DCMI, 11 October 2010. DCMI Recommendation. URL: http://dublincore.org/documents/2010/10/11/dcmi-terms/

[FOAF]

Dan Brickley; Libby Miller. *FOAF Vocabulary Specification 0.99 (Paddington Edition)*. FOAF project, 14 January 2014. URL: http://xmlns.com/foaf/spec

[HTTP11]

R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. IETF, June 2014. Proposed Standard. URL: https://httpwg.org/specs/rfc7230.html

[LDP]

Steve Speicher; John Arwe; Ashok Malhotra. *Linked Data Platform 1.0*. W3C, 26 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/ldp/

[OSLCCore2]

S. Speicher; D. Johnson. *OSLC Core 2.0*. http://open-services.net. Finalized. URL: http://open-services.net/bin/view/Main/OslcCoreSpecification

[RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF, March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

[RFC8174]

B. Leiba. *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*. IETF, May 2017. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc8174

[rdf-schema]

Dan Brickley; Ramanathan Guha. *RDF Schema 1.1*. W3C, 25 February 2014. W3C Recommendation. URL:

https://www.w3.org/TR/rdf-schema/

[rdf11-concepts]

Richard Cyganiak; David Wood; Markus Lanthaler. *RDF 1.1 Concepts and Abstract Syntax*. W3C, 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/rdf11-concepts/

1.2.2 Informative references

[CSS21]

Bert Bos; Tantek Çelik; Ian Hickson; Håkon Wium Lie. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C, 7 June 2011. W3C Recommendation. URL: https://www.w3.org/TR/CSS21/

[SHACL]

Holger Knublauch; Arthur Ryman. *Shapes Constraint Language (SHACL)*. http://www.w3.org/. Draft. URL: https://w3c.github.io/data-shapes/shacl/

[WEBARCH]

Ian Jacobs; Norman Walsh. *Architecture of the World Wide Web, Volume One*. W3C, 15 December 2004. W3C Recommendation. URL: https://www.w3.org/TR/webarch/

[skos-reference]

Alistair Miles; Sean Bechhofer. *SKOS Simple Knowledge Organization System Reference*. W3C, 18 August 2009. W3C Recommendation. URL: https://www.w3.org/TR/skos-reference/

1.3 Typographical Conventions and Use of RFC Terms

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this specification are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Common Properties

Unlike the rest of the Core specification, these properties change and grow as new common properties are added. The properties that we list here are available for use in OSLC domain specifications when defining OSLC resources, but this does not mean that they are required to be in OSLC resources. OSLC domain specifications decide which properties are allowed and required for resources needed to realize their use cases. The OSLC common properties include properties defined in other standard vocabularies including:

- Friend of a Friend (FOAF)
- Dublin Core (dcterms)
- RDF Schema (rdfs)

2.1 Properties on Any Resource

- **Describes:** `Common Properties`
- **Summary:** Defines common properties that are be applicable to any OSLC resource. OSLC domains **SHOULD** use these properties where applicable rather than defining their own properties [cc-2]. The cardinality, representations, ranges, and other columns of the following table indicate typical usage. However, a domain **MAY** apply its own constraints for particular resource shapes [cc-3].

Common Properties Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:contributor`	Zero-or-many	unspecified	AnyResource	Either	`oslc:Any`, `foaf:Person`	Contributor or contributors to the resource. It is likely that the target resource will be a foaf:Person but that is not necessarily the case.
`dcterms:created`	Zero-or-one	unspecified	dateTime	N/A	Unspecified	Timestamp of resource creation.
`dcterms:creator`	Zero-or-many	unspecified	AnyResource	Either	`oslc:Any`, `foaf:Person`	Creator or creators of the resource. It is likely that the target resource will be a foaf:Person but that is not necessarily the case.
`dcterms:description`	Zero-or-many	unspecified	XMLLiteral	N/A	Unspecified	Descriptive text about resource represented as rich text in XHTML content.
`dcterms:identifier`	Zero-or-many	unspecified	string	N/A	Unspecified	A unique identifier for a resource. Typically read-only and assigned by the service provider when a resource is created. Not typically intended for end-user display.
`dcterms:modified`	Zero-or-many	unspecified	dateTime	N/A	Unspecified	Timestamp of latest resource modification.
`dcterms:references`	Zero-or-many	unspecified	AnyResource	Either	Unspecified	A related resource that is referenced, cited, or otherwise pointed to by the described resource.

