OSLC bootstrap discovery

Thursday, 4 October 2018 by Andrew Berezovskyi

Analysis

Introduction

As everyone understands, a zero-knowledge discovery has to start with something known. Then, the only source of new information is an HTTP response. So, we need to be able to issue an HTTP request to a certain URI and from its response we can get:

1) A HTTP response body in a given MIME type (i.e. RDF in our case)
2) HTTP headers:
    a. “Link” headers that bear a destination URI and a “rel” attribute;
    b. a “Location” URI in case of a redirect.
3) A body with the server (hyper) information through an OPTIONS request according to the HTTP spec, the OPTIONS response may[1] have a body, so I would consider that every URI may serve different bodies for a GET and for an OPTIONS requests).

As you see, there is nothing specific about “Link” headers or the well-known URIs. Now that we have enumerated all sources of information, let’s look at the questions a discovery implementer would have to answer:

1) Which URI to make a request to?
2) Whether it is a GET/HEAD or an OPTIONS request?
3) Where to look for the information in the response.

The questions no. 2 and 3 are not easy but can be standardised. The question no. 1 requires us to look at the scenarios that provide the motivation for this bootstap discovery feature in the first place.

Scenario no. 1 (sunny day): A user has browsed through the web UI of a tool to a certain resource and wants to link to it from another tool (or perform another operation, like searching for all resources pointing to it). The tool shall be able to run the discovery based on a URI of an OSLC Resource.

Scenario no. 2 (rainy day): A user has browsed through the web UI of a tool to a certain resource and wants to link to it from another tool (or perform another operation, like searching for all resources pointing to it). *However, the user is browsing a legacy UI and the URIs there are not the same as the OSLC resource URIs. The OSLC resources have URIs like api.tool.ex/sp1/r123. This means an OSLC client does not have any means to do OPTIONS request or content negotiation at all on that URI a user has copied!*

[1] See https://tools.ietf.org/html/rfc2616.html#section-5.1.2 and https://tools.ietf.org/html/rfc2616.html#section-9.2 (despite what Mozilla says in https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS)

Scenario no. 3: A system is using Kubernetes or a similar mechanism to deploy many OSLC adaptors for various legacy tools in addition to a limited number of tools already supporting the OSLC specification. Certain OSLC adaptors want to discover all Service Providers in a system for certain Resource Shapes and, for example, subscribe to the TRS Servers of their OSLC Servers.

Scenario no. 4: An OSLC Client encounters an RDF object URI pointing to an OSLC Resource in one of the responses. An OSLC Client might want to fetch a resource but also to subscribe to the TRS Server and needs to discover the SPC.

Proposal

The issues no. 9 neither deals with the `RootServices` document standardisation nor with the ".well-known" / "OPTIONS *" approaches. Instead, it proposes to use HTTP "Link" headers in order to indicate the right URI. I think this is a viable approach, but insufficient on its own due to the following problems:

1) Not all users will have control over the HTTP headers, for example admins of the static sites published on Github or plugin developers for various systems (e.g. Jira).
2) Systems requiring an authentication to take place will not be able to return a 2xx response and an OSLC Client might not be able to read the Link header.

My proposal lies in two orthogonal but closely related steps:

1) File an application for "/.well-known/oslc/" URI prefix to allow zero-knowledge configuration of an OSLC Client given:
    a. a domain name where an OSLC Server is deployed, or
    b. a URI pointing to any of its managed resources.
2) Standardise the Root Services document as a formal machine-readable XSD schema and allocate the "/.well-known/oslc/rootservices.xml" URI for it.

This way, if a user tries to link to the URI app.example/bugs.jsp?b=123, an OSLC Client can make a HEAD request to it and see it cannot be retrieved as an RDF document. app.example/.well-known/oslc/rootservices.xml can redirect to api.app.example/rootservices. Finally, our app can show a JS dialog that the URI cannot be linked to directly BUT hey, here are some selection dialogs to find the right link to the same resource!

