Guidance on Linking Resources

This working document discusses OSLC guidance on how to represent, manage and interpret links between OSLC resources, and in general any linked data resources.

OSLC version 2.0 provided guidance on linking. This document will provide an updated set of guidance to be included as part of the Core 3.0 set of specifications.

Design Goals and Constraints

1. Keep links simple.
2. The term "Link" is a relationship between two resources when one resource (representation) refers to the other resource by means of a URI.
3. Links are uni-directional.
4. Links are represented in RDF with a single statement (triple).
5. A link triple may be stored in any graph. It may be stored in the graph of the subject URI, the object URI, or some other arbitrary graph.
6. The RDF representation of a resource is a general graph. The graph may have statements about the resource or other resources. The resource graph provides the context of the triple. The decision of which resource graph to insert a triple should be consistent with the normal lifecycle of the resource.
7. Avoid creating aliases (multiple names for same meaning) for link predicates (true for properties in general).
8. Promote the reuse of existing vocabularies.
9. Do not ~~want to~~ rely on OWL and RDFS inferencing.
10. Predicate URIs should be directly resolvable and its properties include an rdfs:label that is used in user interfaces.
11. Properties on a link can be stored using reified statements, however ~~they should~~ do NOT ~~be~~ use~~d~~ reified statements to cache information about subject or object resources. They should only represent properties of the link itself.
12. Avoid c~~C~~aching resource information outside of the service provider that owns that information ~~should be avoided~~. If caching is used, it should be treated like HTTP caching.
13. To improve performance, minimize HTTP ~~calls should be~~message flows ~~minimized~~.
14. Maintain alignment with principles in the Linked Data Platform.
15. Maintain alignment with the Core 3.0 guidance on vocabularies.

Comment [JA1]: define

Comment [JA2]: how is translation handled?

HTTP resource and RDF resource

A resource accessible with a~~n~~ HTTP URI using the standard HTTP methods (HEAD/GET/PUT/DELETE) is considered an HTTP resource. It may contain a graph of RDF data, whose subject URI(s) may or may not include the HTTP resource URI. In the absence of named graphs in the resource, consider all the statements ~~are considered~~ part of the graph identified by the HTTP request~~-~~URI. An HTTP resource has an ETag and a Last-Modified date.

These values apply to the *graph*, not necessarily to the *HTTP resource*. Most of the time the triples in the graph have the same subject URI as the HTTP resource URI.

Formatted: Font: Italic

An RDF resource is simply a URI, about which statements can be made. When we refer to an RDF resource, we are talking about the subject URI, that may have statements about it (i.e. statements where the resource URI is the subject).

Comment [JA3]: Doesn't parse. If it's a subject URI, then I do (not may) have at least one statement. Otherwise it's just a naked URI. Maybe you meant:
When we refer to an RDF resource, we are talking about the subject URI shared by a set of triples. Hence, a graph may refer to more than one RDF resource.

It is possible ~~for~~ to make statements ~~to be made~~ about a subject URI yet store the statements at a different URI. ~~to not be stored in graph of the subject URI.~~ This might be the case when the user does not have rights to edit the HTTP resource

Requirements

A set of requirements that the link guidance will satisfy. The link guidance statements to be part of the OSLC Core 3.0 set of specifications shall ensure that the following requirements could be satisfied by OSLC service provider and client implementations.

1. Users can create links from one OSLC resource to another from any resource's context (i.e. either service provider's native user interface).
2. When viewing a resource in a user interface, the user wishes to see all the links that the resource is either a subject or an object of.
3. User interfaces show links with human readable link types, and human readable labels (oslc:shortTitle, or dcterms:title) for both the subject and object of the link.
4. User interfaces should use human-readable link role names. Role names may vary for subject, object and link type, but not by link instance (i.e., there will not be support for two links of the same type to have different role names).
5. Additional properties can be stored on the link itself, however they should describe the link and NOT the object or subject.
6. If a user does not have view rights to a graph that a link triple is stored in, the link is not visible to the user.
7. Conceptually a link appears as a property of an RDF resource. This property is normally stored in the graph identified by the subject URI, but it may be stored anywhere. When a client GETs a URI used as ~~the RDF resource with~~ the subject URI in an RDF triple, the HTTP server returns the entire ~~all the properties in that~~ graph stored at that URI~~are returned~~. This graph is the HTTP resource. ETag and Last-Modified headers represent HTTP resources. A HTTP resource is a single graph that may or may not contain all the properties of the RDF resource with the same subject URI.
8. When a link is created or edited the subject RDF resource's dcterms:modified and dcterms:contributor properties should change. However, if the link is managed in some other graph, the service provider might not know to change ~~it~~those property values. ~~So c~~Clients should be aware that those properties might not be concep~~e~~tually accurate across all graphs in all service providers.
9. Any system of OSLC service providers should provide a way/mechanism for a client to get all links to and from a given RDF resource.

Comment [JA4]: You carefully distinguished between HTTP and RDF resources, then switch to a new term here that is neither.

Comment [JA5]: define

Comment [JA6]: which kind

Comment [JA7]: would be useful to clarify that no corresponding restriction exists on the object side.

Use Cases

1. Find all links - A client finds all the links that a resource is either the subject or object of.
2. Create link - A client creates a link from a resource to another
3. Create link in other resource - A client creates a link from a resource that it does not have edit rights on, to another resource that it does.

Additional use cases to be discussed

1. As a administrator, I should be able to restrict what links can be made between what resources, so that I can enforce a consistent information architecture.

OSLC Core 3.0 Linking Guidance

1. Whenever possible, links should be stored in the graph identified by the subject URI.
2. Link predicates should be directly resolvable.
3. Clients should not make assumptions about the type or properties of a resource that is the target of a link based solely on the link predicate and the rdf:type of the subject resource.
4. When absolutely necessary properties can be associated with a link statement itself using reified statements. However these properties should not be used to cache or store properties about the subject or object resources. They should only store properties about the link itself, not the resources.
5. When defining a new vocabulary with link types, there should be exactly one predicate for each link type concept. That is, ~~there should not be~~avoid defining~~ed~~ two different link predicates representing the same ~~thing~~meaning, with just the subject and object reversed (e.g. validates / validatedBy).
6. Clients should be aware that links may be stored in other graphs. Clients wishing to get all links to/from a given resource (or set of resources) can use OSLC Simple Query to query all providers in a given configuration.

Comment [JA8]: Did you mean Query Syntax?

Comment [JA9]: This is implementation-specific, so not clear it belongs in OSLC.