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
dcterms:relation	Zero-or-many	unspecified	AnyResource	Either	Unspecified	Relation which identifies a related resource.
dcterms:subject	Zero-or-many	unspecified	string	N/A	Unspecified	Tag or keyword for a resource. Each occurrence of a dcterms:subject property denotes an additional tag for the resource.
dcterms:title	Zero-or-many	unspecified	XMLLiteral	N/A	Unspecified	Title of the resource represented as rich text in XHTML content.
oslc:archived	Zero-or-one	unspecified	boolean	N/A	Unspecified	Indicates whether the subject has been marked as archived, no longer an actively updating resource.
oslc:discussedBy	Zero-or-one	unspecified	Resource	Either	oslc:Discussion	A series of notes and comments about this resource.
oslc:error	Zero-or-many	unspecified	AnyResource	Either	Unspecified	A series of errors associated with this resource.
oslc:instanceShape	Zero-or-many	unspecified	Resource	Reference	oslc:ResourceShape	The URI of a Resource Shape that describes the possible properties, occurrence, value types, allowed values and labels. This shape information is useful in displaying the subject resource as well as guiding clients in performing modifications. Instance shapes may be specific to the authenticated user associated with the request that retrieved the resource, the current state of the resource and other factors and thus should not be cached.
oslc:modifiedBy	Zero-or-many	unspecified	Resource	Either	oslc:Any, foaf:Person	The URI of a resource describing the entity that most recently modified the subject resource. The link target is usually a foaf:Person or foaf:Agent, but could be any type. This is modeled after dcterms:creator, but Dublin Core currently has no equivalent property.

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
`oslc:queryable`	Zero-or-one	unspecified	boolean	N/A	Unspecified	Indicates whether a property is queryable (can appear in oslc.where and olsc.select clause) or not. Defaults to true if unspecified.
`oslc:serviceProvider`	Zero-or-many	unspecified	Resource	Reference	`oslc:ServiceProvider`	A link to the resource's OSLC Service Provider. There may be cases when the subject resource is available from a service provider that implements multiple domain specifications, which could result in multiple values for this property.
`oslc:shortId`	Zero-or-many	unspecified	string	N/A	Unspecified	A short, human-readable, plain text value. This value should be unique in some context that is apparent to human users of a service.
`oslc:shortTitle`	Zero-or-many	unspecified	XMLLiteral	N/A	Unspecified	Shorter form of dcterms:title for the resource represented as rich text in XHTML content.
`rdf:type`	Zero-or-many	unspecified	Resource	Reference	`rdfs:Class`	The resource type URIs.
`rdfs:member`	Zero-or-many	unspecified	Resource	Either	Unspecified	OSLC domains might define a number of member or contains relationships between resources. The rdfs:member property is suitable for use when only one such relationship needs to be defined, or when no additional semantics need to be implied by the property name.

2.2 Person Properties

- **Describes:** `http://xmlns.com/foaf/0.1/Person`
- **Summary:** Person is a resource defined by FOAF that is used as the value for a dcterms:creator or dcterms:contributor property. This shape specifies the recommended minimal FOAF Person properties that should be provided for OSLC.

Person Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
`foaf:familyName`	Zero-or-many	unspecified	string	N/A	Unspecified	Family name of person expressed as simple text string.
`foaf:givenName`	Zero-or-many	unspecified	string	N/A	Unspecified	Given name of person expressed as simple text string.
`foaf:mbox`	Zero-or-many	unspecified	string	N/A	Unspecified	A personal mailbox for this person, typically identified using the mailto: URI scheme (see RFC 2368).
`foaf:name`	Zero-or-many	unspecified	string	N/A	Unspecified	The full name of a person expressed as simple text string.
`foaf:nick`	Zero-or-many	unspecified	string	N/A	Unspecified	A short informal nickname or login identifer expressed as simple text string.

2.3 Implementation Conformance

Changes to the OSLC Core Vocabulary **MUST** be approved by the OASIS OSLC Open Project. [cc-4] The OSLC Core Vocabulary is assigned the namespace URI of the `http://open-services.net/ns/core#`.

Domain TCs and other extensions **MUST** contribute their vocabulary terms in a namespace which is assigned to them as an authority. [cc-5]

OSLC Core, domain and other extensions **SHOULD** reuse existing vocabulary terms from stable vocabularies such as [DC-TERMS], RDF [rdf11-concepts], RDF Schema [rdf-schema], [FOAF], [skos-reference] and OSLC. [cc-6] New vocabulary terms **SHOULD** only be created when there is no clear existing choice available. [cc-7] See the [LDP] similar clause on reuse.

3. Discussion

3.1 Shape: Discussion

It is common to collect a series of comments on a lifecycle resource, often referred to as a discussion. For example: tasks, bug reports, requirements, assets and so on, are often collected across various types of resources such as project. A project might reflect the planning of work to deliver a product that realizes the requirements as validated through test cases and bug reports. Discussions allow users to collaborate with each other for more efficient and effective delivery. This Discussion resource definition provides a minimal shape describing the needed properties.

- **Describes:** `http://open-services.net/ns/core#Discussion`
- **Summary:** OSLC Core Discussion Shape

Discussion Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`oslc:comment`	Zero-or-many	false	AnyResource	Either	`oslc:Comment`	Comment about resource.
`oslc:discussionAbout`	Exactly-one	false	Resource	Reference	Unspecified	Reference to associated resource.

3.2 Shape: Comment

Used in conjunction with Shape: Discussion to provide a minimal resource definition for a collection of comments.

