D2.1 – Metadata Retrieval Services Based on Semantic Web Technologies

This deliverable is software.

Co-funded by the European Union

The project is co-funded by the European Union, through the **ICT Policy Support Programme** as part of the **Competitiveness and Innovation Framework Programme (CIP)**.

http://ec.europa.eu/information_society/activities/ict_psp/

Europeana Creative is coordinated by the Austrian National Library

Deliverable

Project Acronym: Europeana Creative

Grant Agreement Number: 325120

Project Title: Europeana Creative

D2.1 Metadata Retrieval Services Based on Semantic Web Technologies

Revision: Final

Authors: Despoina Trivela (NTUA)
Vassilis Tzouvaras (NTUA)
Jana Parvanova (ONTO)
Vladimir Alexiev (ONTO)
Sergiu Gordea (AIT)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	
C	Confidential, only for members of the consortium and the Commission Services	

Österreichische Nationalbibliothek

Europeana Creative is coordinated by the Austrian National Library

Revisions

Version	Status	Author	Date	Changes
0.1	Initial Version	Despoina Trivela, NTUA	May 7, 2014	Table of Contents
0.2	OAI-PHM server documentation	Jana Parvanova, ONTO	June 5, 2014	Integration into the document
0.3	Europeana-client library	Sergiu Gordea, AIT	June 17, 2014	Integration into the document
0.4	Owlim database description	Vladimir Alexiev, ONTO	June 24, 2014	Integration into the document
0.5	Final draft	Despoina Trivela, NTUA	July 9, 2104	Integration Comments
0.6	Final draft	Elisabeth Stricker, ONB Susanne Tremml, ONB	July 31, 2014	Layout, Editing
1.0	Final	Fritz Niemann, ONB	July 31, 2014	Final changes

Distribution

Version	Date of sending	Name	Role in project
0.1	July 17, 2014	Nico Kreinberger, MFG	Reviewer
0.2	July 29, 2014	Saso Zagoranski, SEM	Reviewer
0.5	July 30, 2014	Despoina Trivela, NTUA	
1.0	July 31, 2014	Marcel Watelet, EC	Project Officer

Approval

Version	Date of approval	Name	Role in project
1.0	July 31, 2014	Fritz Niemann, ONB	Project Manager

Table of Contents

Figures

1. Introduction

The purpose of WP2 is to provide services that enable the re-use of Europeana content. These services support the Open Culture Lab environment of WP1 and the development of the five Theme projects within WP4 and WP5. Towards this direction some of the central services provided within WP2 involve the storage and retrieval of the Europeana metadata. The content retrieval services provide closed or open access to content based on the access rules defined by the Content Re-use Framework, CRF established in WP3.

Access to semantically annotated data is enabled by using the semantic repositories which allow for storage retrieval and effective management of data. In general, when data are described by semantic schemata (ontologies) they are stored into semantic repositories that additionally allow for querying and reasoning. In the case of Europeana metadata the OWLIM platform is deployed. OWLIM consists of several semantic repositories that can efficiently handle data expressed in RDF, RDFs or OWL 2 RL and OWL 2 QL languages. It is used to enhance data integration within the Europeana Creative project since it is the most scalable system that additionally allows for SPARQL query evaluation.

Furthermore, effective manipulation of the data stored in the semantic repository requires a mechanism for synchronizing its content with the Europeana EDM repository. The users of the semantic endpoint need to have access to an updated quality and quantity of information. However, the semantic repository is not part of the United Ingestion Manager and therefore it is not updated on a regular basis. Towards this direction an OAI-PMH server has been integrated in the metadata storage system to notify the semantic repository about the updates concerning the Europeana metadata.

The aim of this deliverable is to present the services developed for the metadata storage and retrieval. The rest of this document is organized as follows: in Section 1 we describe the use of OWLIM for storing and accessing semantic data and give some details concerning the quality of data; in Section 2 we present the OAI-PMH server used to synchronize the semantic repository with the EDM data; finally, in Section 3 we give a description of the Europeana Client library that was built to facilitate the access to the content of Europeana repository for Java developers.

