INFRA-2011-1-284432

COLLABORATIVE EUROPEAN DIGITAL ARCHIVE INFRASTRUCTURE

Project Acronym: CENDARI
Project Grant No.: 284432
Theme: FP7-INFRASTRUCTURES-2011-1
Project Start Date: 01 February 2012
Project End Date: 31 January 2016

Deliverable No. :	9.1
Title of Deliverable:	Prototype for search and faceted search tools
Date of Deliverable:	June 2014
Revision No.:	1
WP No.:	6
Lead Beneficiary:	University of Göttingen (UGOE)
Author (Name and email address):	Jean-Daniel Fekete Jean-Daniel.Fekete@inria.fr Nadia Boukhelifa Nadia.Boukhelifa@inria.fr Evanthia Dimara Evanthia.Dimara@inria.fr Carsten Thiel thiel@sub.uni-goettingen.de
Dissemination Level:	PP = restricted to other programme participants
Nature of Deliverable:	O = other
Abstract (150 words):	Prototype for search and faceted search tools: To be created using data samples that showcase results by the data integration. This report provides an overview of the CENDARI infrastructure for search and faceted search tools. It relies on a distributed search engine, ElasticSearch, a search and faceted search system, XMLFacets, and a set of rules and conventions to allow all the tools built or used by the CENDARI platform to update the distributed search

	environment. The environments instrumented to work with this search tool are: 1) The repository of XML documents 2) The note-taking environment 3) The MediaWiki system Others will be integrated in a continuous fashion, based on our Agile Development Methodology. Compared to other search and faceted search environments, our tools should adapt to a large variety of programs managing their own data. Therefore, it has to be open to complex data schemata while still enforcing some unification of metadata to allow searching and visualizing a wide variety of resources.

Table of Contents

Introduction 4
CENDARI Services to Integrate 5
Searching and Faceting 6
Indexing 6
Search 6
Aggregation and Faceting 6
Aggregation Features for Visualization 7
The XMLFacets Program 7
Standard Index Structure in XMLFacets 8
Management of XML documents 9
Archives in Europe encoded as EAG Documents 10
Archival Descriptions encoded as EAD Documents 10
Text Encoding Initiative encoded documents 11
Imported Archive/Library Catalogs and Portals (Text, PDF, HTML, or XML) 11
CENDARI MediaWiki Contents (Text, RDF) 11
CENDARI Note-Taking Environment (Text, RDF) 11
Updating from the Data API 12
The User Interface 13
Visualizations 14
Editing 14
Conclusion 14
References 16
Appendix A: ElasticSearch Mapping 17
Appendix B: ElasticSearch Filter with Permission 18

Introduction

This report provides an overview of the technical components required to support search and faceted search for the CENDARI infrastructure project.
In CENDARI, data arrives from multiple sources, either by harvesting external repositories such as library catalogs and collections, or archives catalogs. Data also arrives from CENDARI partners, well curated under the authority of project historians, such as the description of archives entered in EAG format by CENDARI partners. Data is also newly created by CENDARI participants and partners, such as the Archival Research Guides (ARG) mainly gathered from WP5 with the help of international specialists. It is also collected from well-known data repositories, such as the TRAME portal[1] (Manuscript texts and traditions of the European Middle Ages) maintained by medievalists and their institutions. Finally, it is also collected from notes taken by historians through the CENDARI Note-Taking environment.

All this information arrives in various formats, more or less structured, sometimes with images, manuscripts scans, or even videos. The Faceted Search Service described in this document is aimed at allowing easy search and extraction of data relevant for historians, whatever its initial format, provenance, and status.

While many systems exist that provide faceted search for a specific kind of document, e.g. XML-encoded text (XTF[2]), they are not suited for a wide-variety of formats. Besides, XTF suffers from limitations in adding information dynamically. On the other side, new technologies are arriving to provide high-performance text-based search, document indexing, and faceted browsing (Solr[3], ElasticSearch[4]), but they are not usable directly for the CENDARI services, they need some surrounding application or framework to prepare data for ingestion in the right format, as well as extraction and presentation for users.

