What Does it Mean for a URI to Resolve?

Joel Sachs, Tim Finin

Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
1000 Hilltop Circle Baltimore, MD 21090
jsachs@umbc.edu, finin@umbc.edu

Abstract

Amongst the best practices that constitute Linked Data, one of the foremost is to use only HTTP-URIs as identifiers for RDF resources. This is so that the URI will resolve in a Linked Data browser to give information about the named resource.

At the same time, Linked Data takes a resource-centric, as opposed to page-centric, approach to resolution. We argue that this approach can, in certain cases, obviate the need for insisting on HTTP-URIs. As a use of our "expanded" notion of Linked Data, we present as an example Life Science Identifiers.

Background

A Uniform Resource Identifier (URI) is a name with a special property – it is also a set of instructions for retrieving a representation of the thing being named [1]. The W3C RDF [2] and OWL [3] Recommendations are built around URIs. One of the first principles of Linked Data [4] is that all URIs should be HTTP URIs. The primary motivation for this is to make URIs resolvable by web browsers.

Linked Data, however, is changing the very way we think about resolution; Linked Data browsers typically take a resource-centric, as opposed to page-centric, approach to resolving URIs. For example, if you put *http://dbpedia.org/resource/Coffea* into a standard web browser, it will simply display the page *http://dbpedia.org/page/Coffea*. But if you put the same URI into a Linked Data browser, the browser will not simply list all triples stored at *http://dbpedia.org/data/Coffea*. Rather, it will list all triples it is aware of (i.e. that are in its cache) that are about the resource *http://dbpedia.org/resource/Coffea*. In other words, the URI resolves not to the address, but to all known information about the resource.

Now what if the server *dbpedia.org* goes down? Obviously, *http://dbpedia.org/resource/Coffea* will no longer resolve in a standard web browser. But, assuming that other servers have made assertions about *dbpedia:Cofeea*, the URI *can* still "resolve" in a Linked Data browser, in the sense that the browser can still know things about the resource. But this suggests that a URI doesn't need to resolve in the traditional sense in order to be a part of the Linked Data ecology.

LSIDs and Linked Data

Life Science Identifiers (LSIDs) represent an attempt to supply GUIDs (Globally Unique Identifiers) to life science resources, including genes, proteins, species, species occurrence records, and journal articles. Eventually, URNs were chosen as the mechanism for representing LSIDs [5], and so an LSID looks like this
urn:lsid:ncbi.nlm.nih.gov.lsid.i3c.org:omim:601077 .

The question of how to bring LSIDs into the linked data cloud has received much discussion in the biodiversity informatics community. The current thinking [6; 7] seems to be to support both traditional LSIDs, as well as HTTP URIs for life science resources. The latter is achieved by embedding LSIDs within HTTP URIs, e.g.
http://bioguid.info/urn:lsid:ipni.org:names:20012728-1:1.1

(Note that this URI is a non-information resource, about which the following is true:
http://bioguid.info/urn:lsid:ipni.org:names:20012728-1:1.1
owl:sameAs
urn:lsid:ipni.org:names:20012728-1:1.1)

Is this necessary? LSIDs are URIs, which means that we we can make assertions about them using the Resource Description Framework (RDF). Therefore, although they will not resolve in standard web browsers, they can resolve in a Linked Data browser, provided the browser is aware of the assertions that have been made about it. And, of course, there are many services (Google, Swoogle, Sindice, etc.) that can bring those assertions to the attention of the browser.

So LSIDs can perhaps join the Linked Data cloud as is, without having to be transformed into http URIs via http lsid resolvers. Not only this, but the minter of the LSID need not implement a resolution protocol, since standard HTTP architecture (search engines, linked data browsers) can take care of the resolution.

The LSID creation process would look like this:

- Create names that look like: ***lsid:YourOrganization:xyz***
- Say things about them on web/semantic web pages.

- Ping search engines to make sure that everyone knows about those web pages.

In other words, embrace the fact that HTTP search engines provide a sort of uniform resolution mechanism for all URIs.

