A Data API with Security and Graph-Level Access Control

Barry Norton and Maciej Dziardziel

British Museum BNorton@britishmuseum.org [*]

Abstract. In this paper we bring together and extend two trends in building API and access control for RDF-based data in the form of an open source data API implementation. Parameterised SPARQL queries and updates are made available to form a RESTful API providing isolation from the underlying database in the style of most database-driven enterprise architectures. Access to query and update resources is governed by LDAP profiles. In effecting queries and updates, rewriting is employed to provide graph-level access controls, again according to LDAP groups.

1 Introduction

The notion of a 'public SPARQL endpoint' is prevelant among the Linked Data community. In the world of mainstream IT, however, the idea of an open query endpoint is baffling. Even within enterprise firewalls, developers are rarely able to execute arbitrary queries, let alone updates. Instead APIs are formed that encapsulate queries and updates; traditionally these have been SOAP and XML-based, but increasingly REST and representations such as JSON are the norm.

It has become common, alongside enterprise adoption of semantic technologies, that RESTful APIs are formed to encapsulate native RDF databases — often misleadingly-named 'triplestores' — which are just one of an arsenal that are now commonly used alongside the formerly-ubiquitous relational databases. Document-oriented XML and JSON-based databases, such as BaseX[1] and MongoDB[2] respectively, make it easy to form messages as these are identified with the stored documents. This is more involved with RDF, as transformation between stored statements entities and API content messages may be required; one has either to manually transform between representations, or choose direct RDF serialisations. Originally RDF/XML was the only choice for the latter, which consequently gained a very poor reputation. More recently Turtle [8] has grown in popularity, for instance in the approach of the W3C's Linked Data Platform Working Group [4]. More recently still a native JSON syntax that allows one to express RDF models, JSON-LD, has been standardised [7].

[*] This work is supported by the Andrew W. Mellon Foundation, and is carried out at the British Museum as part of the ResearchSpace project.

[1] http://basex.org/

[2] https://www.mongodb.org/

One would hope, therefore, that software frameworks that support RESTful, or at least pseudo-RESTful[3], API construction to easily encapsulate database queries and updates, like the plethora that exist for more established database technologies, would appear and would quickly adopt the new technologies, especially JSON-LD. In reality, those promising candidates, such as the BBC's Linked Data Platform[4] (unconnected with the W3C Working Group, and having prior claim to the name) and Talis' Kasabi platform [5], remain closed-source and have even, in the latter case, seemingly been discontinued with no further access available.

In light of this depressing situation, the ResearchSpace project has decided to implement a new and open source solution to fill this gap. The ResearchSpace Data API[5] is built using python and the Django framework and intends, from the start, to be enterprise grade, encompassing features such as LDAP-based security and access [6].

A second feature of the API is to provide access control over the RDF database. Unfortunately even after its second revision [10], SPARQL provided no means to provide security and access control, or even a suggestion on whether this should be carried out at database, graph or triple-level. It is worth mentioning that SPARQL 1.1 did introduce the Graph Store Protocol [9], which specifies access to named graphs in a RESTful manner, making these persistently-identified resources. Oddly the W3C Linked Data Platform has duplicated much of this work, calling graphs 'containers'. Adding access control directly to graphs with only CRUD (creative, retrieve, update, delete) operations, however, is insufficient to allow triple-level queries across graphs efficiently. By analogy with relational databases, APIs might be built over access control allowing certain users/groups access only to certain tables, but users are still able to (SQL) query across tables to which they are allowed access. Our contribution, therefore, is to associate LDAP groups with graph-based access control, allowing both complex queries across graphs and scalability to large numbers of graphs.

In order to explain our contributions further, we shall first introduce the ResearchSpace project, in Section 2, provide more details on the Data API in Section 3, and conclude, discussing further work, in Section 4.

2 ResearchSpace

ResearchSpace is an Andrew W. Mellon Foundation funded project aimed at developing an open source platform to support collaborative internet research and information sharing with Web-based applications for the cultural heritage scholarly community.

[3] I.e. HTTP-based, without the overhead of SOAP encapsulation, but without necessarily following REST principles [3].

