Achieving Linked Enterprise Data with RESTful Services and Link Relations

W3C Workshop on Linked Enterprise Data

Cornelia Davis
EMC
cornelia.davis@emc.com

25 October 2011

While the proponents of the Linked Data movement prescribe the use of RDF, RDF Schema, OWL and SPARQL, with the “5-star model”, even the staunchest supporters acknowledge that there are other ways to achieve at least some level of data linking. In the general sense, Linked Data means that data entities are connected to one another in a manner that can be “understood” by programmatic clients. While the body of Linked Open Data has grown respectably, adoption of this paradigm in the enterprise has been slow. The transition of web services interfaces from an RPC/SOAP-based style to a RESTful style, a phenomenon that is happening in most enterprises today, is an excellent opportunity to begin making enterprise data available as linked data, even without requiring corporate buy-in on the full “Semantic Web.”

RESTful Services

The uptake of RESTful services out on the web[1] and the popularization of cloud computing has helped drive enterprises to develop RESTful interfaces to their products. In general, however, true understanding of the principles of REST remains weak and poses a risk to the potential for linked data that RESTful interfaces offer. All four of the primary REST principles are critical to the cause:

- **Identification and addressability of resources**: All interesting bits of information are identified with URIs and those URIs are generally URLs.
- **Uniform interface**: Interaction with resources is through a standardized set of operations, with well understood semantics (usually HTTP).
- **Resource representations**: Manipulation of resources is through representations. Resources are not objects and representations are not serializations of data structures.
- **Hypermedia constraint**: Resource representations must carry hyperlinks that allow the application flow to progress [2]. Links may be to other resources or may be to actions that drive the application.

Of these tenets, resource orientation and uniform interface are often reasonably well applied (though there are a few common pitfalls even here), in large part because frameworks for the development of RESTful services address them very well. The transfer of resource representations, so fundamental to

1 Albeit, many so called “RESTful services” available on the Web today are, unfortunately, not very RESTful.

REpresentational State Transfer (REST) [1], is a pattern that is also generally followed, yet usually without full leverage of the power that media types afford. The hypermedia constraint, however, is often ignored, and without it RESTful services do not project linked-data. Within our enterprise we are championing RESTful services design patterns that deliver resource representations in the Atom Syndication Format [3] with liberal use of the `atom:link` element which provides the foundation for us to link our data.

The Atom Syndication Format

While the Atom Syndication Format (Atom) originated to serve blogging use cases, its use in practice has extended well beyond this initial target domain. Very popular, publicly available interfaces including those for Netflix, eBay and many Google applications, all deliver resource representations in this format and the uptake of these APIs has been significant. We have found that familiarity with Atom, or at least RSS, has made this concept very accessible to our developer base, eliminating one possible barrier to success in our goal to expose linked data in the enterprise. Atom has two primary domain objects, a singular entity called an *entry* and a collection of entries called a *feed*. In addition to carrying content, an Atom entry also includes properties that are defined in an atom namespace and any number of additional properties defined in other namespaces. This model is general enough to represent virtually any type of entity; data center entities such as switches, routers, hosts and storage devices, content objects such as documents and images, and even virtual machines. The Atom feed collects individual entries and adds metadata properties, again, both in the atom namespace and in other namespaces. It is natural for us to use this construct to present the list of storage devices connected to a host, or the list of documents stored in a folder, for example.