2. Europeana Semantic Repository

Europeana SPARQL[1] endpoint provides an alternative way for data search and exploration. It allows developers and advanced users to create elaborate queries for data retrieval – much like they would do with a SQL database. The table below presents some example queries over Europeana data:

# List of datasets from Italy

```
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
SELECT DISTINCT ?dataset
WHERE {
?europeanaAggregation edm:collectionName ?dataset ;
  edm:country "italy" .}
```

#The landing page of objects provided by the European Library, together with the general and controlled right statements associated to it

```
PREFIX edm: <http://www.europeana.eu/schemas/edm/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ens:  <http://www.europeana.eu/schemas/edm/>

SELECT DISTINCT ?CHO ?generalRights ?controlledRights
WHERE {
            ?aggregation edm:provider "The European Library" .
    ?aggregation dc:rights ?generalRights .
    ?aggregation edm:rights ?controlledRights .
    ?aggregation edm:isShownAt ?EuropeanaObjectURL .
    ?aggregation ens:aggregatedCHO ?CHO
}
```

[1] http://www.w3.org/TR/sparql11-query/

2.1 Semantic Repository and SPARQL Endpoint

The Europeana semantic repository and SPARQL endpoint at http://europeana.ontotext.com/ is maintained by Ontotext and the data is stored in OWLIM semantic database. The provided SPARQL end-point offers application developers with valuable alternatives for data querying and accessing and has full SPARQL 1.1 support.

The last data update was done on September 14, 2012. Repository contains ca. 20 million objects in EDM format. This results in:

- 998,471,854 explicit statements
- 265,799,020 entities
- 3,798,446,742 statements (explicit and inferred)

Through the whole time Ontotext has been maintaining the repository and has been helping users to make most out of it (relevant discussions https://basecamp.com/1768384/projects/2156886/messages/24995790#comment_164532323, accessed on July 31, 2014).

2.2 Custom Views

Data exploration is achieved through a graphical user interface which supports various features:

1. Content negotiation based on headers and URLs. Data can be viewed/downloaded in several formats, including RDF/XML, JSON, Turtle etc.
2. Dynamic pagination that allows for effective exploration of very large data sets
3. Full-text search by title or extended content

Moreover, several different views have been developed for cultural objects:

- Europeana: shows the object data as it appears on europeana.eu
- CHO: shows all triples of the CHO, as laid out in the CHO graph
- Custom shortening of URLs, so it's easier to understand the CHO
- Graph view: shows the layout of triples in CHO, provider/europeana aggregation/proxy

The following figure illustrates the graph view:

