OSLC Core Version 3.0. Part 6: Resource Shape

Committee Specification Draft 03 / Public Review Draft 03

24 May 2018

Specification URIs

This version:

http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd03/part6-resource-shape/oslc-core-v3.0-csprd03-part6-resource-shape.html (Authoritative)
http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd03/part6-resource-shape/oslc-core-v3.0-csprd03-part6-resource-shape.pdf

Previous version:

http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/cs01/part6-resource-shape/oslc-core-v3.0-cs01-part6-resource-shape.html
http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/cs01/part6-resource-shape/oslc-core-v3.0-cs01-part6-resource-shape.pdf

Latest version:

http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/oslc-core-v3.0-part6-resource-shape.html (Authoritative)
http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/oslc-core-v3.0-part6-resource-shape.pdf

Technical Committee:

OASIS OSLC Lifecycle Integration Core (OSLC Core) TC

Chairs:

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

Martin Sarabura (msarabura@ptc.com), PTC

Editors:

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

Arthur Ryman (arthur.ryman@gmail.com), Ryerson University

Additional artifacts:

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

- *OSLC Core Version 3.0. Part 1: Overview*. http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd03/part1-overview/oslc-core-v3.0-csprd03-part1-overview.html
- *OSLC Core Version 3.0. Part 2: Discovery*. http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd03/part2-discovery/oslc-core-v3.0-csprd03-part2-discovery.html
- *OSLC Core Version 3.0. Part 3: Resource Preview*. http://docs.oasis-open.org/oslc-core/oslc-

core/v3.0/csprd03/part3-resource-preview/oslc-core-v3.0-csprd03-part3-resource-preview.html
- *OSLC Core Version 3.0. Part 4: Delegated Dialogs*. http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd03/part4-delegated-dialogs/oslc-core-v3.0-csprd03-part4-delegated-dialogs.html
- *OSLC Core Version 3.0. Part 5: Attachments*. http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd03/part5-attachments/oslc-core-v3.0-csprd03-part5-attachments.html
- *OSLC Core Version 3.0. Part 6: Resource Shape (this document)*. http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd03/part6-resource-shape/oslc-core-v3.0-csprd03-part6-resource-shape.html
- *OSLC Core Version 3.0. Part 7: Vocabulary*. http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd03/part7-core-vocabulary/oslc-core-v3.0-csprd03-part7-core-vocabulary.html

Related work:

This specification is related to:

- *OSLC Core Version 3.0: Link Guidance. Work in progress. Current draft*: https://tools.oasis-open.org/version-control/svn/oslc-core/trunk/supporting-docs/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 OASIS OSLC Lifecycle Integration Core (OSLC Core) TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=oslc-core#technical.

TC members should send comments on this specification to the TC’s email list. Others should send comments to the TC’s public comment list oslc-core-comments@lists.oasis-open.org, after subscribing to it by following the instructions at the “Send A Comment” button on the TC’s web page at https://www.oasis-open.org/committees/oslc-core/.

This Committee Specification Public Review Draft is being developed under the RF on Limited Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC’s web page (https://www.oasis-open.org/committees/oslc-core/ipr.php).

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. 25 May 2018. OASIS Committee Specification Draft 03 / Public Review Draft 03. http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd03/part6-resource-shape/oslc-core-v3.0-csprd03-part6-resource-shape.html. Latest version: http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/oslc-core-v3.0-part6-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.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark for above guidance.

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 [OSLCCore3] 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 [OSLCCore3], 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*. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7230

[LDP]

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

[OSLCCore3]

Steve Speicher; Jim Amsden. *OSLC Core 3.0*. URL: https://tools.oasis-open.org/version-control/svn/oslc-core/trunk/specs/oslc-core.html

[RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

1.2.2 Informative references

[JSONSchema]

JSON Schema. Draft. URL: http://json-schema.org/

[LDDI]

Tim Berners-Lee. *Linked Data Design Issues*. URL: http://www.w3.org/DesignIssues/LinkedData.html

[LDINT]

