OASIS OPEN PROJECTS

OSLC Core Version 3.0. Part 2: Discovery

Project Specification 01
17 September 2020

This stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/discovery.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/discovery.pdf

Previous stage:

https://docs.oasis-open-projects.org/oslc-op/core/v3.0/psd04/discovery.html (Authoritative)
https://docs.oasis-open-projects.org/oslc-op/core/v3.0/psd04/discovery.pdf
(published as Project Specification Draft on 20 December 2019)

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

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.* oslc-core.html
- *OSLC Core Version 3.0. Part 2: Discovery (this document).* discovery.html
- *OSLC Core Version 3.0. Part 3: Resource Preview.* resource-preview.html
- *OSLC Core Version 3.0. Part 4: Delegated Dialogs.* dialogs.html
- *OSLC Core Version 3.0. Part 5: Attachments.* attachments.html
- *OSLC Core Version 3.0. Part 6: Resource Shape.* resource-shape.html
- *OSLC Core Version 3.0. Part 7: Vocabulary.* core-vocab.html
- *OSLC Core Version 3.0. Part 8: Constraints.* core-shapes.html

- *OSLC Core Version 3.0. Part 9: Machine Readable Vocabulary Terms*. core-vocab.ttl
- *OSLC Core Version 3.0. Part 10: Machine Readable Constraints*. 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 OASIS Open Services for Lifecycle Collaboration (OSLC) OP 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.

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. 17 September 2020. OASIS Project Specification 01. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/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*. June 2014. Proposed Standard. URL: https://httpwg.org/specs/rfc7230.html

[LDP]

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

[OSLCCore2]

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

[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

[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].

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.

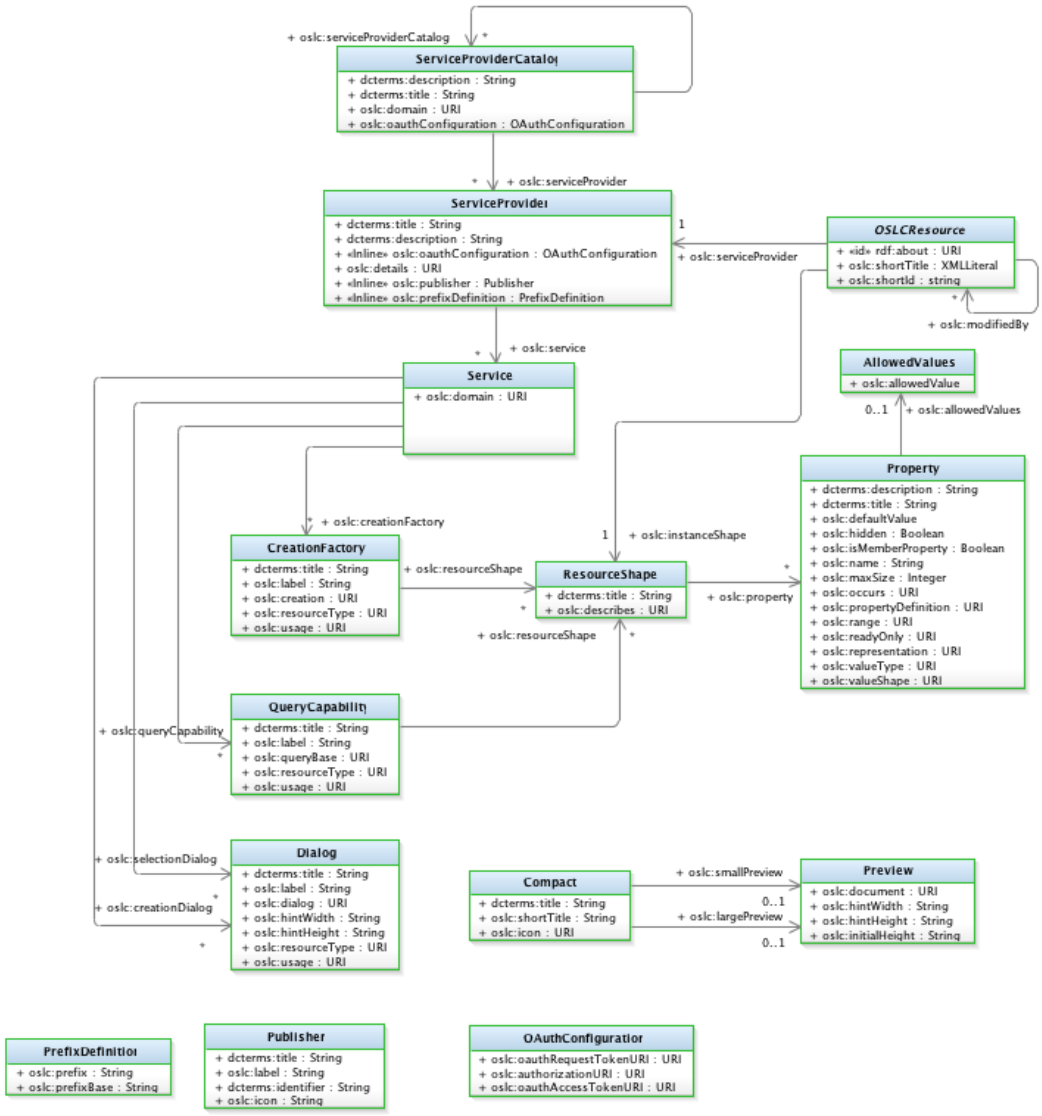

Fig. 1 Service Provider concepts and relationships

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 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-3]

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-4]

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-5]

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-6]

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-7]

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"
```

4.3 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-8]

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-9]

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-10]

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-11]

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.

4.4 Resource User Interface Preview Discovery

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

4.5 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.

4.6 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-12]

Servers **MAY** provide OAuth configuration information in the OAuthConfiguration member of a ServiceProviderCatalog as described in 5. Resource Constraints. [dis-13]

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. 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.
`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.
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.
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.
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. Each Prefix Definition defines a namespace prefix that clients **MAY** use in forming OSLC Query Syntax strings.

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.

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	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-4	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-5	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-6	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-7	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-8	Servers **MAY** describe constraints enforced on resource representations through the `http://open-services.net/ns/core#resourceShape` property of a Creation Factory resource.
dis-9	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-10	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-11	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-12	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-13	Servers **MAY** provide OAuth configuration information in the OAuthConfiguration member of a ServiceProviderCatalog as described in 5. Resource Constraints.