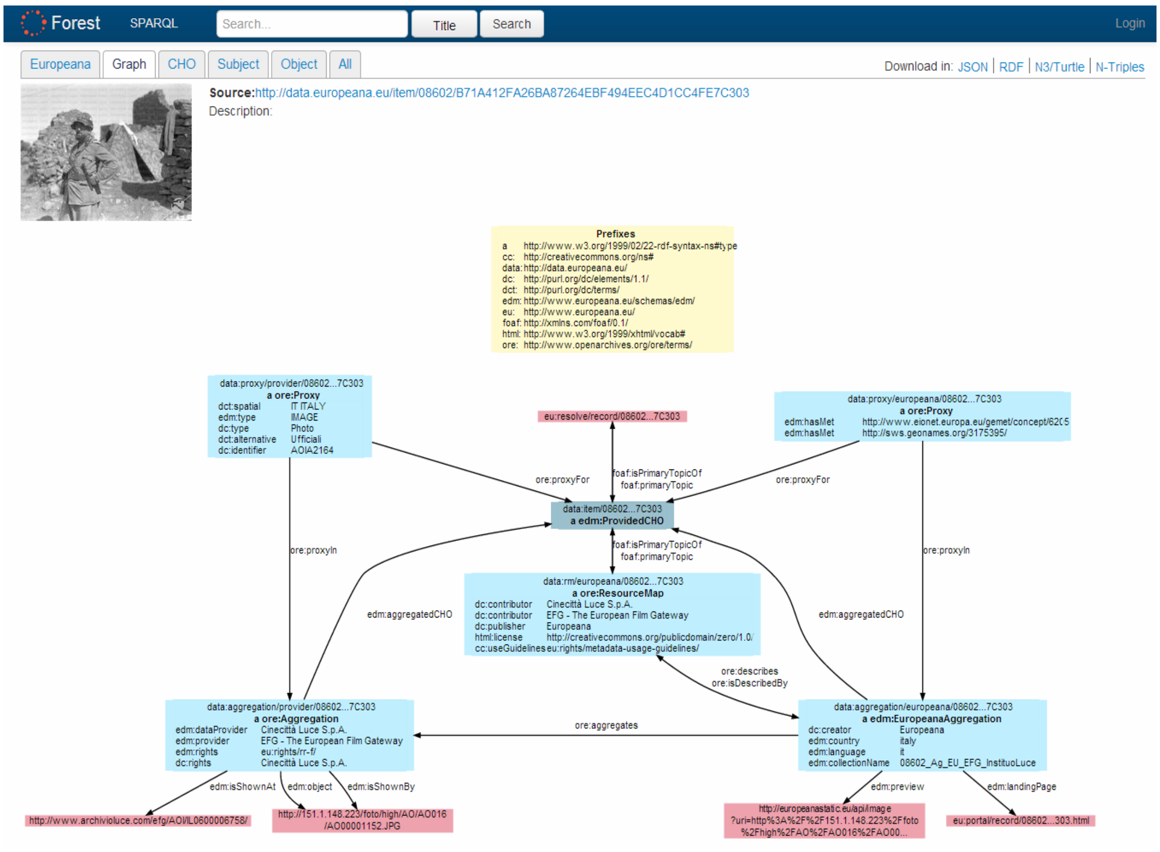

Forest
SPARQL
Search...
Title
Search
Login
Europeana
Graph
CHO
Subject
Object
All
Download in: JSON | RDF | N3/Turtle | N-Triples
Source:http://data.europeana.eu/item/08602/B71A412FA26BA87264EBF494EEC4D1CC4FE7C303
Description:
Prefixes
a http://www.w3.org/1999/02/22-rdf-syntax-ns#type
cc: http://creativecommons.org/ns#
data: http://data.europeana.eu/
dc: http://purl.org/dc/elements/1.1/
dct: http://purl.org/dc/terms/
edm: http://www.europeana.eu/schemas/edm/
eu: http://www.europeana.eu/
foaf: http://xmlns.com/foaf/0.1/
html: http://www.w3.org/1999/xhtml/vocab#
ore: http://www.openarchives.org/ore/terms/
data:proxy/provider/08602...7C303
a ore:Proxy
dct:spatial IT ITALY
edm:type IMAGE
dc:type Photo
dct:alternative Ufficiali
dc:identifier AOIA2164
eu:resolve/record/08602...7C303
data:proxy/europeana/08602...7C303
a ore:Proxy
edm:hasMet http://www.eionet.europa.eu/gemet/concept/62(5
edm:hasMet http://sws.geonames.org/3175395/
ore:proxyFor
foaf:isPrimaryTopicOf
foaf:primaryTopic
ore:proxyFor
data:item/08602...7C303
a edm:ProvidedCHO
ore:proxyIn
foaf:isPrimaryTopicOf
foaf:primaryTopic
ore:proxyIn
data:rm/europeana/08602...7C303
a ore:ResourceMap
dc:contributor Cinecittà Luce S.p.A.
dc:contributor EFG - The European Film Gateway
dc:publisher Europeana
html:license http://creativecommons.org/publicdomain/zero/1.0/
cc:useGuidelines eu:rights/metadata-usage-guidelines/
edm:aggregatedCHO
edm:aggregatedCHO
ore:describes
ore:isDescribedBy
data:aggregation/provider/08602...7C303
a ore:Aggregation
edm:dataProvider Cinecittà Luce S.p.A.
edm:provider EFG - The European Film Gateway
edm:rights eu:rights/rr-f/
dc:rights Cinecittà Luce S.p.A.
ore:aggregates
data:aggregation/europeana/08602...7C303
a edm:EuropeanaAggregation
dc:creator Europeana
edm:country italy
edm:language it
edm:collectionName 08602_Ag_EU_EFG_InstituoLuce
edm:isShownAt
edm:object
edm:isShownBy
edm:preview
edm:landingPage
http://www.archivioluce.com/efg/AOI/IL0600006758/
http://151.1.148.223/foto/high/AO/AO016/AO00001152.JPG
http://europeanastatic.eu/api/image?uri=http%3A%2F%2F151.1.148.223%2Ffoto%2Fhigh%2FAO%2FAO016%2FAO00...
eu:portal/record/08602...303.html