- **Describes:** `http://open-services.net/ns/core#Comment`
- **Summary:** OSLC Core Comment Shape

Comment Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:created`	Exactly-one	unspecified	dateTime	N/A	Unspecified	When the comment resource was created.
`dcterms:creator`	Exactly-one	unspecified	AnyResource	Either	`foaf:Person`	The person who created the comment.
`dcterms:description`	Exactly-one	unspecified	XMLLiteral	N/A	Unspecified	Details or body of the comment; **SHOULD** include only content that is valid and suitable inside an XHTML `<div>` element [cc-8].
`dcterms:identifier`	Exactly-one	unspecified	string	N/A	Unspecified	A service defined identifier.
`dcterms:title`	Zero-or-one	unspecified	XMLLiteral	N/A	Unspecified	A brief title for the comment; **SHOULD** include only content that is valid and suitable inside an XHTML `<span>` element [cc-9].
`oslc:inReplyTo`	Zero-or-one	unspecified	Resource	Reference	`oslc:Comment`	Reference to the comment to which this comment replies.

4. Errors

4.1 Implementation Conformance

When an OSLC Server incurs an error, it is **RECOMMENDED** that useful information be provided to clients in the body of the HTTP response. [cc-10]

OSLC Servers **SHOULD** use the Error resource defined below as the basis for forming error responses. [cc-11]

OSLC Servers **SHOULD** return an Error resource using the same representation format requested by the client via the HTTP `Accept` request header. [HTTP11] [cc-12]

OSLC Clients **SHOULD** treat the `oslc:statusCode` as a String that starts with digits, but **MAY** contain non-digit text. [cc-13]

4.2 Shape: Error

Used when servers need a consistent shape to communicate error messages.

- **Describes:** `http://open-services.net/ns/core#Error`
- **Summary:** OSLC Core Error Shape

Error Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
`dcterms:created`	Zero-or-one	unspecified	dateTime	N/A	Unspecified	Optional indication of when the error was detected.
`dcterms:identifier`	Zero-or-many	unspecified	string	N/A	Unspecified	A unique human-readable string identifier for this resource, such as an error number or code.
`dcterms:references`	Zero-or-many	unspecified	AnyResource	Either	Unspecified	A reference to any resources that are the subject of this error.
`oslc:cause`	Zero-or-many	true	AnyResource	Either	`oslc:Error`	An error that was the cause of this error.
`oslc:extendedError`	Zero-or-one	true	AnyResource	Either	`oslc:ExtendedError`	Extended error information.
`oslc:message`	Exactly-one	true	string	N/A	Unspecified	An informative message describing the error that occurred.
`oslc:statusCode`	Exactly-one	true	string	N/A	Unspecified	The HTTP status code reported with the error.

4.3 Shape: ExtendedError

Additional details about an error the server had when processing the request.

- **Describes:** `http://open-services.net/ns/core#ExtendedError`
- **Summary:** OSLC Core ExtendedError Shape

ExtendedError Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:hintHeight	Zero-or-one	true	string	N/A	Unspecified	Values MUST be expressed in relative length units as defined in the W3C Cascading Style Sheets Specification (CSS 2.1) Em and ex units are interpreted relative to the default system font (at 100% size) [cc-14].
oslc:hintWidth	Zero-or-one	true	string	N/A	Unspecified	Values MUST be expressed in relative length units as defined in the W3C Cascading Style Sheets Specification (CSS 2.1) Em and ex units are interpreted relative to the default system font (at 100% size) [cc-15].
oslc:moreInfo	Zero-or-one	true	Resource	Reference	Unspecified	A resource giving more information on the error SHOULD be of an HTML content-type [cc-16].
oslc:rel	Zero-or-one	true	string	N/A	Unspecified	If present and set to 'alternate' then indicates that work-around is provided, behavior for other values is undefined.

4.4 Shape: ResponseInfo

Resource representations returned via [OSLCCore2] Resource Paging MUST include a resource of type oslc:ResponseInfo, as defined in this section. [cc-17] A response info resource representation describes information about a paged HTTP response body in which it appears.

- **Describes:** `http://open-services.net/ns/core#ResponseInfo`
- **Summary:** The shape of a resource providing information about a paged HTTP response body.

ResponseInfo Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
dcterms:description	Zero-or-one	unspecified	XMLLiteral	N/A	Unspecified	Descriptive text about resource represented as rich text in XHTML content.
dcterms:title	Zero-or-one	unspecified	XMLLiteral	N/A	Unspecified	Title of the resource represented as rich text in XHTML content.
oslc:nextPage	Zero-or-one	true	Resource	Reference	Unspecified	Link to the next page of a response.
oslc:postBody	Zero-or-one	true	string	N/A	Unspecified	The body of a POST request to return the next page if the response was to a POST request. Where a paged resource supports POST with an application/x-www-form-urlencoded body as an alternative to GET to avoid the request URI exceeding server limitations, the oslc:ResponseInfo in the response to the POST SHOULD contain this property so that a client knows what to POST to get the next page [cc-18].

