OSLC Link Discovery Management Version 1.0. Part 1: Specification

Project Specification Draft 01
21 December 2023

This stage:
https://docs.oasis-open-projects.org/oslc-op/ldm/v1.0/psd01/link-discovery-management-spec.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/ldm/v1.0/psd01/link-discovery-management-spec.pdf

Previous stage:
N/A

Latest stage:
https://docs.oasis-open-projects.org/oslc-op/ldm/v1.0/link-discovery-management-spec.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/ldm/v1.0/link-discovery-management-spec.pdf

Latest version:
https://open-services.net/spec/ldm/latest

Latest editor's draft:
https://open-services.net/spec/ldm/latest-draft

Open Project:
OASIS Open Services for Lifecycle Collaboration (OSLC) OP

Project Chairs:
Jim Amsden (jamsden@us.ibm.com), IBM
Jad El-khoury (jad@kth.se), KTH

Editors:
Eran Gery (eran.gery@il.ibm.com), IBM
Martin Ulrich (Martin.Ulrich@de.bosch.com), Bosch

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

- *OSLC Link Discovery Management Version 1.0. Part 1: Specification*. https://docs.oasis-open-projects.org/oslc-op/ldm/v1.0/psd01/link-discovery-management-spec.html
- *OSLC Link Discovery Management Version 1.0. Part 2: Machine-readable OpenAPI yaml*. https://docs.oasis-open-projects.org/oslc-op/ldm/v1.0/psd01/ldm-service.yaml

RDF Namespaces:

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

Abstract:

OSLC Link Discovery Management service defines an OSLC RESTful web services API for the discovering incoming links to a set of possibly versioned resources. To support these scenarios, this specification defines a set of HTTP-based RESTful interfaces in terms of HTTP methods: GET, POST, PUT and DELETE, as well as HTTP response codes, content type handling and resource formats.

Status:

Citation format:

When referencing this specification the following citation format should be used:
[OSLC-LDM-1.0]
OSLC Link Discovery Management Version 1.0. Part 1: Specification. Edited by Eran Gery and Martin Ulrich. 21 December 2023. OASIS Project Specification Draft 01. https://docs.oasis-open-projects.org/oslc-op/ldm/v1.0/psd01/link-discovery-management-spec.html. Latest stage: https://docs.oasis-open-projects.org/oslc-op/ldm/v1.0/link-discovery-management-spec.html.

Notices

Table of Contents

1. Introduction

This section is non-normative.

OSLC specifications do not currently specify where links between resources are stored, or which server owns which links. OSLC specifications also do not address which links are stored for properties that might be considered inverses such as implements or implementedBy. Some implementations may choose to store both of these links, other implementations may store one link and use OSLC Query or other mechanisms to get incoming links.

With the introduction of OSLC Configuration Management, the simple technique of storing redundant backlinks becomes problematic for maintaining data consistency is not recommended. OSLC Link Guidance recommends storing the link on only one side and using some mechanism to query incoming links from the other direction. One approach in common practice is to use OSLC query where a client queries each of its related servers for incoming links. While this peer-to-peer query addresses the fundamental issues of getting incoming links without storing redundant backlinks, it does not scale or perform well in situations with more than a small number of related providers.

A more scalable approach that has been practiced by several implementors such as IBM ELM and MID Smartfacts, is to manage a centralized link discovery service that enables OSLC clients to discover incoming links. For example, a requirements management (RM) application can display incoming links to specific requirement resources from test cases stored in a QM application. To an end user the incoming links are essential to carry out various lifecycle activities, such as understanding coverage and/or impact of a set of requirements.

OSLC link Discovery Management is an OSLC specification defines a standard means for client applications to discover incoming links to resources. This specification follows existing practices for incoming links discovery and specifies the following services:

- How to inquire incoming links to versioned resources for a provided set of (displayed) concept resource URIs and a given configuration context
- Incoming links service providers operate in configuration enabled contexts. Therefore, the incoming link inquiries shall specify configuration contexts for the links
- A means of discovering service providers that contribute links to an LDM server.

