Distributed Linked Data Business Communication Networks: The LUCID Endpoint

Sebastian Tramp[1(✉)], Ruben Navarro Piris[1], Timofey Ermilov[1], Niklas Petersen[2], Marvin Frommhold[1], and Sören Auer[3]

[1] eccenca GmbH, Hainstr. 8, 04109 Leipzig, Germany
{sebastian.tramp,ruben.navarro.piris,timofey.ermilov}@eccenca.com
[2] Enterprise Information Systems (EIS) at the Institute for Applied Computer Science at University of Bonn, Römerstr. 164, 53117 Bonn, Germany
petersen@cs.uni-bonn.de
[3] Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), Schloss Birlinghoven, 53757 Sankt Augustin, Germany
soeren.auer@iais.fraunhofer.de

Abstract. With the LUCID Endpoint, we demonstrate how companies can utilize Linked Data technology to provide major data items for their business partners in a timely manner, machine readable and with open and extensible schemata. The main idea is to provide a Linked Data infrastructure which enables all partners to fetch, as well as to clone and to synchronize datasets from other partners over the network. This concept allows for building of networks of business partners much like as social network but in a distributed manner. It furthermore provides a technical infrastructure for business communication acts such as supply chain communication or master data management.

1 The LUCID Endpoint

The LUCID endpoint[1] provides the necessary technology stack to manage and publish Linked Data, as well as to consume Linked Data from other LUCID endpoints.

This includes authentication and authorization mechanisms to guarantee that data consumers access only the data that the endpoint owner explicitly allows. To ensure consumer authentication, OAuth2 [6] is used. Access control rules can be defined on a named graph level.

Once a local graph was modified, the endpoint will notify its subscribers by sending the latest change sets for inclusion[2]. An example of this approach is depicted in Fig. 1. By sending only the modifications instead of the complete

[1] Which is based on the eccenca Linked Data Suite Backend.

[2] The overall process is compatible with the PubSubHubbub working draft v0.4 [4]. The specification of a more general architecture for this kind of distributed semantic social networks is available as [8].

F. Gandon et al. (Eds.): ESWC 2015, LNCS 9341, pp. 154–158, 2015.
DOI: 10.1007/978-3-319-25639-9_30

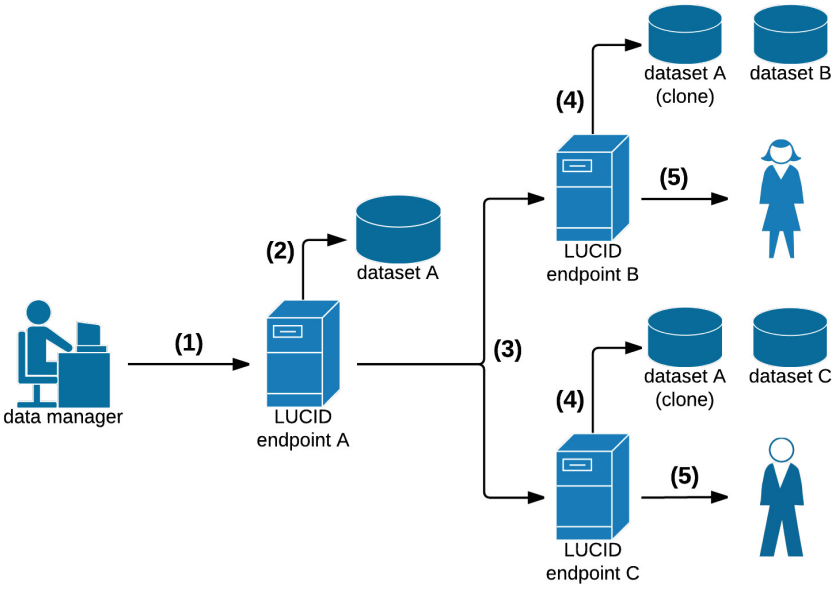


Fig. 1. The LUCID endpoint data management and publication process consists of the following steps: **(1)** modification of a dataset over SPARQL or a GUI **(2)** update of the dataset as well as revisioning **(3)** publication of the updates to all subscribers **(4)** application of the changes to the cloned datasets on subscriber site **(5)** user notification

new dataset, subscribers can easily recognize the changed triples without the need for calculation of expensive data diffs itself. Subscriber endpoints can then apply those modifications to their local dataset clones automatically. In order to describe these dataset changes, a vocabulary and exchange format is needed, which we will explain in the next section.