[4] http://www.bbc.co.uk/blogs/internet/posts/Linked-Data-Connecting-together-the-BBCs-Online-Content

[5] http://stash.researchspace.org/projects/DATA

ResearchSpace will provide a range of flexible tools to support a wide range of workflows and will develop these tools on an ongoing basis. Semantic technology is at the core of the infrastructure because it provides an effective mechanism for research and collaboration across data provided by different organisations and projects. ResearchSpace aims to reduce the costs of developing and operating new and innovative systems, creating a more sustainable research and production environment and is committed to providing modular open source solutions to promote uptake of this technology in the cultural heritage sector.

At the centre of the ResearchSpace architecture is a SPARQL-compliant RDF database, better called a 'quadstore' than a 'triplestore' due to these standards, as every statement lives in a named graph. The base data are open datasets representing museum collections, modelled in the CIDOC-CRM ontology [2]. Currently, for instance, the ResearchSpace store contains the data of the British Museum [6] and the Yale Centre for British Art Collection [7], as well as growing numbers of other institutions', such as the RKD and Rijksmuseum from the Netherlands.

Since these datasets must be synchronised with internal, currently non-RDF, collection systems, the 'unit of update' when changes occur is the object record. For this reason named graphs are used in the first instance as containers for statements that concern single objects, and can be updated using the SPARQL Graph Store Protocol. The right-hand side of Figure 1 illustrates the graphs for a couple of prominent British Museum objects.

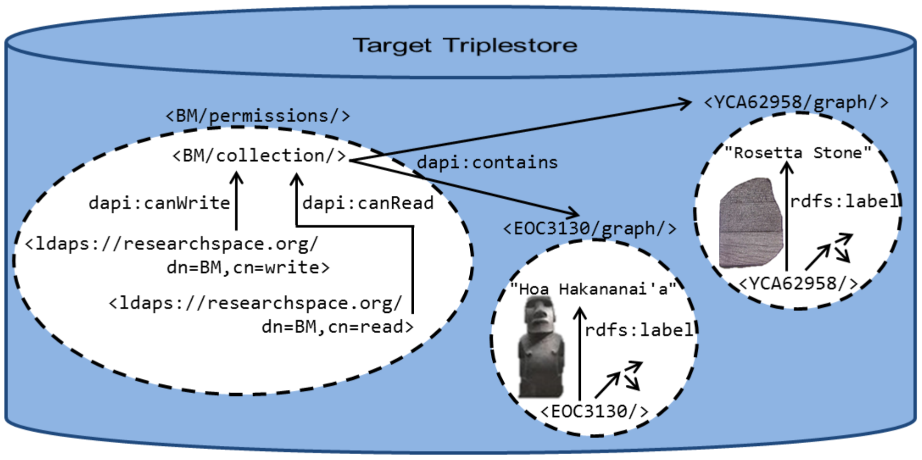
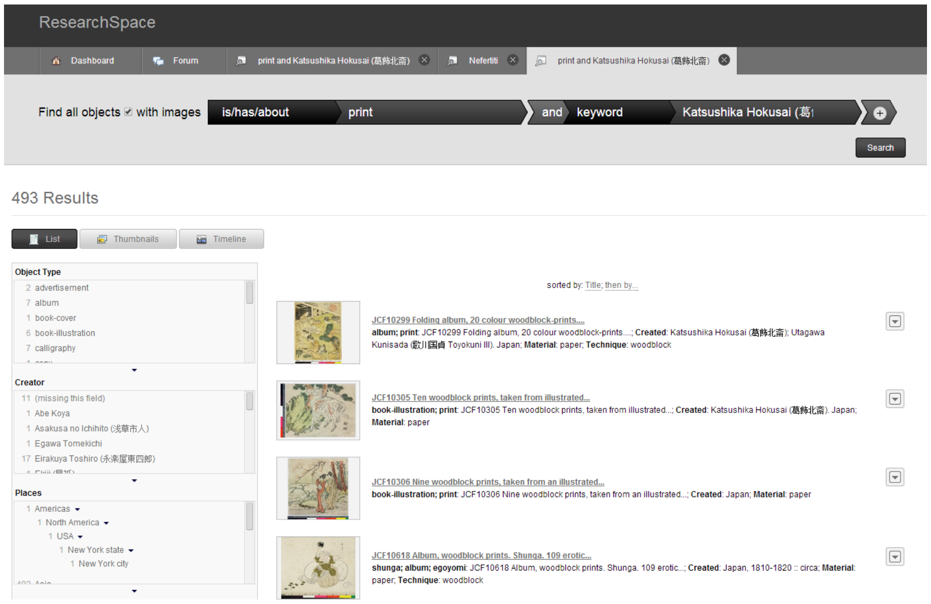