The CENDARI Faceted Browsing and Search environment is an implementation of a Faceted Search Engine over ElasticSearch, meant to unify the search and exploration of the resources provided by the CENDARI project. It is aimed at historians, archivists, and history enthusiasts interested by CENDARI resources.

It also provides limited editing capabilities for XML and textual documents to allow corrections and updates without resorting to external applications when possible.

[1] http://trame.fefonlus.it
[2] http://xtf.cdlib.org/
[3] http://lucene.apache.org/solr/
[4] http://www.elasticsearch.org

CENDARI Services to Integrate

Currently, CENDARI aims at integrating several types of documents that are quite heterogeneous and maintained by various authorities:

- XML documents
  - Archives in Europe encoded as EAG Documents
  - Archival Descriptions encoded as EAD Documents
  - Text Encoding Initiative encoded documents
- Imported Archive/Library Catalogs and Portals (Text, PDF, HTML, or XML)
- CENDARI MediaWiki Contents (Text, RDF)
- CENDARI Note-Taking Environment (Text, RDF)

Furthermore, CENDARI maintains a triple-store with RDF resources gathered from external sources such as DBPedia[5] or Freebase[6] for enriching its internal resource base. Therefore, faceted searching should provide access to all these resources seamlessly and in an extensible way.

The services provided by the search and faceted search system are not limited to simple human queries or faceted browsing, it also encompasses multiple applications-specific queries, either for searching inside a particular applications (e.g. MediaWiki[7]) or to use special format for the output, such as visualizations or structured-data extraction.

The current implementation of the Faceted Browsing environment is a web service that allows users to browse a large database of documents using faceted browsing. It relies on Django[8] connected to ElasticSearch to manage the faceting functionality. The current configuration of the service allows users to search and browse an XML database using the following facets: Theme (e.g. WWI, Medieval time), Country, Tag, Language and Creator. More facets will be added when new services will be connected, but the service is already running, indexing and faceting on XML resources gathered or created by CENDARI.

We will return to these services once the mechanisms used for searching and faceting are described.

[5] http://wiki.dbpedia.org/
[6] https://www.freebase.com/
[7] https://www.mediawiki.org/
[8] https://www.djangoproject.com/

Searching and Faceting

To allow fast, scalable, and flexible searching and faceting, a search engine should be used. Among the several engines available, we have chosen ElasticSearch for several reasons:

1) It uses a modern architecture that is both efficient and scalable
2) It is usable as a web service, facilitating its connection with CENDARI applications that are also web oriented
3) It provides aggregation functions that are extremely important to control the amount of data sent from the indexer to applications, in particular for visualizations presented in web browsers.

Indexing

Search engine store data to retrieve it very quickly. ElasticSearch maintains two kinds of data, literals and text. Literals are meant to be searched as they are (integer values, float values, dates, simple strings) whereas text needs some preprocessing to allow full-text search and suggestion of word completion. Modern search engines perform the two kinds of search efficiently, but some kind of schema should be defined to explain the search engine how data inserted should be treated in term of indexing and searching. This is what is called a “mapping” in ElasticSearch terminology. We provide the CENDARI mapping in Appendix A. The originality of our approach is that, in most cases, the mapping is defined by each application to fit their needs. In our case, we need to index all CENDARI’s applications using the same mapping to allow global search/aggregation/faceting.

Search

Once indexing is done efficiently, search is also efficient. However, in CENDARI, information should be found efficiently but with some privacy constraints. ElasticSearch can perform search very efficiently provided that the queries are well planned and we describe how queries and filters should be done in Appendix B.

Aggregation and Faceting

Text search is popular thanks to web search engines; they mostly search based on free-text. For CENDARI free-text search is important, but more structured search is also essential. This is why each record maintained by the search engine can define “facets” and search results should provide information about the facets, and allow drilling down according to these facets. Faceting is a special case of aggregation; general aggregation is also essential for CENDARI for scalability and visualization.