Arthur Ryman. *Linked Data Interfaces: Define REST API contracts for RDF resource representations*. URL: http://www.ibm.com/developerworks/rational/library/linked-data-oslc-resource-shapes/

[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. URL: http://www.w3.org/2012/12/rdf-val/

[REST]

Roy Thomas Fielding. *Representational State Transfer (REST)*. URL: http://www.ics.uci.edu/%7Efielding/pubs/dissertation/rest_arch_style.htm

[RWLD]

Tim Berners-Lee. *Read-Write Linked Data*. URL: http://www.w3.org/DesignIssues/ReadWriteLinkedData.html

[SHACL]

Holger Knublauch; Arthur Ryman. *Shapes Constraint Language (SHACL)*. Draft. URL: https://w3c.github.io/data-shapes/shacl/

[ShapesRDFVAL]

Achille Fokoue; Arthur Ryman. *OSLC Resource Shape: A Linked Data Constraint Language*. 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*. URL: http://www.w3.org/TR/turtle/

[WEBARCH]

Ian Jacobs; Norman Walsh. *Architecture of the World Wide Web, Volume One*. 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**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL** in this specification are to be interpreted as described in [RFC2119].

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 [OSLCCore3]. 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:

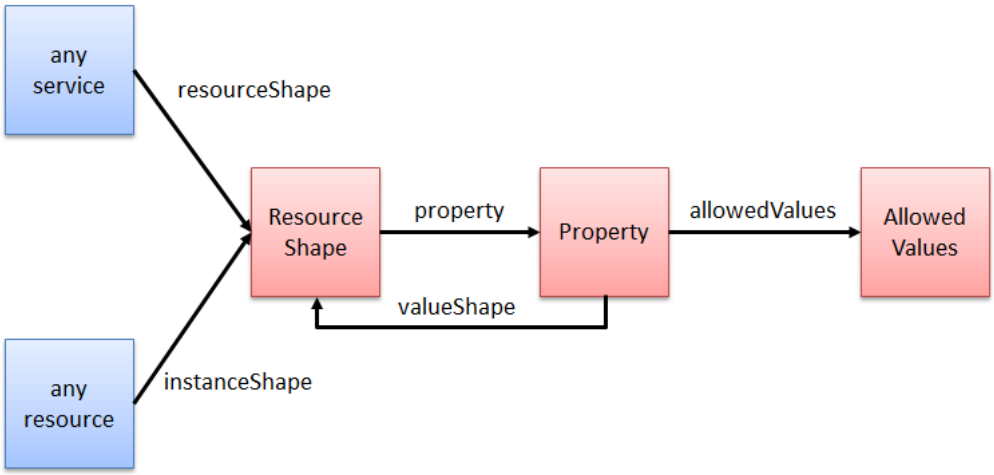

Fig. 2 Diagram of main concepts and relations.

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.

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 normally be interpreted as an error condition.

If exactly one shape applies to a resource then that resource SHOULD satisfy all the constraints defined by that shape.

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).

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.

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

[LDINT], [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. This property may have one of the values oslc:Exactly-one, oslc:One-or-many, oslc:Zero-or-many, and oslc:Zero-or-one.

Read-Only

`true` if the property is read-only. If omitted, or set to false, then the property is writable. Servers 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. Clients should note that the converse does not apply: Servers MAY reject a change to the value of a writable property.

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.

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.

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

- **Name:** `ResourceShape`
- **Type URI:** `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.

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.

`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.

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.

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

- **Name:** `Property`
- **Type URI:** `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.
`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: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-	true	Resource	Reference	`oslc:ResourceValueType`	The type of values of the defined

many	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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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 RDF and XML Schema. 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.

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.

5.3 AllowedValues Constraints

- **Name:** `AllowedValues`
- **Type URI:** `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-	true	unspecified	Either	Unspecified	Specifies the allow values in

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 Core TC 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.

Appendix B. Change History

This section is non-normative.

Revision	Date	Editor	Changes Made
01	04 April 2017	Jim Amsden	CS was approved and published.
03	24 May 2018	Jim Amsden	Minor editorial changes.