The specification does not specify how link discovery services ingest data from the various lifecycle providers, in order to effectively support the inquiries. That said, a common way of ingesting and maintaining a link index to support the inquiries uses already standardized OSLC Tracked Resource Sets [OSLC-TRS-v3.0]

This specification is a [OSLCCore3] compliant specification, and as such most of its content are references to [OSLCCore3].

1.1 Terminology

Link

An assertion or RDF triple consisting of a (subject, predicate, object) that manifests an instance of a relationship between the referenced subject and object.

LDM Client

An implementation of an application that consumes services provided by LDM servers to discover the scope of available links, and incoming links to a particular set of target resources.

LDM Server

A server implementing the OSLC Link Discovery Management specification. OSLC LDM clients consume services provided by LDM Servers to discover incoming links. The use of the terms Client and Server are intended to distinguish typical consumers and providers of OSLC resources in a distributed environment based on REST. A particular application component could be a client for some OSLC domain services and a server for the same or another domain.

Concept Resource

An artifact (a requirement, test case, source code file, etc.) as an overall entity, independent of any specific version of that artifact. In Product Lifecycle Management (PLM) systems, this is often referred to as a 'master' item or part, of which there are revisions and versions or iterations. As defined in [OSLC-Config-1.0-Part1]

Versioned Resource

The state (including the properties) of a concept resource can vary over time, or for other reasons. A versioned resource is a concept resource that keeps track of different states. A version resource is a resource that represents one specific state of a versioned resource. Not every past state is necessarily kept; a server may elide or discard states that are no longer referenced. As defined in [OSLC-Config-1.0-Part1]

1.2 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. Base Requirements

The following sub-sections define the mandatory and optional requirements for an OSLC Architecture Management (OSLC AM) server.

2.1 Base Compliance

This specification is based on [OSLCCore3]. OSLC LDM servers **MUST** be compliant with both the core specification, **MUST** follow all the mandatory requirements in the normative sections of this specification, and **SHOULD** follow all the guidelines and recommendations in both these specifications. [ldm-1]

The following table summarizes the requirements from OSLC Core Specification as well as some additional requirements specific for LDM Servers. Note that this specification further restricts some of the requirements from the OSLC Core Specification. See the previous sections in this specification or the OSLC Core Specification to get further details on each of these requirements.

Requirement	Meaning
Absolute URIs	LDM Servers **MUST** use absolute URIs for all references to resources by properties [ldm-2]
Resource Operations	LDM Servers **MUST** support resource operations via standard HTTP operations [ldm-3]
Resource Paging	LDM Servers **SHOULD** provide resource paging, as defined in [OSLC-Core-3.0-Part1]. In case resource paging is supported, stable paging is required. Note: as the LDM server implements a post-method as defined below, consider that instead of using oslc:nextPage in oslc:ResponseInfo, the use of oslc:postBody is required. [ldm-4]
Discovery	LDM Service URI is configured explicitly in each LDM client. Therefore no dedicated discovery protocol is needed. Note: a scenario with multiple LDM Servers is possible, in which case multiple LDM Service URI are configured. This specification does not prescribe how these multiple LDM servers are managed together. A LDM Service **SHOULD** provide discovery of its link contributors via Service Provider Catalog which is pointing via oslc:serviceProviderCatalog to the contributing servers catalogs. [ldm-5]
Authentication	LDM Services **SHOULD** follow the recommendations for Authentication specified in [OSLCCore3] [ldm-6]
Error Responses	LDM Servers **SHOULD** provide error responses using OSLC Core defined error formats [ldm-7]
RDF/XML Representations	LDM Servers **MUST** support RDF/XML and **SHOULD** support Turtle representations for OSLC Defined Resources [ldm-8]
JSON+LD Representations	LDM Servers **MAY** support JSON+LD representations; those which do **MUST** conform to the OSLC Core Guidelines for JSON+LD [ldm-9]

2.2 Specification Versioning

This specification follows the specification version guidelines given in [OSLCCore3].