Other ideas

There are more proposals one can think of once these two are accepted, but I will only enumerate them here without proposing any concrete action:

1) Allow "OPTIONS *" request to serve the Root Services definition and/or have the Link headers. "OPTIONS *" is even harder to handle than adding Link headers and may be impossible, for example, when using AWS Lambda functions behind AWS API Gateway or similar solutions that require URI mapping to be performed (because "*" is not a URI path that can be mapped).
2)

Older proposals

There were the following discovery-related issues before:

- [OSLCCORE-9] Bootstrapping discovery
- [OSLCCORE-16] Discovery should include text about vocabulary discovery
- [OSLCCORE-49] OSLC discovery supports discovering server capabilities, but not client needs
- [OSLCCORE-82] Propose method for access context discovery
- [OSLCCORE-85] How to discover TRS
- [OSLCCORE-185] Prepare the spec for .well-known allocation from IANA

Link rel=resourceDiscoveryRoot (OSLCCORE-9)

Martin Pain: *Here's a draft of the proposal (in specification-like) form for review and possibly a vote by the TC at the Aug 6 meeting.*

In order for end users or tool integrators to leverage OSLC Discovery capabilities, they must have a URI that establishes a starting point. A starting point could be a URI to a Service Provider Catalog, or a URI to a particular LDPC. Either of these URIs would allow a client to discover other URIs and capabilities that may be exploited to meet their needs.

OSLC Core Discovery specifies how to discover capabilities. But there also needs to be a standard way of determining the URIs a client would need to know in order discover the supported capabilities. **By convention, some OSLC servers utilize a `rootservices` document at the root of the server URI** that contains properties whose value is the URI for a service provider catalog. LDP specifically states that it does not define how clients discover LDPCs (see LDP 1.0, section 5.2.1). Tool integration is simplified by providing a standard way of discovering URIs that are intended to support discovery of capabilities provided by an OSLC compliant server.

Kommenterad [AB1]: At the very least, we could have this URI standardised as /.well-known/oslc/rootservices.xml

A user or client application of course has to know some URI in order to access a server at all. This URI would include the supported protocol, host name, port, path and any other URI components the server requires. Often servers will have some well-known home page, often '/', that provides and entry point into hyper-linked content provided by that server. This URI may not be the root discovery URI itself, but can provide a means of discovering root discovery URIs.

Kommenterad [AB2]: I would like to consider two scenarios here:

1) A user is browsing the web UI of a tool and finds a page with a resource they want to link to.
2) An OSLC client encounters a URI as an object of a certain property in a response and wants to traverse it but is not sure whether the resource under that URI is managed by an OSLC Server.

1. Servers SHOULD include one or more `Link: <discovery-URI>; rel="http://open-services.net/nx/core#resourceDiscoveryRoot"` headers on an HTTP GET, HEAD, or OPTIONS request on the server's advertised or default home page. The discovery-URI is the URI of a ServiceProviderCatalog resources, ServiceProvider resources or LDPC that can be used for subsequent discovery of OSLC capabilities.

Kommenterad [AB3]: That makes sense, but is impossible to implement on Github and other static hosts without access to the `.htaccess` document and/or a hander on the '`/`' context.

2. The `Link: <discovery-URI>; rel="http://open-services.net/nx/core#resourceDiscoveryRoot"` headers SHOULD allow discovery of all OSLC resources and capabilities on that server either directly, or indirectly through referenced ServiceProviderCatalogs and/or contained LDPCs.

Kommenterad [AB4]: This is hard to guarantee.

3. Servers MAY provide `Link: <discovery-URI>; rel="http://open-services.net/nx/core#resourceDiscoveryRoot"` headers on an HTTP GET, HEAD,

or OPTIONS request on other URIs that are likely to be seen to represent the OSLC server's root or the OSLC server as a whole. These different URIs could represent a partitioning of the OSLC resources and capabilities provided by the server.