Fig. 1: OWLIM Graph view

2.3 Data Quality

Poor data quality has been recognized as a serious problem for developers trying to use it. As part of the semantic repository effort and the OAI-PMH server development we observed, analysed and reported a number of problems. We believe we have helped Europeana to deliver a higher quality service by fixing them. Some of the reported issues are the following:

- Europeana1321 EDM breaks structure of edm:Place with geo: URN
- Europeana1247 Some language tags from providers don't conform with the IANA list.
- Europeana1248 Missing edm:hasMet
- Europeana1307 Enrichment error for Geographic place
- Europeana1409 Objects with fundamentally faulty identifiers
- Europeana1453 Dereferencing Europeana URIs
- Europeana1454 Compare some old & new EDM records

- Europeana1455 EdmUtils addAsList SEVERE: NoSuchMethodException setIncorporatesList
- Europeana1458 How to ensure URL stability for Hierarchical objects?
- Europeana1475 How to output line breaks in dc:description?
- Europeana1245 Duplicated namespace declaration in RDF

3. OAI-PMH Server

A major issue to take into account while manipulating the data stored in the Europeana semantic repository is the fact that it is almost 2 years old. That is because it has been loaded in 2012 and not updated since. As a result, the semantic database has 20 million records while current European database has more than 30 million. Therefore, it was essential to come up with some mechanism that performs incremental update. Several applications and services can benefit from such mechanism and store Europeana data or parts of it for their own purposes.

3.1 OAI-PMH

'The Open Archives Initiative Protocol for Metadata Harvesting[2] (OAI-PMH) is a low-barrier mechanism for repository interoperability. *Data Providers* are repositories that expose structured metadata via OAI-PMH. *Service Providers* then make OAI-PMH service requests to harvest that metadata. OAI-PMH is a set of six verbs or services that are invoked within HTTP. '

The aim behind setting up OAI-PMH service is to allow for external storages to be synchronized with Europeana data by retrieving incremental updates. A typical scenario of usage would be the following:

1. External system queries OAI-PMH server to get a list of all data sets. For each data set, external system retrieves all changed records since last retrieval. External system records timestamp of data harvesting.
2. Records are listed by chunks. Basically, OAI-PMH service should answer the question "What are the changes on data since data X?"

3.2 Current Data Import Process

Providers serve data via different means - publishing on HTTP/FTP, using OAI-PMH, placing files on file system etc. A REPOX server is used to harvest data that are then passed to a mapping component, the MINT tool, that converts data in ESE or EDM external formats to EDM Internal format. Providers serve data via different means - publishing on HTTP/FTP, using OAI-PMH, placing files on file system etc. A REPOX server is used to harvest data that are then passed to a mapping component, the MINT tool, that converts data in ESE or EDM external formats to EDM Internal format.

