The necessity of hypermedia RDF and an approach to achieve it

Kjetil Kjernsmo[1]

Department of Informatics, Postboks 1080 Blindern, 0316 Oslo, Norway
kjekje@ifi.uio.no

Abstract. This paper will give an overview of the practical implications of the HATEOAS constraint of the REST architectural style, and in that light argue why hypermedia RDF is a practical necessity. We will then sketch a vocabulary for hypermedia RDF using Mike Amundsen's H Factor classification as motivator. Finally, we will briefly argue that SPARQL is important when making non-trivial traversals of Linked Data graphs, and see how a bridge between Linked Data and SPARQL may be created with hypermedia RDF.

1 Introduction

Mike Amundsen defines hypermedia types [3] as

> Hypermedia Types are MIME media types that contain native hyperlinking semantics that induce application flow. For example, HTML is a hypermedia type; XML is not.

Furthermore, he defines a classification scheme called H Factor as "a measurement of the level of hypermedia support and sophistication of a media-type." The REST & WOA Wiki defines "the Hypermedia Scale" [5], where the categorization is based on capabilities to do CRUD (Create, Read, Update, Delete) operations. Considered on their own, RDF serializations are hypermedia types, but only at the LO (outbound links) and CL (control data for links) levels (see [3] for details on this notation), and just an R Type (read) on the Hypermedia Scale. We aim at improving this situation.

Amundsen argues in a blog post [2] that the Semantic Web community should not pursue an API for RDF, but rather make RDF serializations more powerful hypermedia types. He argues that a key factor in the success of the Web is that messages not only contain data but also application control information, and that this is needed for the Web to scale.

Semantic Web services are likely to be an important source of data for future applications, but for foreseeable future they are just one of many. A key promise of the Semantic Web is the ability to integrate many data sources easily. There are two key issues, first read-write operations must be similar to how it is done with other hypermedia types, to make it possible to use generic code to minimize the effort required to support RDF. Importantly, if interacting with Linked Data

requires extensive out-of-band information, it will be harder to use than media types that do not. Secondly, it requires relevant links and that the resulting graph can be traversed by reasonable means.

While links are abundant across the Linked Open Data (LOD) cloud, this paper will argue that key links are missing to make it possible to traverse the resulting graph and to enable read-write operations. Moreover, we will argue that this shortcoming is due to that the community does not adequately take into account the constraint known as "Hypermedia as the Engine of Application State" (abbreviated HATEOAS) from the REST architectural style, see [6] Chapter 5.

In a read-only situation, the HATEOAS constraint requires that the application can navigate from one resources to another using the hypermedia links in the present resource *only*, they should not require any out-of-band information. Since RDF is built on URIs, many of which can be dereferenced to obtain further links, it will therefore almost always satisfy the HATEOAS constraint in the read-only case.

Contrast the read-write situation: It is very common that specifications that advertise themselves as RESTful has an API that is specified in terms of URI structure as well as HTTP verbs. This is not a violation of the HATEOAS constraint *per se*, but it would usually be superfluous as the same information *must* be available in a hypermedia message in order to satisfy the HATEOAS constraint. The constraint does not require this information to be available in *all* messages, and so, it is not a very strict constraint.

It is debatable whether a protocol can be said to be RESTful if the links don't enable any useful interactions. Even in the read-only case, we must carefully add links that enable the desired interactions to take place, e.g. linking a SPARQL endpoint. Importantly, if there is to be such a thing as a RESTful read-write Semantic Web protocol, there must be something in the RDF itself that can be used by practical applications to write data. Therefore, whether the HATEOAS constraint is satisfied must be judged on the basis of the practical applications the protocol enables.

2 Required links

We note that the Atom Publishing Protocol [7] has a single service endpoint. From there, you can navigate to what you need. This motivates the first discussion topic of this paper:

Question 1. Is a single service endpoint enough for Linked Data?

We note that the distributed graph structure of the LOD Cloud makes this awkward: For every new data source encountered in a graph traversal, a new service endpoint must be queried. Moreover, it would be awkward if fine-grained access controls for writing are being used to record permissions for all resources in a single service description. It seems likely that a few triples attached to each information resource are better suited in most cases.

3 Defining hypermedia RDF

We noted that for a read-write RESTful protocol, we need to say in the RDF message itself what kind of operations can be made. We also note that only information resources can be manipulated, and we argued that it should be possible to add the required triples to every resource. We should be able to define hypermedia RDF in terms of a minimal vocabulary, but like the H Factor web page, we will find it instructive to use examples. In the following, we use Turtle syntax, with prefixes omitted for brevity. Also, we note that in many cases, we are making statements about the current resource, which given a reasonable base URI can be written as <>. We have not found the following factors to be relevant to RDF: LE (embedded links), CR, CU (control data for read and update requests respectively) and LT (templated queries); and LO and CL are trivially supported. The remaining are:

LN Support for non-idempotent updates (HTTP POST)

```
<> hm:canBe hm:mergedInto ;
   hm:createSimilarAt <../> .
```

The first triple says that the current resource can be merged with another resource. In a typical HTTP case, this would be achieved by POSTing to the current resource with an RDF payload, and the server would perform a RDF merge of the payload with the current resource.
This term is motivated by that POST operations in [10] are specified to result in an RDF Merge of the payload into the named graph. Amongst other possibilities is a union of graphs.
The second triple exists to make it possible to POST an RDF payload to some URI and the server will itself assign a URI to the posted RDF payload. This may for example be a named graph, as defined in [10] or a follow the protocol recently suggested in [9] Section 5.4.