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:totalCount	Zero-or-one	true	integer	N/A	Unspecified	This optional property indicates the total number of results across all pages, its value SHOULD be non-negative [cc-19]. In the context of a query resource, this value SHOULD be the total number of results, i.e. the number of resources that match the query [cc-20]. In the context of other resources, the value SHOULD be the total number of property values (i.e. RDF triples) of the resource [cc-21]. Unless Stable Paging is in effect, the total count MAY vary as a client retrieves subsequent pages [cc-22].

5. Resource Shape

The shape of an RDF resource is a description of the set of triples it is expected to contain and the integrity constraints those triples are required to satisfy. Applications of shapes include validating RDF data, documenting RDF APIs, and providing meta-data to tools, such as form and query builders, that handle RDF data. OSLC Core uses shapes to:

- Define specific vocabulary constraints including allowed values, max size, cardinality, representation in RDF specifications and if the property is read only.
- Specify the properties required for resource creation.
- Specify what servers all allow for prefilling delegated dialogs.
- Describe the results of query operations.

Constraints on OSLC Core and Domain resources **SHOULD** be described using ResourceShapes which is included as part of the OSLC Core multi-part specifications. [cc-23] Servers **MAY** use other constraint languages such as [SHACL] to define resource constraints. [cc-24]

ResourceShape Constraints

- **Describes:** `http://open-services.net/ns/core#ResourceShape`
- **Summary:** A shape resource describes the contents of and constraints on some set of described resources.
- **Description:** A resource should satisfy all the constraints defined by its applicable shapes.

ResourceShape Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:description`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	The description of the defined constraint.
`dcterms:title`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	The summary of this shape.
`oslc:describes`	Zero-or-many	true	Resource	Reference	`rdfs:Class`	The described resource types that this shape applies to.
`oslc:hidden`	Zero-or-one	true	boolean	N/A	Unspecified	Indicates the resource or property should not be displayed to users.
`oslc:property`	Zero-or-many	true	Resource	Inline	`oslc:Property`	Indicates an expected property of the described resources.
`rdf:type`	Zero-or-many	true	Resource	Reference	Unspecified	An OSLC resource shape **SHOULD** have an RDF type of oslc:ResourceShape.

Property Constraints

- **Describes:** `http://open-services.net/ns/core#Property`
- **Summary:** Specifies the name, description, summary, occurrence, value type, allowed values, and several other aspects of the defined property.

Property Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:description`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	The description of the defined constraint.
`dcterms:title`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	The summary of the defined property.

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:allowedValue	Zero-or-many	true	unspecified	Either	Unspecified	Specifies the allowed values of a property.
oslc:allowedValues	Zero-or-one	true	Resource	Reference	oslc:AllowedValues	The resource containing a set of allowed values of the defined property.
oslc:defaultValue	Zero-or-one	true	unspecified	Either	Unspecified	The default value of the defined property.
oslc:hidden	Zero-or-one	true	boolean	N/A	Unspecified	Indicates the resource or property should not be displayed to users.
oslc:isMemberProperty	Zero-or-one	true	boolean	N/A	Unspecified	If true then the described resource is a container and the defined property is used for container membership.
oslc:maxSize	Zero-or-one	true	integer	N/A	Unspecified	For string datatype properties, the maximum number of characters.
oslc:name	Exactly-one	true	string	N/A	Unspecified	The local name of the defined property.
oslc:occurs	Exactly-one	true	Resource	Reference	oslc:Cardinality	The number of times the defined property may occur.
oslc:propertyDefinition	Exactly-one	true	Resource	Reference	rdf:Property	The URI of the defined or constrained property.
oslc:queryable	Zero-or-one	true	boolean	N/A	Unspecified	Indicates whether a property is queryable (can appear in oslc.where and olsc.select clause) or not.
oslc:range	One-or-many	true	Resource	Reference	rdfs:Class	For object properties, specifies what the target resource type is expected to be, but that is not necessarily the case.
oslc:readOnly	Zero-or-one	true	boolean	N/A	Unspecified	If true then the defined property cannot be directly writen by clients, but may be updated indirectly by servers.
oslc:representation	Zero-or-one	true	Resource	Reference	oslc:Representation	For object properties, how the object resource is represented in the representation of the described resource.
oslc:valueShape	Zero-or-one	true	Resource	Reference	oslc:ResourceShape	For object properties, the URI of a shape resource that describes the object resource.
oslc:valueType	Zero-or-many	true	Resource	Reference	oslc:ResourceValueType	The type of values of the defined property.

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
`rdf:type`	Zero-or-many	true	Resource	Reference	Unspecified	An OSLC property **SHOULD** have an RDF type of oslc:Property.

AllowedValues Constraints

- **Describes:** `http://open-services.net/ns/core#AllowedValues`
- **Summary:** Defines a set of allowed values for a defined property.