2 The Eccenca Revision Vocabulary

In order to both keep track of the modifications on the local quad store and notify subscribers of it about those modifications, we developed the eccenca Revision Vocabulary[3]. This vocabulary is modelled using OWL (OWL 2 DL profile) and extends as well as reuses several concepts of the PROV-O ontology [7].

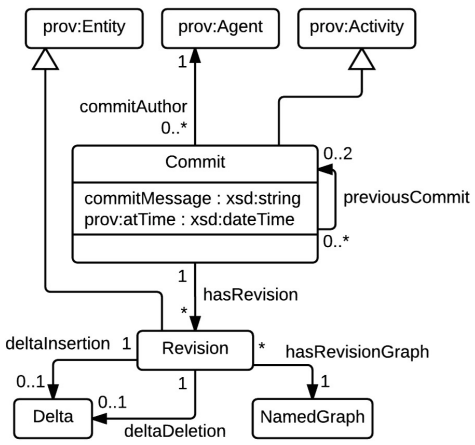
Unlike other approaches, such as [1], which try to describe changes on higher semantic levels, our approach is based on triple (or rather quad) changes, where each revision or modification event (called commit) contains a diff representing the changed (either inserted and/or deleted) quads. This simple model enables applications to rebuild and revert each commit as well as to merge diverted evolution branches as explained in [3].

[3] The eccenca Revision Vocabulary is available at https://vocab.eccenca.com/revision/.

Our data modelling approach is build on top of the one proposed in [5], but instead of holding separate revision histories for each revisioned named graph, our approach keeps a unified revision history on any number of named graphs. This enables applications to track revisions across different graphs or for the whole quad store.

Figure 2 illustrates the main parts of the vocabulary: The `Commit` class defines an instantaneous event containing a set of graph revisions. This class contains also the meta data associated to this event such as author, date and commit message. Revisions (modelled as the `Revision` class) refer each to a specific named graph which was changed. Changes in an RDF store are defined either as triple insertions (`deltaInsertion`) or deletions (`deltaDeletions`) inline with the approach in [2].

Further work to support branching, commit signing and blank nodes is in progress.




```
@prefix     : <https://vocab.eccenca.com/revision/>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix prov: <http://www.w3.org/ns/prov#>.
@prefix   ex: <http://example.com/>.

{ ex:CommitA a :Commit ;
    :commitAuthor   ex:RubenNavarroPiris ;
    prov:atTime     "2015-03-12T13:37:00+01:00" ;
    :commitMessage  "change label and add commit" ;
    :previousCommit ex:CommitB ;
    :hasRevision    ex:RevisionA1, ex:RevisionA2 .

  ex:RevisionA1 a :Revision ;
    :hasRevisionGraph ex:NamedGraphX ;
    :deltaInsert      <insertA> ;
    :deltaDelete      <deleteA> .

  ex:RevisionA2 a :Revision ;
    :hasRevisionGraph ex:NamedGraphY ;
    :deltaDelete      <insertB> .
}

<insertA> { ex:AResource rdfs:label "foo". }
<deleteA> { ex:AResource rdfs:label "bar". }
<insertB> { ex:AResource rdfs:comment "my comment". }
```

Fig. 2. LUCID revision vocabulary & example commit instance

3 Demonstration Use-Case: Master Data Management

Our setup for the demonstration of the LUCID endpoint deploys a very basic but pressing use case in business to business communication: master data management. Enterprise master data is the single source of basic business information used across all enterprise systems, applications and processes for an entire enterprise. This includes resources such as persons, company sites and subsidiaries as well as contact details.

Our proposed demo consists of the following parts:

– Publishing of master data datasets with a browser based user interface: A LUCID endpoint provides a dataset for each account. The account owner

is free to upload any data to this dataset. All resources from the dataset namespace are available as Linked Data and enabled for publish/subscribe as well as OAuth (in case the dataset is non-public). In addition to the generic access via SPARQL, the user can utilize a master data management application. This single page JavaScript application allows for creation of master data resources such as company subsidiaries and contact details. The RDF data model for these resources is based on the master data model from Odette International, a collaboration platform for the automotive supply chain.