Faceting is the ability to answer queries by giving an overview of the results broken-down by "facets". For example, the CENDARI data store contains documents of various types such as XML, PDF, HTML, TXT. A search request could retrieve thousands of matching documents so only a few can be displayed. However, the number of documents in each type can be displayed in the "type" facet. Other facets can display document size distributions, creation dates, language, etc. The whole document set can be summarized by all its facets, showing an overview of the whole CENDARI repository. Furthermore, some facets are best shown using visualizations, such as histograms for e.g. sizes or dates, and maps for places (provided some geo-location is available).

Aggregation Features for Visualization

Compared to competing systems such as Solr[9], the aggregation functions working on time-series and geo-localized entities are much more scalable in ElasticSearch. To illustrate the importance of the functionality, imagine the CENDARI repository contains 1 million documents with dates. We could collect all the dates to visualize them, send them to the web browser that would compute a histogram of dates. However, sending one million dates over a regular internet connection would take minutes: the interface would be unusable.

Server-side aggregation allows the server to return directly a histogram to an aggregation request; in our example, we need to specify how many bins we want (e.g. 100) and the search engine will return a table with 100 entries containing the dates of each bin, and the count of documents for each bin. This can be sent in a fraction of a second for display to the web browser.

In addition to histogram aggregation for values (e.g. document sizes) and dates, ElasticSearch provides aggregation capabilities for geo-localized points: it also returns the count of geo-localized objects under a geographical grid with a user-specified size. Therefore, whatever the number of documents in the index, an aggregated geographical query will return a small result, adaptable to the connection speed and screen size.

The XMLFacets Program

The main interface to the search and faceted search environment is XMLFacets, a web-based system programmed in the Python language using the Django application framework. It is available on Github[10].

[9] http://solr-vs-elasticsearch.com/
[10] https://github.com/CENDARI/xmlfacets

The program is meant to evolve as more data types will populate the CENDARI repositories, and more information extraction will be achieved on this data. Still, it currently allows querying by facet or by full-text search on a large number of documents, up to millions potentially. The main issue we need to address is the scalability of the search results. As much as the search engine can manage large numbers of documents, a regular text-based search query interface cannot cope with more than 50-100 results. Thus, faceting and visualization is used to go beyond that limit.

Standard Index Structure in XMLFacets

ElasticSearch defines a small number of required fields each time it indexes a document. We use them internally. These fields are:

1) Unique ID: supplied or generated internally
2) Internal Type: the name of an ElasticSearch schema, which is a logical format

Our top-level index in ElasticSearch, similar to a database, is called "CENDARI" and contains all the indexed data.

We rely on the Dublin Core metadata specifications[11] to define our schema. ElasticSearch can work without a schema, but specifying the exact type of operations we want to perform with the indexed fields improves greatly its performance. The Dublin Core standard items are:

1) Contributor: An entity responsible for making contributions to the resource.
2) Coverage: The spatial or temporal topic of the resource, the spatial applicability of the resource, or the jurisdiction under which the resource is relevant.
3) Creator: An entity primarily responsible for making the resource.
4) Date: A point or period of time associated with an event in the lifecycle of the resource.
5) Format: The file format, encoded as a MIME (Multi-Purpose Internet Mail Extensions) format.
6) Language: A language of the resource.
7) Publisher: An entity responsible for making the resource available.
8) Relation: A related resource.
9) Rights: Information about rights held in and over the resource.
10) Source: A related resource from which the described resource is derived.

[11] http://dublincore.org/documents/dces/

11) Subject: The topic of the resource.
12) Title: A name given to the resource.
13) Type: The nature or genre of the resource.