[2] http://www.openarchives.org/pmh/

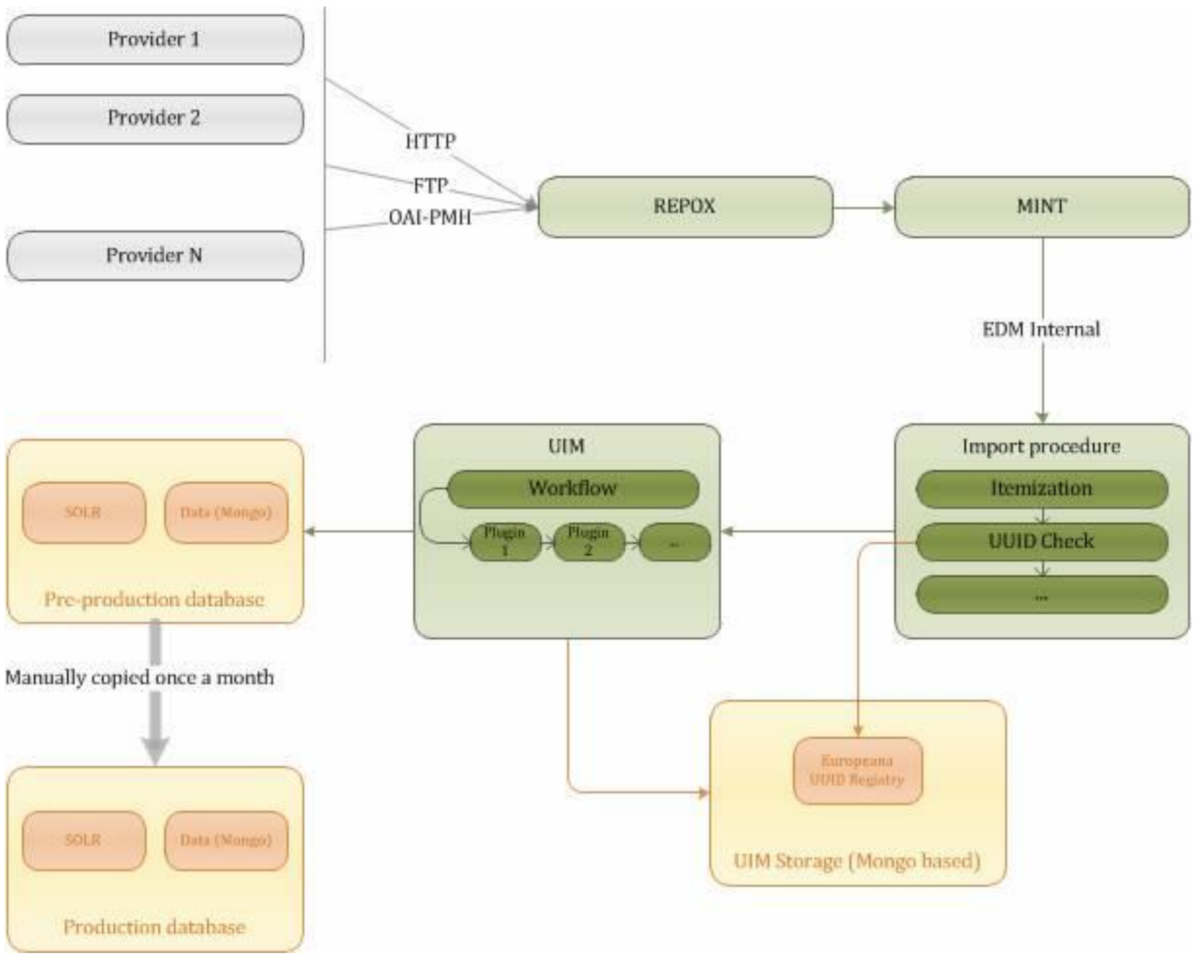


Fig. 2: Data import process

Next, ingestion process starts for the new data. One of the first steps is to check and generate UUIDs. Each record comes with its external identifier assigned by its provider. These identifiers are mapped to internal Europeana UUIDs. Europeana Registry is used in the process. This is a special table where mapping is recorded together with timestamp of latest change for the record. Hash based on record's content is also recorded so that old and new versions could be easily compared. We present an example in the following table.

Original ID	**Europeana UUID**	**Collection Name**	**Timestamp**	**MD5**
A-123	*BFAEF253A74CCEC3AE9D93E9DA5E1E8FC90E9553*	*Museo Galileo - Istituto e Museo di Storia della Scienza*	*16/12/09*	10252f76b41dd06377f1fbac8d3c61f8
RFB0092	0596D11CB28C2990CF348E688A35765D255FBF72	Slovenia	*23/01/10*	396dd1f1a8c42d953ac929bbb99d9b41

Then, for each record an UIM ingestion workflow is invoked. UIM includes a number of workflows - ingestion, publishing etc. Each workflow consists of number of plugins. For example, specialized plugins within the ingestion workflow allow for enrichment.