Fig. 1. Target Triplestore Example

[6] http://collection.britishmuseum.org/

[7] http://britishart.yale.edu/collections/using-collections/technology/linked-open-data



493 Results

2 advertisement
7 album
1 book-cover
6 book-illustration
7 calligraphy

11 (missing this field)
1 Abe Koya
1 Asakusa no Ichihito (浅草市人)
1 Egawa Tomekichi
17 Eirakuya Toshiro (永楽屋東四郎)

1 Americas
1 North America
1 USA
1 New York state
1 New York city

sorted by: Title; then by...

JCF10299 Folding album, 20 colour woodblock-prints....
album; print: JCF10299 Folding album, 20 colour woodblock-prints....; **Created**: Katsushika Hokusai (葛飾北斎); Utagawa Kunisada (歌川国貞 Toyokuni III). Japan; **Material**: paper; **Technique**: woodblock

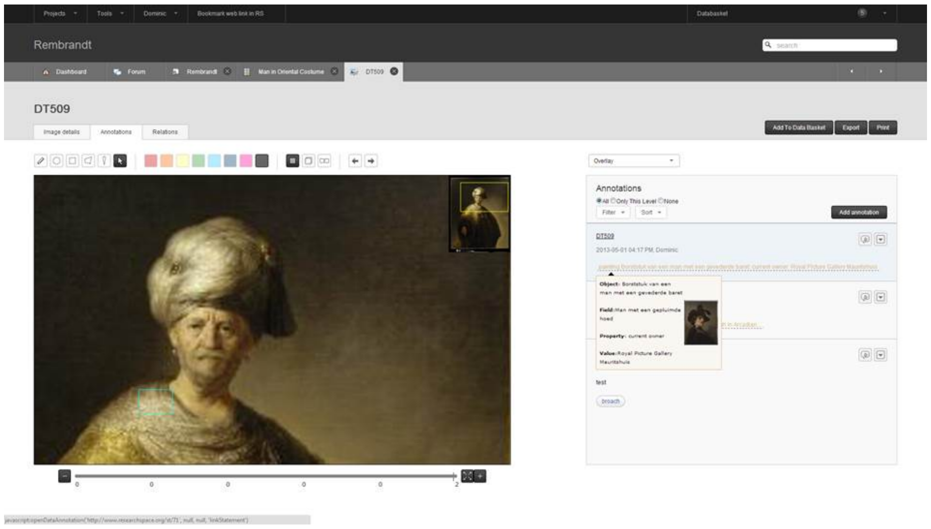
JCF10305 Ten woodblock prints, taken from illustrated...
book-illustration; print: JCF10305 Ten woodblock prints, taken from illustrated...; **Created**: Katsushika Hokusai (葛飾北斎). Japan; **Material**: paper

JCF10306 Nine woodblock prints, taken from an illustrated...
book-illustration; print: JCF10306 Nine woodblock prints, taken from an illustrated...; **Created**: Japan; **Material**: paper

JCF10618 Album, woodblock prints. Shunga. 109 erotic...
shunga; album; egoyomi: JCF10618 Album, woodblock prints. Shunga. 109 erotic...; **Created**: Japan, 1810-1820 :: circa; **Material**: paper; **Technique**: woodblock

Fig. 2. Search Component in the ResearchSpace Prototype

The ResearchSpace system provides powerful but intuitive query and update functionalities over this aggregated data, as illustrated respectively in Figures 2 and 3.



DT509

2013-05-01 04:17 PM, Dominic

Object: Borststuk van een man met een gevederde baret

Field: Man met een gepluimde hoed

Property: current owner

Value: Royal Picture Gallery Mauritshuis

test

Fig. 3. Image Annotation Component in the ResearchSpace Prototype

The Search component provides means to construct complex conjunctive queries — by building adding clauses using controlled vocabulary, via auto-complete — using abstracted properties — via dropdown, which is responsive to the relationship of the class of the chosen term and the range of the abstract properties, or , 'fundamental relationships' — and with provision for filtering — via faceting.