Additionally, we define several faceted fields, usable by any object that is indexed:

1) Url: the URL of the related document on the CENDARI system or elsewhere.
2) Application: the application that created the data, such as MediaWiki, CENDARI-data, Note-Taking Environment, etc.
3) Text: the full text used for full-text indexing/searching
4) Length: the length (in bytes or characters) of the document.
5) Place: an array of place names mentioned in the document. If possible, the place name should be the dbpedia entry name for the place when it exists.
6) Event: an array of event names, again following dbpedia if possible.
7) Person: an array of person names, also following dbpedia if possible.
8) Tag: an array of tag names.
9) Org: an array of organization names, also following dbpedia if possible.
10) Artifact: an array of artifact names (e.g. weapon, train, tank), also following dbpedia if possible.
11) Ref: an array of canonical references (e.g. ISBN numbers, DOI, URIs, etc.)

Finally, access rights are managed through two fields:

1) Groups_allowed: the list of groups that can access the resource for reading. Null means that the access is not restricted to any group.
2) Users_allowed: the list of users that can access the resource for reading. Null means that the access is not restricted to any user.

Specific documents can add other fields if needed, but they also need to let XMLFacets know of these fields or they will be ignored in generic searches. They can still be used by specific applications.

Applications, such as the VRE, using the “CENDARI” index should, in turn, comply with the security requirement by checking groups and users to avoid disclosing unwanted contents.

Management of XML documents

XML documents come from multiple origins in the CENDARI repository. They are stored in the Data Store and collected through the DATA API. Therefore, the program XMLFacets crawls the Data Store to collect and update XML documents for indexing them.

One issue with XML documents is that textual search should be done inside their "displayed" contents, not the raw XML contents. Therefore, the XMLFacets program needs to manage XML documents according to their Schema through a "style sheet". This style sheet is used to convert the XML into text or HTML-rendered text that can then be indexed by the search engine. The XML Schema and style sheets are extracted according to the standard XML methods. First, the document header is searched for processing instructions declaring the Schema and style sheet. If they are specified, they are used in priority to the generic rules.

If the Schema is missing, a table of well-known schema associated with the opening tag is maintained by the XMLFacets program. This association is part of a database table available in the program that can be extended and enriched and possibly externalized for use by other components of the CENDARI infrastructure. The same mechanism is provided for the style sheet. At this moment, we hope this mechanism will be enough to maintain the indexes, but we foresee some difficulties with some XML formats known to be complex.

Archives in Europe encoded as EAG Documents

The CENDARI project has produced several EAG (Encoded Archival Guide) (APEnet, 2011-a) documents to describe most of the archives in Europe. The encoding is controlled by the project and the style sheet is also available and maintained by the CENDARI project WP7. These documents are easy to index but they are updated frequently and require XMLFacets to visit the Data Store to chase for changes. This is likely to change once the DATA API will provide a notification mechanism but, until then, polling is used every hour to update changes.

Archival Descriptions encoded as EAD Documents

EAD (Encoded Archival Description) (APEnet, 2011-b) documents are produced routinely by archives and libraries; however, they are complex to manage due to the large set of incompatible interpretations of the standard and local variations. Therefore, the CENDARI project is faced with several options to handle these documents and the final decision has not been made yet. One option would be to "standardize" the EAD documents CENDARI receives. However, as the work involved seems daunting and resources are insufficient to do it manually, this can not be realized. Another option consists of ignoring at least part of the structure more or less entirely and apply automated entity extraction to it in order to get as much information as possible without manual intervention. The CENDARI project will explore this option further, trying to define common EAD elements for faceting and search and explore further options. But for the time being, XMLFacets has to use a

very simple style sheet to get textual contents at first before all the "smart" extractions work effectively.

Text Encoding Initiative encoded documents

TEI (Text Encoding Initiative) (TEI Consortium, eds.) XML documents are gathered from several sources. In CENDARI, the XML files come mostly from Europeana[12] and the Manuscriptorium[13], as well as other sources. The main problem with TEI files is that they cannot be rendered into text easily: there is usually one style sheet for each class of encoding. We still apply our heuristic to find the appropriate style sheet and, if not specified in the document, we use a simplistic style sheet that tries to guess what elements to show.