2.3 Resource Operations

LDM does not feature any resource operations other than retrieving a service provider catalog. LDM services return results in LDP containers.

2.4 Authentication

See [OSLCCore3], OSLC LDM puts no additional constraints on authentication.

2.5 Error Responses

The following error situations **MUST** be handled by the LDM-Server. In case the error occurs, the subsequent oslc:error **SHALL** be returned. [ldm-10]

- No Object resource provided

  dcterms:identifier = “MissingObject” oslc:message = “No Object resource provided”

  oslc:statusCode = 400 (Bad Request)

- Too many Object resources requested The LDM-Server **MAY** limit the amount of Object resources to be requested.

  dcterms:identifier = “LimitReached” oslc:message = “Too many Object resources requested. Limit = $1” $1 = Maximum amount of Object concept resources

  oslc:statusCode = 400 (Bad Request)

  [ldm-11]

- Concept resource requested for Object resource without specified Configuration Context

  dcterms:identifier = “BadObjectRequest” oslc:message = “Concept resource requested for Object resource without specified Configuration Context”

  oslc:statusCode = 400 (Bad Request)

2.6 Pagination

OSLC LDM Servers **SHOULD** support pagination of query results and **MAY** support pagination of a single resource's properties as defined by [OSLCCore3]. [ldm-12]

3. LDM Service Description

Discover links inquiry

LDM clients inquire for incoming links related to a set of target object resources, typically owned and/or visualized by the client. The discover links inquiry is exposed under the discover-links path as described in the open-API section below For the incoming links inquiry, the client provides:

- Configuration Context [0..1]: URI
- Object resource [0..*]: URI
- Incoming link predicates [0..*]: URI

In case the Configuration Context is specified, any object- or subject-resource shall be specified based on concept resources. The server **MUST** respond with a set of triples for the incoming links, which are valid for the given Configuration Context. In case the Configuration Context is not specified, any object- or subject-resource shall be specified based on unversioned resources. the server **MUST** respond with a set of triples for the incoming links, which are not bound to a Configuration Context. Rationale: to support bidirectionality for non-configuration aware links In case no predicates are provided, the server **MUST** provide all incoming links irrespective of their predicate. [ldm-13]

The LDM Server response is a set of triples consisting of

- Subject resource: URI
- Predicate: URI

  should be one of the requested predicates, in case specified

- Object resource: URI

  should be one of requested Object concept resources

Based on the LDM service response, the LDM client would typically show the incoming link with an inverse predicate label (e.g., “validated by” vs. “validates”). The inquiry **MUST** be implemented as an HTTP post request, where the target concept resources and the predicates are provided with the request body. [ldm-14]

The following LDM example illustrates an OSLC RM client that inquires incoming links to requirements req1 and req2 in configuration baseline1 with predicates oslc_cm:tracks and oslc_cm:implements

EXAMPLE 1

```
POST http://ldm.example.com:8080/
Content-Type: application/x-www-form-urlencoded
Configuration-Context: http://elm.example.com/gcm/baseline1URI
objectConceptResources=http://elm.example.com/rm/req1URI,http://elm.example.com/rm/req2URI
predicateFilters=http:// elm.example.com/implementsURI, http://elm.example.com/tracksURI
```

Response (in turtle):

EXAMPLE 2

```
HTTP/1.1 200 OK
Content-Type: application/turtle


  oslc_cm:implements http://elm.example.com/rm/req1URI
  oslc_cm:tracks http://elm.example.com/rm/req1URI.
```

```
oslc_cm:tracks http://elm.example.com/rm/req2URI.
```

LDM contributions discovery

A LDM Service provides discovery of its link contributors via Service Provider Catalog which is pointing via oslc:serviceProviderCatalog to the contributing servers catalogs.

The link contributions discovery is exposed under the root-services path as described in the open-API section below.