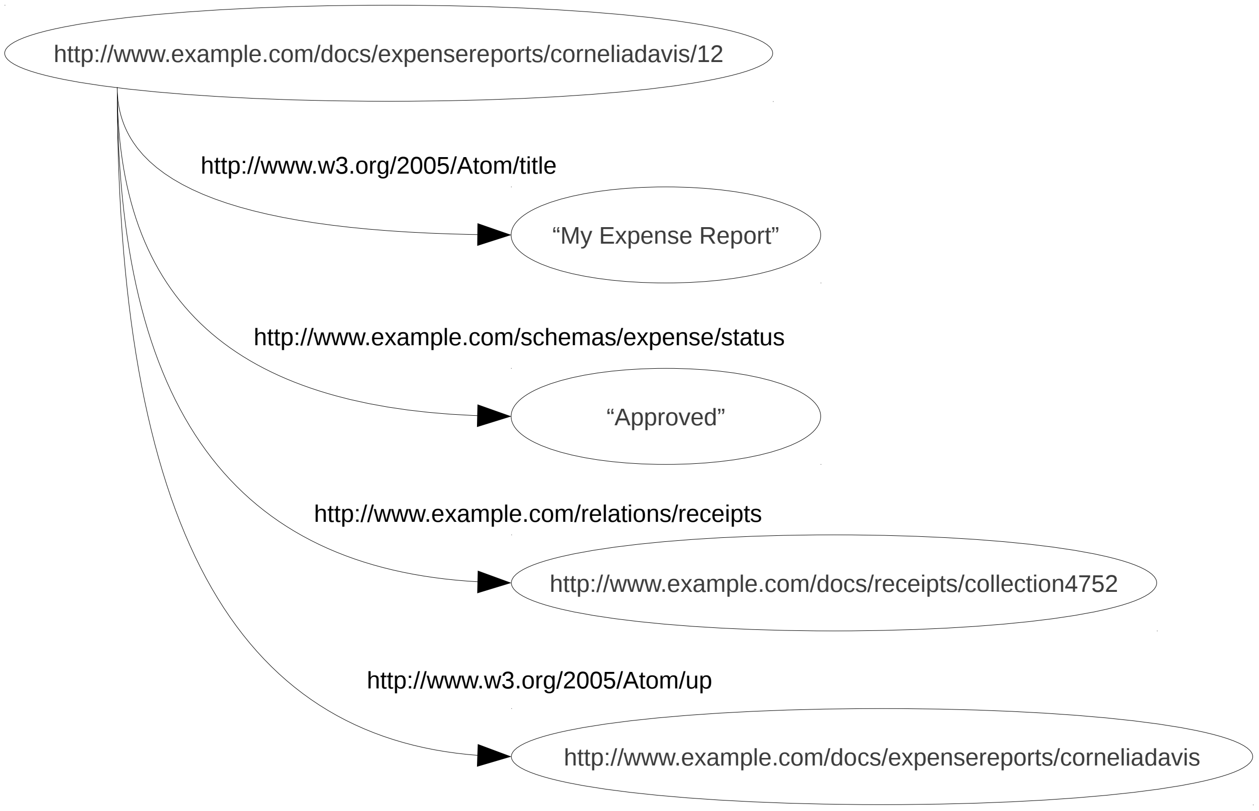
But Atom entries and feeds are not enough to give us linked data. Fortunately, Atom's focus on the web-based use case of blogging necessitated the inclusion of a link element in the Atom domain model and this forms the basis of our linked data approach. The `atom:link` takes the concept of a hyperlink, and adds relationship semantics (among other things) and while the Atom specification defines numerous attributes on the `atom:link` element, the two most important ones are `href` and `rel`. The `href` attribute value is a URL for a resource that is associated in some way to the resource represented by the entry or feed that the link appears in. The value of the `rel` attribute indicates the nature of this relationship and comes in one of two forms. In the event that the semantics of the relationship are commensurate with a standardized set of relationship types defined in the IANA Atom link relations registry [4], the short name published there may be used. If a different relationship type is required, it is designated with a full URI. The Atom-based resource representations that we serve through our RESTful services include link relations that use both standardized relationship types as well as some that we have defined specific to our usage domains.

One of the relationship types defined in the IANA registry has the short name “self” and it is used to supply a URL for the entity the link is embedded in. All of our Atom resource representations include this link relation, ensuring that all entities served as Atom entries and feeds have URIs exposed. Add to that namespace qualified entry and feed properties, as well as link relations to other resources, completes an abstraction that is roughly equivalent to the RDF triple. An Atom entry served by our RESTful services has a URI, as found in the “self” link relation. Some predicate URIs are given via XML namespace-qualified elements and others are supplied as rel values on atom:link elements. For the latter, a URI to the object of the predicate is given in the `href` value on the link relation.