AllowedValues Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
`oslc:allowedValue`	One-or-many	true	unspecified	Either	Unspecified	Specifies the allow values in an AllowedValue constraint.

6. Discovery constraints

6.1 Resource: ServiceProviderCatalog

- **Describes:** `http://open-services.net/ns/core#ServiceProviderCatalog`
- **Summary:** Service Provider Catalog
- **Description:** An LDPC describing an OSLC server that offers one or more ServiceProvider LDPCs. Servers MAY also organize the ServiceProviders in one or more ServiceProviderCatalog LDPCs to enable OSLC clients to find ServiceProviders offered [cc-25]. The members of these catalogs may include other nested catalogs as well as service providers.

ServiceProviderCatalog Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:description`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	Description of the services provided.
`dcterms:publisher`	Zero-or-one	true	AnyResource	Inline	`oslc:Publisher`	Describes the software product that provides the implementation.
`dcterms:title`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	Title of this resource.
`oslc:domain`	Zero-or-many	true	Resource	Reference	Unspecified	Namespace URI of the specification that is implemented by this service. In most cases this namespace URI will be for an OSLC domain, but other URIs MAY be used [cc-26].
`oslc:oauthConfiguration`	Zero-or-many	true	AnyResource	Inline	`oslc:OAuthConfiguration`	Defines the three OAuth URIs required for a client to act as an OAuth consumer.
`oslc:serviceProvider`	Zero-or-many	true	AnyResource	Either	`oslc:ServiceProvider`	A service provider LDPC offered by this server.
`oslc:serviceProviderCatalog`	Zero-or-many	true	AnyResource	Either	`oslc:ServiceProviderCatalog`	Additional service provider catalog LDPCs used to organize services.

6.2 Resource: ServiceProvider

- **Describes:** `http://open-services.net/ns/core#ServiceProvider`
- **Summary:** Service Provider
- **Description:** An LDPC whose members are the Service LDPCs offered by an OSLC server.

ServiceProvider Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:description`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	Description of the services provided.
`dcterms:publisher`	Zero-or-one	true	AnyResource	Inline	`oslc:Publisher`	Describes the software product that provides the implementation.
`dcterms:title`	Zero-or-one	true	XMLLiteral	N/A	Unspecified	Title of this resource.
`oslc:details`	Zero-or-many	true	Resource	Reference	Unspecified	A URL that may be used to retrieve a resource to determine additional details about the service provider such as a web page describing it.
`oslc:oauthConfiguration`	Zero-or-many	true	AnyResource	Inline	`oslc:OAuthConfiguration`	Defines the three OAuth URIs required for a client to act as an OAuth consumer.
`oslc:prefixDefinition`	Zero-or-many	true	AnyResource	Inline	`oslc:PrefixDefinition`	Defines a namespace prefix for use in JSON representations and in forming OSLC Query Syntax strings.
`oslc:service`	One-or-many	true	AnyResource	Inline	`oslc:Service`	Describes a service LDPC offered by the service provider.

6.3 Resource: Service

- **Describes:** `http://open-services.net/ns/core#Service`
- **Summary:** Service
- **Description:** An LDPC whose properties describe specific services offered by a server, and the URIs to use for those services in the context of that ServiceProvider.

Service Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`oslc:creationDialog`	Zero-or-many	true	AnyResource	Inline	`oslc:Dialog`	Enables clients to create a resource via UI.
`oslc:creationFactory`	Zero-or-many	true	AnyResource	Inline	`oslc:CreationFactory`	An LDPC that enables clients to create new resources.

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:domain	Exactly-one	true	Resource	Reference	Unspecified	Namespace URI of the specification that is implemented by this service. In most cases this namespace URI will be for an OSLC domain, but other URIs MAY be used [cc-27].
oslc:queryCapability	Zero-or-many	true	AnyResource	Inline	oslc:QueryCapability	Enables clients query across a collection of resources.
oslc:selectionDialog	Zero-or-many	true	AnyResource	Inline	oslc:Dialog	Enables clients to select a resource via UI.
oslc:usage	Zero-or-many	true	Resource	Reference	Unspecified	An identifier URI for the domain specified usage of this resource. If a resource has multiple uses, it may designate the primary or default one that should be used with a property value of oslc:default.

6.4 Resource: CreationFactory

- **Describes:** http://open-services.net/ns/core#CreationFactory
- **Summary:** Creation Factory
- **Description:** A Creation Factory describes a capability for creating resources, including an LDPC capable of creating and containing new resources via HTTP POST.

CreationFactory Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
dcterms:title	Exactly-one	true	XMLLiteral	N/A	Unspecified	Title string that could be used for display.
oslc:creation	Exactly-one	true	Resource	Reference	ldp:Container	To create a new resource via the factory, post it to this URI.
oslc:label	Zero-or-one	true	string	N/A	Unspecified	Very short label for use in menu items.
oslc:resourceShape	Zero-or-many	true	Resource	Reference	oslc:ResourceShape	A Creation Factory MAY provide Resource Shapes that describe shapes of resources that may be created [cc-28].
oslc:resourceType	Zero-or-many	true	Resource	Reference	rdfs:Class	The expected resource type URI of the resource that will be created using this creation factory. These would be the URIs found in the result resource's rdf:type property.
oslc:usage	Zero-or-many	true	Resource	Reference	Unspecified	An identifier URI for the domain specified usage of this resource. If a resource has multiple uses, it may designate the primary or default one that should be used with a property value of oslc:default.