LI Support for idempotent updates (HTTP PUT, DELETE)

```
<> hm:canBe hm:replaced, hm:deleted .
```

The first triple says that the current resource may be replaced by PUTting an RDF payload to the resource URI. The second triple says that the resource may be deleted, typically by a HTTP DELETE on the resource URI.

CM Support for indicating the interface method for requests (e.g. HTTP GET, POST, PUT, DELETE methods).
For example (with similar triples for other HTTP methods):

```
hm:replaced hm:httpMethod "PUT" .
```

We see that with a simple vocabulary, RDF serializations can become a very powerful hypermedia types. We also note that this vocabulary enables agents to do the same operations as the SPARQL 1.1 Graph Store HTTP Protocol [10], while being fully RESTful. The Protocol specification is currently not RESTful

per the discussion above as it requires extensive out-of-band information, even though it was a key design goal. It should also be possible to use this on the default (nameless) graph by assigning it a name in the service description rather than defining it in an out-of-band specification like is currently being done.

Question 2. Should the SPARQL 1.1 Graph Store HTTP Protocol be replaced by a vocabulary like the above?

3.1 New H Factors

Mike Amundsen solicits feedback on the completeness of his classification scheme. The following are suggested as discussion items:

1. Support for self-description.
   The self-describing nature of RDF is an important characteristic of the model and arguably an important characteristic of RDF as hypermedia as agents may be able to infer possible interactions based on vocabulary definitions.
2. Support for supplying control data relevant to subsequent requests.
   It may be useful for agents to know e.g. what formats they can send to a given resource before the request is made. This may require a new control data factor. An example of this use may be e.g.:

   ```
   <> hm:acceptsFormat <http://www.w3.org/ns/formats/Turtle> .
   ```

3. A factor encompassing RaUL [8] templates.
   RaUL goes beyond HTTP GET queries, which is what the LT factor is about. The interaction RaUL enables are quite different from those discussed in this paper and are not captured by any of the current factors.

Question 3. Can we create a better vocabulary for hypermedia RDF than the sketch above?

4 Bridging LOD and SPARQL

In many cases, it is sufficient to get a small number of resources to collect the data needed for a given usage, but slightly more involved queries (e.g. "what kind of connections exists between Kate Bush, Roy Harper and bands that have sold more than 200 million albums?") would likely cause thousands of resources to be downloaded and examined. For this, more advanced graph pattern matching, as well as more advanced mechanisms for source selection, is required. [11] provides some techniques that should prove very valuable in this respect and so it becomes important that SPARQL can be used with Linked Data in a RESTful manner.

In [4] Tim Berners-Lee notes "To make the data be effectively linked, someone who only has the URI of something must be able to find their way the SPARQL endpoint.", but this is generally not possible today without requiring out-of-band information.

The triples needed to do this are already defined in VoID (see e.g. [1]) and are in use:

```
<> void:inDataset [ void:sparqlEndpoint </sparql> . ] .
```

5 Conclusion

We have shown how RDF serializations can become very powerful hypermedia types by using a simple vocabulary. As a consequence, developers can use Linked Data in a read-write scenario without specialized knowledge about RDF APIs or other out-of-band information. We have argued briefly that some triples should be added to every information resource, but note that most triples are only relevant to write-operations and should only appear in that case. We also noted that SPARQL is important for non-trivial traversal of Linked Data. To sum up, the addition of the following triples would make life easier for developers of applications that use Semantic Web data:

```
<> hm:canBe hm:mergedInto, hm:replaced, hm:deleted ;
   hm:createSimilarAt <../> ;
   hm:acceptsFormat <http://www.w3.org/ns/formats/Turtle> ;
   void:inDataset [ void:sparqlEndpoint </sparql> . ] .
```

References

1. K. Alexander and M. Hausenblas. Describing linked datasets - on the design and usage of void, the vocabulary of interlinked datasets. In *In Linked Data on the Web Workshop (LDOW 09), in conjunction with 18th International World Wide Web Conference (WWW 09)*, 2009.
2. M. Amundsen. API for RDF? don't do it! http://amundsen.com/blog/archives/1083, 2010.
3. M. Amundsen. Hypermedia Types. http://amundsen.com/hypermedia/, 2010.
4. T. Berners-Lee. Linked Data – Design Issues. http://www.w3.org/DesignIssues/LinkedData, 2009.
5. S. Bjorg. The Hypermedia Scale. http://restpatterns.org/Articles/The_Hypermedia_Scale, 2009.
6. R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, Irvine, California, 2000.
7. J. Gregorio and B. de hOra. The Atom Publishing Protocol. http://www.ietf.org/rfc/rfc5023, 2007.
8. M. Hausenblas, J. Umbrich, and A. Haller. RAUL Vocabulary. http://vocab.deri.ie/raul, 2011.
9. M. Nally, S. Speicher, J. Arwe, and A. L. Hors. Linked Data Basic Profile 1.0. http://www.w3.org/Submission/2012/SUBM-ldbp-20120326/, 2012.
10. C. Ogbuji. SPARQL 1.1 Graph Store HTTP Protocol. http://www.w3.org/TR/2011/WD-sparql11-http-rdf-update-20110512/, 2011.
11. A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. FedX: Optimization techniques for federated query processing on linked data. In *The Semantic Web ISWC 2011*, LNCS 7031:601-616.