To illustrate these concepts more concretely, consider the following Atom entry.

```
<?xml version="1.0" encoding="UTF-8"?>
<atom:entry xmlns:atom="http://www.w3.org/2005/Atom"
            xmlns:exp="http://www.example.com/schemas/expense">
    <atom:title>My Expense Report</atom:title>
    <atom:updated>2011-04-26T18:30:02Z</atom:updated>
    <atom:author>
        <atom:name>Cornelia Davis</atom:name>
    </atom:author>
    <atom:id>myidscheme:12345</atom:id>
    <atom:summary>Expenses from WWW conference</atom:summary>
    <atom:link rel="self"
       href="http://www.example.com/docs/expensereports/corneliadavis/12"/>
    <atom:link rel="up"
       href="http://www.example.com/docs/expensereports/corneliadavis"/>
    <atom:link rel="http://www.example.com/relations/receipts"
       href="http://www.example.com/docs/receipts/collection4752"/>
    <atom:content
       src="http://www.example.com/docs/expensereports/cdavis12.pdf"/>
    <exp:status>Approved</exp:status>
</atom:entry>
```

This atom entry represents a document stored in a content management system. The document is identified via the `href` value of the “self” Atom link relation; http://www.example.com/docs/expensereports/corneliadavis/12. Triples with this document as the subject include http://www.w3.org/2005/Atom/title, “My Expense Report” and http://www.w3.org/2005/Atom/updated, “2011-04-26T18:30:02Z” as predicate and object, respectively. Two other triples with expense report #12 as the subject have a predicate http://www.example.com/relations/receipts and object http://www.example.com/docs/receipts/collection4752, and a predicate “up” and an object http://www.example.com/docs/expensereports/corneliadavis. Because RDF triples require that the subject, object and predicate all be designated with a URI, we should prepend this literal with a URI. The Figure 1 shows a partial representation of the triples supplied in this Atom entry.


http://www.example.com/docs/expensereports/corneliadavis/12
http://www.w3.org/2005/Atom/title
"My Expense Report"
http://www.example.com/schemas/expense/status
"Approved"
http://www.example.com/relations/receipts
http://www.example.com/docs/receipts/collection4752
http://www.w3.org/2005/Atom/up
http://www.example.com/docs/expensereports/corneliadavis

Summary

We are not suggesting that the approach described herein, RESTful services providing highly linked, Atom-based resource representations, provides the full range of capabilities that the RDF, OWL and SPARQL-based technology stack does. We fully acknowledge that off the shelf reasoners can not be used against these data representations, for example. We are, however, suggesting that when the principles of REST are properly applied to service construction, the result is linked data that serves as an solution component for many of the data integration challenges we face today. The approach requires minimal "training" for our developer base and even resonates very well with customers who are also familiar with RSS and/or Atom, and hyperlinking on the World Wide Web.

It bears mention that this pattern does not require use the Atom format, rather any XML format that uses namespace qualified elements to represent resource properties, and a linking approach comparable to the `atom:link` will suffice. That said, the Atom-format has proven generic enough to represent a wide range of entity types and its consistent use eliminates one source of impedance mismatch when building integration solutions. Finally, even when non-Atom XML is used, we strongly advocate the use of the `atom:link` relation as defined in RFC 4287.

A RESTful services-based approach to linking data has been previously proposed [5] with less than enthusiastic response from some in the Linked Data community [6], yet we nevertheless submit this as a pragmatic approach to introducing linked data into the enterprise. We look forward to feedback and dialog.

References

[1] R. T. Fielding. Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, 2000.

[2] Fielding, R. T. REST APIs must be hypertext-driven. Blog post, October 2008. See http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven.

[3] Nottingham, M. and R. Sayre. The Atom Syndication Format. December 2005. See http://tools.ietf.org/html/rfc4287.

[4] IANA Link Relations Registry. Last updated September 2011. See http://www.iana.org/assignments/link-relations/link-relations.xml.

[5] Wilde, E., E. Kansa and R. Yee. Web Services for Recovery.gov, School of Information, UC Berkely, October 2009. See http://escholarship.org/uc/item/0fv601z8.

[6] Wilde, E. The Linked Data Police. Blog post, November 2009. See http://dret.typepad.com/dretblog/2009/11/the-linked-data-police.html.