Imported Archive/Library Catalogs and Portals (Text, PDF, HTML, or XML)

The CENDARI Data Store contains a large quantity of other types of documents, gathered by multiple participants for various uses. We already described our heuristic for XML documents, the other types are mostly textural and we can extract the text to feed it to the indexer. Ideally, in the future, we should also try to extract named entities to facilitate search, but the automatic extraction will start later when the raw indexing will be finalized.

CENDARI MediaWiki Contents (Text, RDF)

The CENDARI MediaWiki[14] is connected with the indexer and changes in articles are reflected immediately. The semantic part should be connected too in a simple way: the RDF entities related to the supported type of entities will be associated with the MediaWiki pages. MediaWiki searches could then be limited to the contents of the MediaWiki or extended to all the CENDARI indexed contents. By default, the MediaWiki search will only mention wiki-internal documents but we will experiment together with our users, to see whether this default behavior is really better than the more open approach of always extending the search by default.

CENDARI Note-Taking Environment (Text, RDF)

The CENDARI Note-Taking environment[15] also produces text (XHTML) and entities. It is already connected to the index engine and has been used to experiment with the

[12] http://www.europeana.eu/
[13] http://www.manuscriptorium.com/
[14] https://wiki.cendari.dariah.eu/wiki/Main_Page
[15] https://pro2.cendari.dariah.eu/enotes/

mapping described in this document. The index is maintained in real-time and allows searching inside of the note-taking environment, as well as outside if the users desire so.

Updating from the Data API

Currently, an external command is used periodically to re-index the contents of the Data API. Currently, the frequency is every hour.

Polling is a simple mechanism that can be generalized to multiple sources if needed so we will continue to support it. However, in the future, we will also rely on notification mechanisms implemented by the Data API to update the index quickly after a document has been added, modified, or deleted.