6.5 Resource: QueryCapability

- **Describes:** `http://open-services.net/ns/core#QueryCapability`
- **Summary:** Query Capability
- **Description:** A Query Capability describes a query capability, capable of querying resources via HTTP GET or POST.

QueryCapability Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:title`	Exactly-one	true	XMLLiteral	N/A	Unspecified	Title string that could be used for display.
`oslc:label`	Zero-or-one	true	string	N/A	Unspecified	Very short label for use in menu items.
`oslc:queryBase`	Exactly-one	true	Resource	Reference	Unspecified	The base URI to use for queries. Queries are invoked via HTTP GET on a query URI formed by appending a key=value pair to the base URI, as described in Query Capabilities section.
`oslc:resourceShape`	Zero-or-one	true	Resource	Reference	`oslc:ResourceShape`	The Query Capability **SHOULD** provide a Resource Shape that describes the query base URI [cc-29].
`oslc:resourceType`	Zero-or-many	true	Resource	Reference	`rdfs:Class`	The expected resource type URI that will be returned with this query capability. These would be the URIs found in the result resource's rdf:type property.
`oslc:usage`	Zero-or-many	true	Resource	Reference	Unspecified	An identifier URI for the domain specified usage of this query capability. If a service provides multiple query capabilities, it may designate the primary or default one that should be used with a property value of oslc:default.

6.6 Resource: Publisher

- **Describes:** `http://open-services.net/ns/core#Publisher`
- **Summary:** Publisher
- **Description:** A Publisher identifies and describes the software product that provides the OSLC implementation.

Publisher Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:identifier`	Exactly-one	unspecified	string	N/A	Unspecified	A URN that uniquely identifies the implementation.
`dcterms:title`	Exactly-one	true	XMLLiteral	N/A	Unspecified	Title string that could be used for display.
`oslc:icon`	Zero-or-one	true	Resource	Reference	Unspecified	URL to an icon file that represents the provider. This icon should be a favicon format and 16x16 pixels in size.
`oslc:label`	Zero-or-one	true	string	N/A	Unspecified	Very short label for use in menu items.

6.7 Resource: PrefixDefinition

- **Describes:** `http://open-services.net/ns/core#PrefixDefinition`
- **Summary:** Prefix Definition

- **Description:** Service Providers **MUST** provide a Prefix Definition for each prefix supported by the service [cc-30]. Each Prefix Definition defines a namespace prefix that clients **MAY** use in forming OSLC Query Syntax strings [cc-31].

PrefixDefinition Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
`oslc:prefix`	Exactly-one	true	string	N/A	Unspecified	Namespace prefix to be used for this namespace.
`oslc:prefixBase`	Exactly-one	true	Resource	Reference	Unspecified	The base URI of the namespace.

6.8 Resource: OAuthConfiguration

- **Describes:** `http://open-services.net/ns/core#OAuthConfiguration`
- **Summary:** OAuth Configuration
- **Description:** Service Providers that support OAuth Authentication **SHOULD** provide a way for clients to automatically discover the three OAuth URIs necessary to act as an OAuth Consumer [cc-32].

OAuthConfiguration Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
`oslc:authorizationURI`	Exactly-one	true	Resource	Reference	Unspecified	URI for obtaining OAuth authorization.
`oslc:oauthAccessTokenURI`	Exactly-one	true	Resource	Reference	Unspecified	URI for obtaining OAuth access token.
`oslc:oauthRequestTokenURI`	Exactly-one	true	Resource	Reference	Unspecified	URI for obtaining OAuth request token.

7. Resource Preview Constraints

7.1 Resource: Compact

- **Describes:** `http://open-services.net/ns/core#Compact`
- **Summary:** Describes how to display a resource preview.