– Versioning of the dataset changes on the SPARQL endpoint backend: All changes to the user dataset are logged as part of the internal LUCID endpoint triple store. The changed triples are calculated directly by the SPARQL query processor and added to the versioning store.
– Subscription to datasets of another LUCID endpoint by employing the dataset URL: All resources which are Linked Data accessible, are enabled for publish/subscribe activities as well. The user interface is able to manage subscriptions to other endpoints as well as to provide a preview for the incoming data.
– A publish/subscribe mechanism which uses commit push notifications based on the eccenca revision vocabulary described in Sect. 2: For each resource, a change log dataset is available, which provides the last `Commit` information. In addition to that, notifications with these `Commit` information as payload are pushed to all subscribers in case of a change. The subscribing endpoint adds the incoming data to its dataset clone as well as hold the change log in order to provide versioning information to the user.

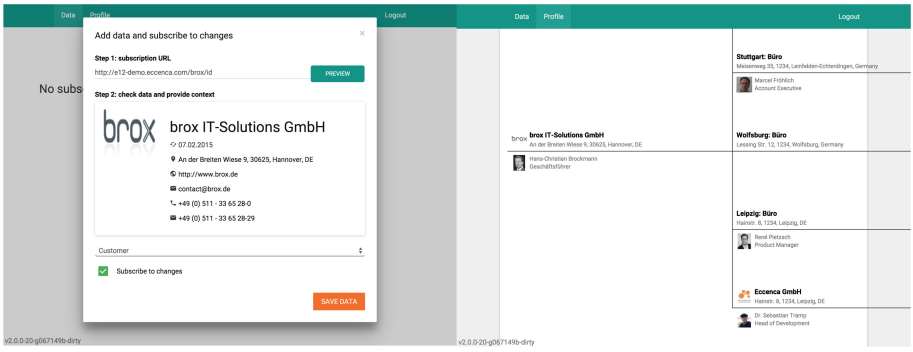


Fig. 3. Screenshots of the master data management user interface: (left) Any user can subscribe to changes of other datasets by employing the subscription URL provided by the publisher. After committing the subscription process, the current version of the data model is fetched with an HTTP Linked Data request. (right) The publisher of a dataset is able to create its company master data which includes sites, contacts and other structures by using the master data manager. The master data manager is a browser-based user interface to an OAuth2 [6] enabled SPARQL endpoint.

Figure 3 depicts two screenshots of the master data management user interface which lies on top of the versioning and OAuth2 enabled SPARQL endpoint[4].

Acknowledgement. This work was partly supported by a grant from the German Federal Ministry of Education and Research (BMBF) in the IKT 2020 funding programme (GA no. 01IS14019) for the LUCID Project (http://lucid-project.org).

References

1. Auer, S., Herre, H.: A versioning and evolution framework for RDF knowledge bases. In: Proceedings of Ershov Memorial Conference (2006)
2. Berners-lee, T., Connolly, D.: Delta: an ontology for the distribution of differences between RDF graphs. Technical report, W3C (2004). http://www.w3.org/DesignIssues/lncs04/Diff.pdf
3. Cassidy, S., Ballantine, J.: Version control for RDF triple stores. In: Filipe, J., Shishkov, B., Helfert, M. (eds.) Proceedings of the Second International Conference on Software and Data Technologies, ICSOFT 2007, ISDM/EHST/DC, 22–25 July 2007, Barcelona, Spain, pp. 5–12. INSTICC Press (2007)
4. Fitzpatrick, B., Slatkin, B., Atkins, M., Genestoux, J.: PubSubHubbub Core 0.4. Working draft, PubSubHubbub W3C Community Group (2013). https://pubsubhubbub.googlecode.com/git/pubsubhubbub-core-0.4.html
5. Graube, M., Hensel, S., Urbas, L.: R43ples: revisions for triples - an approach for version control in the semantic web. In: Knuth, M., Kontokostas, D., Sack, H. (eds.) Proceedings of the 1st Workshop on Linked Data Quality Co-located with 10th International Conference on Semantic Systems, LDQ@SEMANTiCS 2014, 2nd September 2014, Leipzig, Germany, vol. 1215. CEUR Workshop Proceedings. CEUR-WS.org (2014)
6. Hardt, D.: The OAuth 2.0 Authorization Framework. RFC 6749, IETF, October 2012. https://tools.ietf.org/html/rfc6749
7. Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J.: PROV-O: The PROV Ontology. W3C Recommendation, W3C, April 2013. http://www.w3.org/TR/prov-o/
8. Tramp, S., Frischmuth, P., Ermilov, T., Shekarpour, S., Auer, S.: An architecture of a distributed semantic social network. Semant. Web **5**(1), 77–95 (2014)

[4] An annotated demonstration video is available at http://downloads.eccenca.com/2015/03/13/eswc2015-lucid-demo.mp4.