The Image Annotation component, together with others such as Data Annotation — where existing collection data can be challenged, extended, etc. — a Forum and Workflow system — where annotation can be discussed and linked, etc. All of these components depend on pre-defined parameterised queries, to retrieve existing data, and pre-defined updates, to add and update annotations, forum posts, etc.

3 The Data API

The Kasabi platform, like the BBC Linked Data Platform, chose to view such pre-defined and parameterised queries as RESTful resources. The queries can can be enacted by HTTP interactions, where the request includes values to bind to these parameters, which are free variables in the graph patterns of the query. Kasabi chose to call these 'SPARQL Stored Procedures' which is instructive of the overall approach of isolating the database query interface, but misleading in the sense of not involving non-SQL/relational algebra programming; we shall avoid this terminology. Like Kasabi the ResearchSpace Data API uses XML datatype to declare which variables in the query, or update, are intended to be substituted for at run-time, and which type of value is expected.

We extend this fore-going work in four important ways:

1. we expose and maintain our API implementation as open source, inviting community submissions;
2. instead of tying the access model to a specific platform, we use the LDAP standard;
3. we provide means to schedule queries and updates, together with automatic inspection of results for the former — in the form of XPATH for SELECT queries and SPARQL ASK queries for CONSTRUCT queries — and both an API to inspect runs and test results, together with email notifications of timings and test results for the purposes of monitoring;
4. we include graph-level security by query re-writing.

Figure 4 illustrates how an LDAP access model is used to bring together access to queries and updates, and access to the underlying data. Each user of the Data API must have an LDAP identity and, according to their group is allowed access to certain queries and updates. These are illustrated respectively by the ResearchSpace-level groups for read access to image annotations — whereby users like Maciej can view existing image annotations retrieved via pre-defined query — and write access to image annotations whereby users like Barry can furthermore change annotations via pre-defined updates.

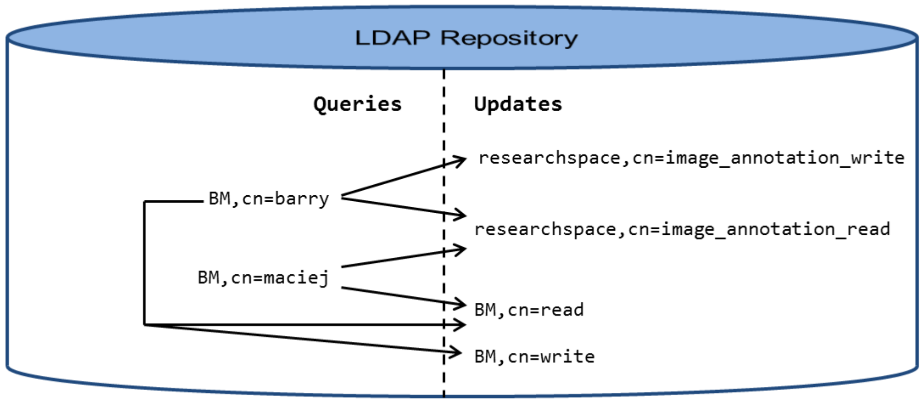

LDAP Repository
Queries
Updates
researchspace,cn=image_annotation_write
BM,cn=barry
researchspace,cn=image_annotation_read
BM,cn=maciej
BM,cn=read
BM,cn=write

Fig. 4. LDAP Example

Being allowed to execute a given query or update, however, does not mean that users in these groups have the ability to view or change arbitrary data. A second level of access control is enacted at run-time when the query or update is rewritten, before execution, according to the LDAP groups to which the requesting user is also a member of. Comparing Figures 4 and 1 we can see that the BM-specific groups 'read' and 'write' are represented in the triplestore (using the standard ldaps: URI scheme). The intuition behind this specific example is that while the image annotation component may be made available to certain ResearchSpace users, on either a read or a read/write basis, the actual objects whose annotations that user is thereby enabled to view or change may be specific to the project on which they work. In particular their project may provide access to non-public datasets.