This approach to incorporating LSIDs into the Linked Data cloud has these advantages:
1. It eases the burden of maintenance. Currently, to introduce an LSIDs, you must either i) provide a mechanism for resolution (traditional); ii) implement linked data - style content negotiation (modern); or iii) depend on the largesse of others to do i or ii for you.

2. It eases the burden of curation. If your server disappears tomorrow, so will all your LSIDs. But if we allow a URI to fucntion as a name, even when it fails as an address, then this is not the case.

Objections and Disadvantages

We have encountered a number of objections to the above. We present these below, together with our responses:
i. Branding and ownership
Data suppliers have to manage their reputation. If they publish something that is wrong they want to be able to correct it. If other people use their data then they want to know about it so they can justify funding, etc. These use cases require some form of normative version of the data.

We are proposing to permit the decoupling of names and webpages. So *lsid:organizationX:abc* would get described at *http://organizationX.org/genes/abc* . That latter URI can still be considered the normative description of the former.

ii. Authority
If I publish an LSID, what I have to say about it should take precedence over what others have to say.

We sympathize with this objection. In an ideal semantic web, applications never consume triples without recording their provenance, and only trust triples according to explicit trust policies. In practice, however, people often take what they can get, so it is important that they get the most trusted source of information about a particular resource, when they query for that resource.

But is the most trusted source for an LSID always the creator of the LSID? Not necessarily. For example, the discoverer of a gene may assert ownership rights over that gene that do not really exist. But even if we accept the notion of authoritative descriptions of resources, our proposed scheme will, in most cases, result in resolution to the authority. Consider a team that discovers a new species. They mint an LSID for, and publish an RDF description of the species. Their description will likely be the most cited description for that species, and so semantic web search engines will rank it higher than other descriptions. (Agreement between webpage domain and LSID authority can be part of the ranking algorithm.)

The current situation results in a resource automatically resolving to a particular description, i.e. the one that gets 303 redirected to (or, if using the hash method, the one that prepends the hash). In this sense, the publisher has absolute authority. If we allow non-HTTP URIS, we are removing some, but not all authority from the publisher.

iii. Why bother with LSIDs at all?
If resolution really doesn't matter (or is considered a bad thing) then why use LSIDs? Why not just use, e.g., urn:uuid:1721b080-db3f-11de-87f3-0002a5d5c51b?

Objections (i) and (ii) above provide the answer to objection (iii). An LSID embeds the organization name, and so allows the organization to be "attached" to the thing being named. It also allows each organization to use its own naming scheme.

iv. Dereferencing provides a means of discovery with no interaction.
The Linked Data browsers you describe would be dependant on full coverage from indices such as Swoogle. What about resources in the Web that have been indexed, and that are not already in the cache?

This is a good point, to which we have 3 responses:
1. Creating a web page and pinging search engines is simpler than other LSID creation mechanisms currently on the table. So it is reasonable to expect this protocol to be followed by those who want their information found. Even if it isn't, web pages describing LSIDs will eventually be found by Google, which will then find the link to the LSID. In the cases of rdf search engines which do not crawl the non-rdf web, there will, indeed, be a problem.

2. The line between browser and search engine (at least on the semantic web) is somewhat blurry, and is getting blurrier. Given the importance of being able to browse resources in the context of data already in your cache, we envision Linked Data browsers that will interact with search engine APIs to pre-load a user's cache with data relevant to the user's browsing.

3. We are not looking to replace the current paradigm, but simply to support, within the linked data ecology, URIs that are currently not supported.

Conclusions

Some communities continue to use non-http URIs as identifiers. Due to the resource-centric approach to resolution taken by Linked Data browsers, these non-http URIs can be incorporated into Linked Data. In particular, Linked Data applications should be able to understand and reason about LSIDs, as they currently exist.

References

1. http://www.w3.org/DesignIssues/NameMyth.html
2. http://www.w3.org/TR/REC-rdf-syntax/
3. http://www.w3.org/TR/owl-features/
4. http://www.w3.org/DesignIssues/LinkedData.html
5. http://xml.coverpages.org/lsid.html
6. http://www2.gbif.org/Persistent-Identifiers.pdf
7. http://www.tdwg.org/stdtrack/article/view/150