Compact Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:title`	Zero-or-one	true	string	N/A	Unspecified	Title that may be used in the display of a link to the resource. The value should include only content that is valid inside an HTML element. Providers should include a dcterms:title property with an informative label for the resource. The title is typically shown to a user as a hyperlink. For a resource with no obvious title, Providers should omit the dcterms:title property. Providers must first HTML escape the contents of the dcterms:title before sending the response.
`oslc:icon`	Zero-or-one	true	Resource	Reference	Unspecified	URI of an image which may be used in the display of a link to the resource.
`oslc:iconAltLabel`	Zero-or-one	true	string	N/A	Unspecified	Alternative label used in association with the oslc:icon, such as HTML img tag's alt attribute.
`oslc:iconSrcSet`	Zero-or-one	true	string	N/A	Unspecified	Specification of a set of images of different sizes based on HTML img element srcset attribute.
`oslc:iconTitle`	Zero-or-one	true	string	N/A	Unspecified	Title used in association with the oslc:icon, such as HTML img tag's title attribute.
`oslc:largePreview`	Zero-or-one	true	AnyResource	Either	`oslc:Preview`	URI and sizing properties for an HTML document to be used for a large preview.
`oslc:shortTitle`	Zero-or-one	true	string	N/A	Unspecified	Abbreviated title which may be used in the display of a link to the resource. The value should include only content that is valid inside an HTML element. Providers should include an abbreviated title for the resource when possible. The abbreviated title is typically shown to a user as a hyperlink in presentations where visual space is limited. As a general guideline, the length of the abbreviated title should be 5 characters or less. A user-visible identifier that ordinarily appears in the dcterms:title, such as a defect number, makes for a good oslc:shortTitle value. When a resource has no obvious identifier or handle, Providers should omit the oslc:shortTitle property. Providers must first HTML escape the contents of the oslc:shortTitle before sending the response.

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:smallPreview	Zero-or-one	true	AnyResource	Either	oslc:Preview	URI and sizing properties for an HTML document to be used for a small preview.

7.2 Resource: Preview

- **Describes:** http://open-services.net/ns/core#Preview
- **Summary:** An HTML representation of a resource that can be embedded in another user interface.

Preview Properties

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
oslc:document	Exactly-one	true	Resource	Reference	Unspecified	The URI of an HTML document to be used for the preview.
oslc:hintHeight	Zero-or-one	true	string	N/A	Unspecified	Recommended height of the preview. Values are expressed using length units as specified in [CSS21].
oslc:hintWidth	Zero-or-one	true	string	N/A	Unspecified	Recommended width of the preview. Values are expressed using length units as specified in [CSS21].

8. Delegated Dialogs Constraints

- **Describes:** `http://open-services.net/ns/core#Dialog`
- **Summary:** Describes information about a dialog such as its title and dimensions.

Dialog Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:title`	Exactly-one	true	XMLLiteral	N/A	Unspecified	Title string that could be used for display.
`oslc:dialog`	Exactly-one	true	unspecified	Either	Unspecified	The URI of the dialog.
`oslc:hintHeight`	Zero-or-one	true	unspecified	Either	Unspecified	Recommended height of the dialog. Values are expressed using length units as specified in [CSS21].
`oslc:hintWidth`	Zero-or-one	true	unspecified	Either	Unspecified	Recommended width of the dialog. Values are expressed using length units as specified in [CSS21].
`oslc:label`	Zero-or-one	true	string	N/A	Unspecified	Very short label for use in menu items.
`oslc:resourceShape`	Zero-or-many	true	Resource	Reference	`oslc:ResourceShape`	Describes constraints on dialog prefill requests.
`oslc:resourceType`	Zero-or-many	true	Resource	Reference	`rdfs:Class`	The expected resource type URI for the resources that will be returned when using this dialog. These would be the URIs found in the result resource's `rdf:type` property.
`oslc:usage`	Zero-or-many	true	Resource	Reference	Unspecified	An identifier URI for the domain specified usage of this dialog. If a resource has multiple uses, it may designate the primary or default one that should be used with a property value of oslc:default.

9. Resource Constraints

9.1 Resource: AttachmentDescriptor

The `oslc:AttachmentDescriptor` resource type is used to describe the binary resource (or non-RDF Resource) associated with a particular resource. When a client POSTs an attachment content to a server, the server stores the attachment content and assigns a URI just like any other type of resource creation but it may also create an `oslc:AttachmentDescriptor` resource to contain data about the attachment.

There is no restriction on the content of each attachment resource. For example, it could be a photo of a kitten, an installation manual, a log file, or a source code patch. Since the attachment cannot be expected to contain additional client or server supplied data, a typical set of properties for each attachment is included with the `oslc:AttachmentDescriptor` resource itself. Thus, the object of each `oslc:attachment` statement is the binary attachment. Issuing an HTTP HEAD or GET operation on that binary attachment resource URL should produce an HTTP response with a header value of `Link: rel='describedBy'` to indicate the URL of the `oslc:AttachmentDescriptor` resource. The properties for the `oslc:AttachmentDescriptor` resource are indicated in the table below.

- **Describes:** `http://open-services.net/ns/core#AttachmentDescriptor`
- **Summary:** LDP-RS to contain data about a LDP-NR(Attachment)

AttachmentDescriptor Properties

Prefixed Name	*Occurs*	*Read-only*	*Value-type*	*Representation*	*Range*	*Description*
`dcterms:created`	Zero-or-one	true	dateTime	N/A	Unspecified	Timestamp of attachment creation.
`dcterms:creator`	Zero-or-many	true	AnyResource	Either	Unspecified	Creator or creators of the attachment. Likely a `foaf:Person`, but not necessarily so.
`dcterms:description`	Zero-or-one	false	XMLLiteral	N/A	Unspecified	Descriptive text about the attachment.
`dcterms:format`	Zero-or-one	true	unspecified	Either	Unspecified	MIME type of the attachment content; **SHOULD** be a PURL media-type resource [cc-33].
`dcterms:identifier`	Zero-or-one	true	string	N/A	Unspecified	System-assigned identifier.
`dcterms:title`	Zero-or-one	false	string	N/A	Unspecified	Client-specified file name or title.
`oslc:attachmentSize`	Zero-or-one	true	integer	N/A	Unspecified	Size in bytes of the attachment content.