Some existing open source approaches to (RDF) graph-level access control over SPARQL, such as the Shi3ld component [1] produced in the DataLift project[8], are based on enumeration of the graphs to which a user/group is allowed access, followed by naive expansion of the query via the addition of FROM clauses. This is infeasible in ResearchSpace due to the low granularity of named graphs. There are four million graphs in the British Museum collection alone; a rewrite that enumerated these would be rejected by most triplestores. On the other hand, if achieved by making an internal join within a query, this is quite feasible, so the Data API introduces *graph collections* to which LDAP groups are allowed access. This makes query expansion more involved, in case the query contains more than simply triple patterns, but still feasible and efficiently realised in a well-indexed RDF database, i.e. one where the named graph, or 'context', forms part of the indices.

[8] http://wimmics.inria.fr/projects/shi3ld/

As a trivial example we shall suggest that the image annotation component requires simply to query for the label of objects. The query shown in Listing 1.1 will be published to the LDAP group researchspace,cn=image_annotation_read, via the Data API, with a specification that the ?obj variable is a parameter that should be substituted at run-time with a URI.

```
SELECT ?label
WHERE {
   ?obj rdfs:label ?label
}
```

Listing 1.1. Example SPARQL Query

On behalf of any user allowed at least read-access to the image annotation component, via membership of this group, the component will execute the stored query, passing the URI of the object whose annotations the user wishes to view. The Data API will then rewrite the query to ensure that the user is allowed access to the particular data in question. For instance if Barry wants to view the annotations for Hoa Hakanai'a, the image annotation component would pass the parameter http://collection.britishmuseum.org/id/object/EOC3130, and the Data API would issue to the database the query shown in Listing 1.2.

```
SELECT ?label
WHERE {
   GRAPH ?g {<EOC3130> rdfs:label ?label} .
   GRAPH <BM/permissions/>
         {?group dapi:canRead/dapi:contains ?g} .
   FILTER(?group IN
     (<ldaps://researchspace.org/dn=BM,cn=read>
      <ldaps://researchspace.org/dn=BM,cn=write>))
}
```

Listing 1.2. Rewritten SPARQL Query

4 Conclusions and Future Work

In this paper we have presented a data API which provides for isolation of the SPARQL interface from developers and users, with group-based access policies. This takes inspiration from existing approaches to API construction, such as the BBC's Linked Data Platform and Talis' discontinued Kasabi platform, but is open source and builds on open standards. The Data API also provides graph-based data access control in a more scalable way than existing solutions such as Shi3ld.

In future work we shall provide Web-based administration which improves on Django's built-in administration interfaces for permissions, and which provides graphical reporting of both monitoring of scheduled queries and of ad hoc query and update usage logging.

5 Bibliography

References

1. Luca Costabello, Serena Villata, and Fabien Gandon. Context-aware access control for rdf graph stores. In *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 282–287. IOS Press, 2012.
2. Martin Doerr. The CIDOC CRM - an ontological approach to semantic interoperability of metadata. *AI Magazine*, 24:2003, 2003.
3. Roy Thomas Fielding. *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.
4. W3C Linked Data Platform Working Group. *Linked Data Platform 1.0*. W3C Last Call Working Draft, 11 March 2014. Available at http://www.w3.org/TR/2014/WD-ldp-20140311/.
5. Knud Möller and Leigh Dodds. The Kasabi information marketplace. In *21nd World Wide Web Conference (WWW2012)*, 2012.
6. Network Working Group. *Lightweight Directory Access Protocol (LDAP): The Protocol*. The Internet Society, June 2006. Available at https://tools.ietf.org/rfc/rfc4511.txt.
7. W3C RDF Working Group. *JSON-LD 1.0: A JSON-based Serialization for Linked Data*. W3C Recommendation, 16 January 2014. Available at http://www.w3.org/TR/2014/REC-json-ld-20140116/.
8. W3C RDF Working Group. *RDF 1.1 Turtle: Terse RDF Triple Language*. W3C Recommendation, 25 February 2014. Available at http://www.w3.org/TR/2014/REC-turtle-20140225/.
9. W3C SPARQL Working Group. *SPARQL 1.1 Graph Store HTTP Protocol*. W3C Recommendation, 21 March 2013. Available at http://www.w3.org/TR/2013/REC-sparql11-http-rdf-update-20130321/.
10. W3C SPARQL Working Group. *SPARQL 1.1 Query Language*. W3C Recommendation, 21 March 2013. Available at http://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

6 Acknowledgements

The Principle Investigator of the ResearchSpace project, Dominic Oldman, provided support to this development at the British Museum, and provided valuable feedback on this report.