The User Interface

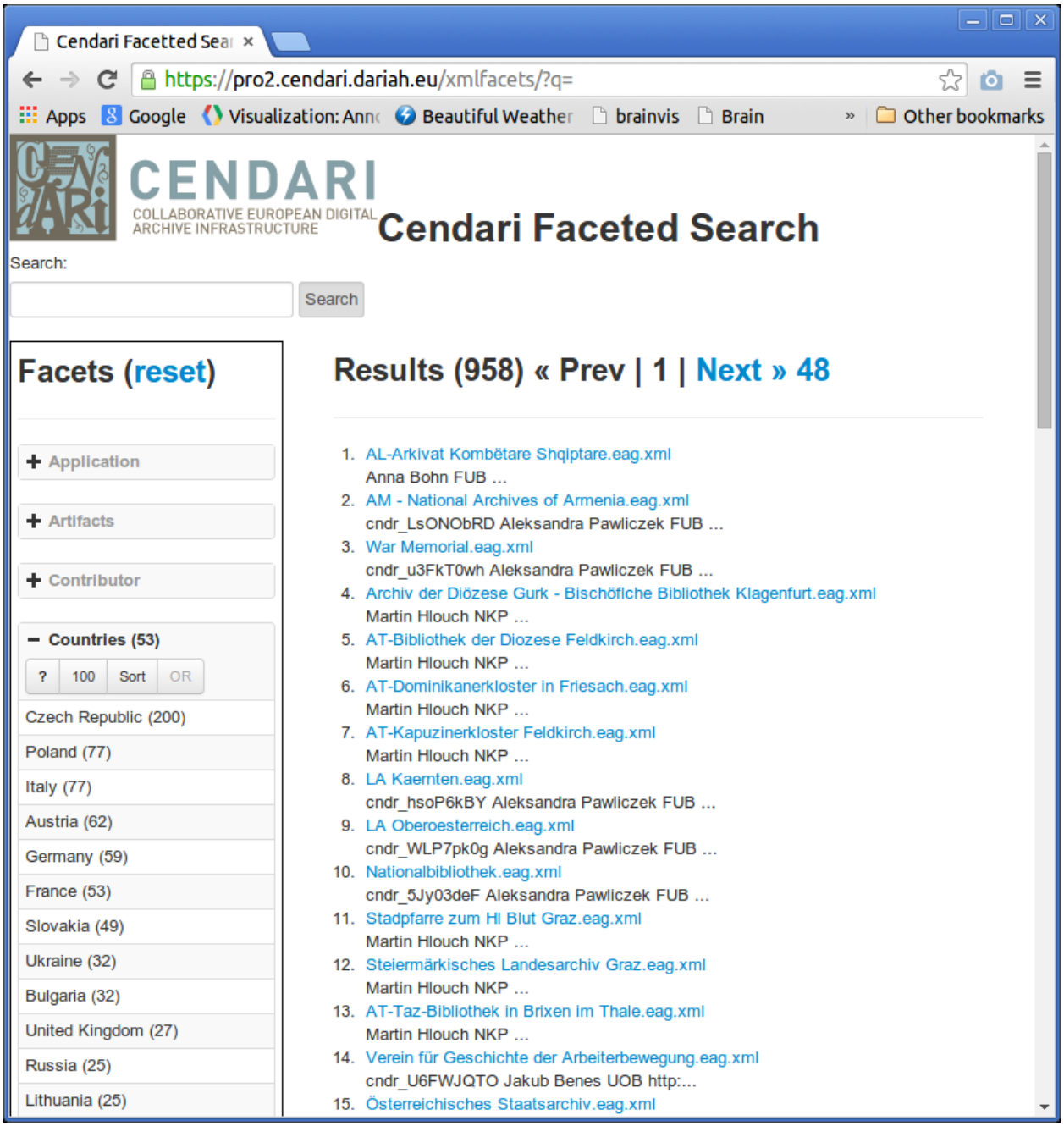

Figure 1: The CENDARI Faceted Search Interface

The CENDARI Faceted Search interface is shown in Figure 1. It allows full-text searching and/or faceting. The figure shows the “Countries” facet that has been opened, revealing the names of the 53 countries that appear in the 958 documents currently indexed. This interface is meant to scale to very large number of documents (millions); therefore, by default, the facet widgets on the left pane only display a limited number of facet values (100) that can be extended by the user in

various ways for exploration and queries. The widget is borrowed from the facetview project[16] and provides many options that we will extend. In particular, we will add small visualizations for specific facets to provide overviews.

Visualizations

Visualizations are effective at showing histograms for values (e.g. document lengths) and dates. They are also useful to show geographical extents of places within the CENDARI repository. However, visualizations should rely on server-side aggregation to avoid sending millions of objects from the web server to the web client; such a huge data transfer would freeze the browser, create long delays, and probably exceed the memory limit of most web browsers.

Therefore, the visualizations will rely on the value, time, and geohash[17] aggregations of ElasticSearch to provide quick responsive pages while providing an accurate overview of the data.

Editing

Some file types and formats can be directly edited in XMLFacets; this feature has been added because the current method to edit XML or text documents available on the CENDARI data store was cumbersome. Instead of extracting the document, editing it with an external program, and updating it with a custom system, XMLFacets allows its edition in place, given the right credentials. To provide this editing feature, XMLFacets relies on the codemirror[18] JavaScript library.

Conclusion

The Searching and Faceted Browsing tool designed for CENDARI is already working on the CENDARI infrastructure. It will be updated using the Agile Development Methodology followed by CENDARI, integrating more data and more visualization components.

Its technology is meant to scale to millions of documents, thanks to the ElasticSearch technology. Currently, we rely on two replicas of ElasticSearch to serve the search index efficiently. We can add more instances if the load increases without changing our tools.

[16] https://github.com/okfn/facetview
[17] http://en.wikipedia.org/wiki/Geohash
[18] http://codemirror.net/

For now, indexing is left to each application that has to follow the conventions described in this document. If needed, we can provide an indexing service that will check for the conformity of data provided to the indexer to avoid introducing errors.

Finally, we expect the search tool to be used as the standard interface for any search; the performed search can be confined to data created by specific applications (e.g. MediaWiki, Note-Taking Environment) if desired. We want to avoid duplicating the design of search interfaces for every tool offered by CENDARI and use XMLFacets as the standard tool, customizable to fit users' and applications' needs. We will evolve XMLFacets to fit these needs.