3.1 OpenApi specification of the LDM service

LDM servers **MUST** implement the OpenAPI specification given in OSLC LDM Version 1.0: Part 2: Machine-readable OpenAPI yaml. [ldm-15]

4. Conformance

This specification is based on [OSLCCore3]. OSLC Link Discovery Management servers **MUST** be compliant with the Core specification, **MUST** follow all the mandatory requirements in the normative sections of each part of this specification, and **SHOULD** follow all the guidelines and recommendations in both these specifications. [ldm-16]

Clause Number	Requirement
ldm-1	This specification is based on [OSLCCore3]. OSLC LDM servers **MUST** be compliant with both the core specification, **MUST** follow all the mandatory requirements in the normative sections of this specification, and **SHOULD** follow all the guidelines and recommendations in both these specifications.
ldm-2	LDM Servers **MUST** use absolute URIs for all references to resources by properties
ldm-3	LDM Servers **MUST** support resource operations via standard HTTP operations
ldm-4	LDM Servers **SHOULD** provide resource paging, as defined in [OSLC-Core-3.0-Part1]. In case resource paging is supported, stable paging is required. Note: as the LDM server implements a post-method as defined below, consider that instead of using oslc:nextPage in oslc:ResponseInfo, the use of oslc:postBody is required.
ldm-5	LDM Service URI is configured explicitly in each LDM client. Therefore no dedicated discovery protocol is needed. Note: a scenario with multiple LDM Servers is possible, in which case multiple LDM Service URI are configured. This specification does not prescribe how these multiple LDM servers are managed together. A LDM Service **SHOULD** provide discovery of its link contributors via Service Provider Catalog which is pointing via oslc:serviceProviderCatalog to the contributing servers catalogs.
ldm-6	LDM Services **SHOULD** follow the recommendations for Authentication specified in [OSLCCore3]
ldm-7	LDM Servers **SHOULD** provide error responses using OSLC Core defined error formats
ldm-8	LDM Servers **MUST** support RDF/XML and **SHOULD** support Turtle representations for OSLC Defined Resources
ldm-9	LDM Servers **MAY** support JSON+LD representations; those which do **MUST** conform to the OSLC Core Guidelines for JSON+LD
ldm-10	The following error situations **MUST** be handled by the LDM-Server. In case the error occurs, the subsequent oslc:error **SHALL** be returned.
ldm-11	Too many Object resources requested The LDM-Server **MAY** limit the amount of Object resources to be requested. dcterms:identifier = “LimitReached” oslc:message = “Too many Object resources requested. Limit = $1” $1 = Maximum amount of Object concept resources oslc:statusCode = 400 (Bad Request)
ldm-12	OSLC LDM Servers **SHOULD** support pagination of query results and **MAY** support pagination of a single resource's properties as defined by [OSLCCore3].
ldm-13	In case the Configuration Context is specified, any object- or subject-resource shall be specified based on concept resources. The server **MUST** respond with a set of triples for the incoming links, which are valid for the given Configuration Context. In case the Configuration Context is not specified, any object- or subject-resource shall be specified based on unversioned resources. the server **MUST** respond with a set of triples for the incoming links, which are not bound to a Configuration Context. Rationale: to support bidirectionality for non-configuration aware links In case no predicates are provided, the server **MUST** provide all incoming links irrespective of their predicate.
ldm-14	Based on the LDM service response, the LDM client would typically show the incoming link with an inverse predicate label (e.g., “validated by” vs. “validates”). The inquiry **MUST** be implemented as an HTTP post request, where the target concept resources and the predicates are provided with the request body.

Clause Number	Requirement
ldm-15	LDM servers **MUST** implement the OpenAPI specification given in OSLC LDM Version 1.0: Part 2: Machine-readable OpenAPI yaml.
ldm-16	This specification is based on [OSLCCore3]. OSLC Link Discovery Management servers **MUST** be compliant with the Core specification, **MUST** follow all the mandatory requirements in the normative sections of each part of this specification, and **SHOULD** follow all the guidelines and recommendations in both these specifications.

Appendix A. 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 (co-chair)

Appendix B. References

B.1 Normative references

[OSLCCore3]

Jim Amsden; S. Speicher. *OSLC Core Version 3.0. Part 1: Overview*. OASIS. Project Specification Draft. URL: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/oslc-core.html

[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