10. Conformance

OSLC servers **MUST** follow the constraints defined here where required, and with the meanings defined here. [cc-34]

OSLC servers **MAY** provide additional constraints for specific purposes. [cc-35]

Clause Number	Requirement
cc-1	Different sets of constraints **MAY** be applied to a vocabulary in order to tailor its use, without overly constraining the vocabulary for other usages.
cc-2	OSLC domains **SHOULD** use these properties where applicable rather than defining their own properties
cc-3	However, a domain **MAY** apply its own constraints for particular resource shapes
cc-4	Changes to the OSLC Core Vocabulary **MUST** be approved by the OASIS OSLC Open Project.
cc-5	Domain TCs and other extensions **MUST** contribute their vocabulary terms in a namespace which is assigned to them as an authority.
cc-6	OSLC Core, domain and other extensions **SHOULD** reuse existing vocabulary terms from stable vocabularies such as [DC-TERMS], RDF [rdf11-concepts], RDF Schema [rdf-schema], [FOAF], [skos-reference] and OSLC.
cc-7	New vocabulary terms **SHOULD** only be created when there is no clear existing choice available.
cc-8	Details or body of the comment; **SHOULD** include only content that is valid and suitable inside an XHTML `<div>` element
cc-9	A brief title for the comment; **SHOULD** include only content that is valid and suitable inside an XHTML `<span>` element
cc-10	When an OSLC Server incurs an error, it is **RECOMMENDED** that useful information be provided to clients in the body of the HTTP response.
cc-11	OSLC Servers **SHOULD** use the Error resource defined below as the basis for forming error responses.
cc-12	OSLC Servers **SHOULD** return an Error resource using the same representation format requested by the client via the HTTP `Accept` request header. [HTTP11]
cc-13	OSLC Clients **SHOULD** treat the `oslc:statusCode` as a String that starts with digits, but **MAY** contain non-digit text.
cc-15	Values **MUST** be expressed in relative length units as defined in the W3C Cascading Style Sheets Specification (CSS 2.1) Em and ex units are interpreted relative to the default system font (at 100% size)
cc-16	A resource giving more information on the error **SHOULD** be of an HTML content-type
cc-17	Resource representations returned via [OSLCCore2] Resource Paging **MUST** include a resource of type oslc:ResponseInfo, as defined in this section.
cc-18	Where a paged resource supports POST with an application/x-www-form-urlencoded body as an alternative to GET to avoid the request URI exceeding server limitations, the oslc:ResponseInfo in the response to the POST **SHOULD** contain this property so that a client knows what to POST to get the next page
cc-19	This optional property indicates the total number of results across all pages, its value **SHOULD** be non-negative
cc-20	In the context of a query resource, this value **SHOULD** be the total number of results, i.e. the number of resources that match the query
cc-21	In the context of other resources, the value **SHOULD** be the total number of property values (i.e. RDF triples) of the resource
cc-22	Unless Stable Paging is in effect, the total count **MAY** vary as a client retrieves subsequent pages
cc-23	Constraints on OSLC Core and Domain resources **SHOULD** be described using ResourceShapes which is included as part of the OSLC Core multi-part specifications.
cc-24	Servers **MAY** use other constraint languages such as [SHACL] to define resource constraints.
cc-25	Servers **MAY** also organize the ServiceProviders in one or more ServiceProviderCatalog LDPCs to enable OSLC clients to find ServiceProviders offered
cc-26	In most cases this namespace URI will be for an OSLC domain, but other URIs **MAY** be used
cc-27	In most cases this namespace URI will be for an OSLC domain, but other URIs **MAY** be used
cc-28	A Creation Factory **MAY** provide Resource Shapes that describe shapes of resources that may be created
cc-29	The Query Capability **SHOULD** provide a Resource Shape that describes the query base URI
cc-30	Service Providers **MUST** provide a Prefix Definition for each prefix supported by the service
cc-31	Each Prefix Definition defines a namespace prefix that clients **MAY** use in forming OSLC Query Syntax strings

Clause Number	Requirement
cc-32	Service Providers that support OAuth Authentication **SHOULD** provide a way for clients to automatically discover the three OAuth URIs necessary to act as an OAuth Consumer
cc-33	MIME type of the attachment content; **SHOULD** be a PURL media-type resource
cc-34	OSLC servers **MUST** follow the constraints defined here where required, and with the meanings defined here.
cc-35	OSLC servers **MAY** provide additional constraints for specific purposes.