References

APEnet. (2011-b). *EAD as used within the Archives Portal Europe for finding aids and holdings guides – Description and best practice guide.* http://www.apenet.eu/images/docs/apenet_ead_finding_aids_holdings_guides.pdf.

APEnet. (2011-a). *EAG as used within the Archives Portal Europe for describing archival institutions – Description and best practice guide.* http://www.apenet.eu/images/docs/apenet_eag_archival_institution.pdf.

TEI Consortium, eds. *TEI P5: Guidelines for Electronic Text Encoding and Interchange.* http://www.tei-c.org/P5/, Date of access (28/07/2014).

Appendix A: ElasticSearch Mapping

The equivalent to a database schema for ElasticSearch is the type mapping. It is defined like this:

```
{
    "document" : {
        "properties" : {
            "application" : {
               "type" : "string", "index" :
"not_analyzed"
            },
            "artifact" : {
               "type" : "string", "index" :
"not_analyzed"
            },
            "contributor" : {
               "properties" : {
                   "email" : {
                      "type" : "string",
index" : "not_analyzed"
                   },
                   "name" : {
                      "type" : "string",
"index" : "not_analyzed"
                   }
               }
            },
            "creator" : {
               "properties" : {
                   "email" : {
                      "type" : "string",
"index" : "not_analyzed"
                   },
                   "name" : {
                      "type" : "string",
"index" : "not_analyzed"
                   }
               }
            },
            "date" : {
               "type" : "date", "format" :
"dateOptionalTime"
            },
            "event" : {
               "type" : "string", "index" :
"not_analyzed"
            },
            "format" : {
               "type" : "string", "index" :
"not_analyzed"
            },
            "language" : {
               "type" : "string", "index" :
"not_analyzed"
            },
            "length" : {
               "type" : "long"
            },
            "org" : {
               "type" : "string", "index" :
"not_analyzed"
            },
            "person" : {
               "properties" : {
                   "email" : {
                      "type" : "string",
"index" : "not_analyzed"
                   },
                   "name" : {
                      "type" : "string",
"index" : "not_analyzed"
                   }
               }
            },
            "place" : {
               "properties" : {
                   "location" : {
                      "type" : "geo_point",
"geohash" : true,
                      "fielddata" : {
                          "format" :
"compressed",
                          "precision" :
"3m"
                      }
                   },
                   "name" : {
                      "type" : "string",
"index" : "not_analyzed"
                   }
               }
            },
            "publisher" : {
               "type" : "string", "index" :
"not_analyzed"
            },
            "ref" : {
               "type" : "string", "index" :
"not_analyzed"
            },
            "tag" : {
               "type" : "string", "index" :
"not_analyzed"
            },
            "text" : {
               "type" : "string"
            },
            "title" : {
               "type" : "string"
            },
            "uri" : {
               "type" : "string", "index" :
"not_analyzed"
            }
        }
    }
}
```

Appendix B: ElasticSearch Filter with Permission

To query ElasticSearch, taking into account user and group permissions, a filter like the one below should be used. In the example, the requester's name is "alice" and she belongs to two groups: "inria" and "greencadres". Applications should replace these values with the ones of the user requesting the search.

Filters are cached in ElasticSearch and will be reused throughout the session, leading to very efficient searches.

```
{
    "query" : {
      "match_all" : {}
    },
    "filter" : {
      "or": [
          { "and" : [
            { "missing": { "field" : "groups_allowed" } },
            { "missing": { "field" : "users_allowed" } }
          ] },
          { "and" : [
            { "exists": { "field" : "users_allowed" } },
            { "term": { "users_allowed" : "alice" } }
          ] },
          { "and" : [
            { "exists": { "field" : "groups_allowed" } },
            { "terms": { "groups_allowed" : ["inria",  "greencadres"] } }
          ] }
        ]
    }
}
```