The UIM uses a Mongo-based storage to persist data and execution states. Results, however, are written to a second storage - which is also Mongo-based. The second storage could be considered a pre-production database. After all records go through ingestion process and results are satisfying, updated data is ready for publishing. Publishing is largely manual operation. SOLR indexes are created on the pre-production data and data and SOLR configuration is copied over existing production setup. This is usually done once a month.

3.3 Initially Adopted Technical Approach

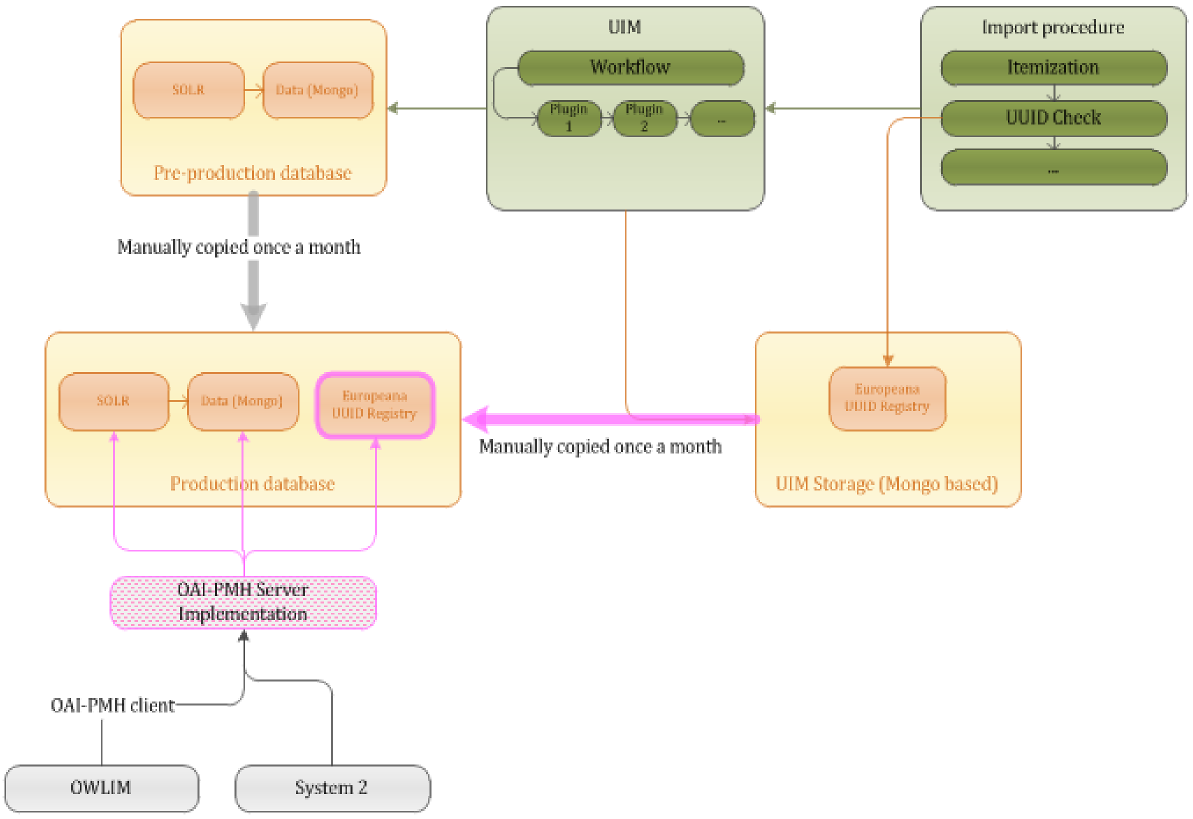


Fig. 3: Initially Adopted Technical Approach

Within our first approach we used an open-source OAI-PMH implementation in Java (oaicat) and provided data for each request by connecting to production database. The Europeana UUID Registry stores essential data - timestamp of record change and IDs of deleted records. It is the primary data source for the OAI-PMH service. Once the UUIDs of the changed records are retrieved, EDM is constructed. As a first step, Europeana API is used.

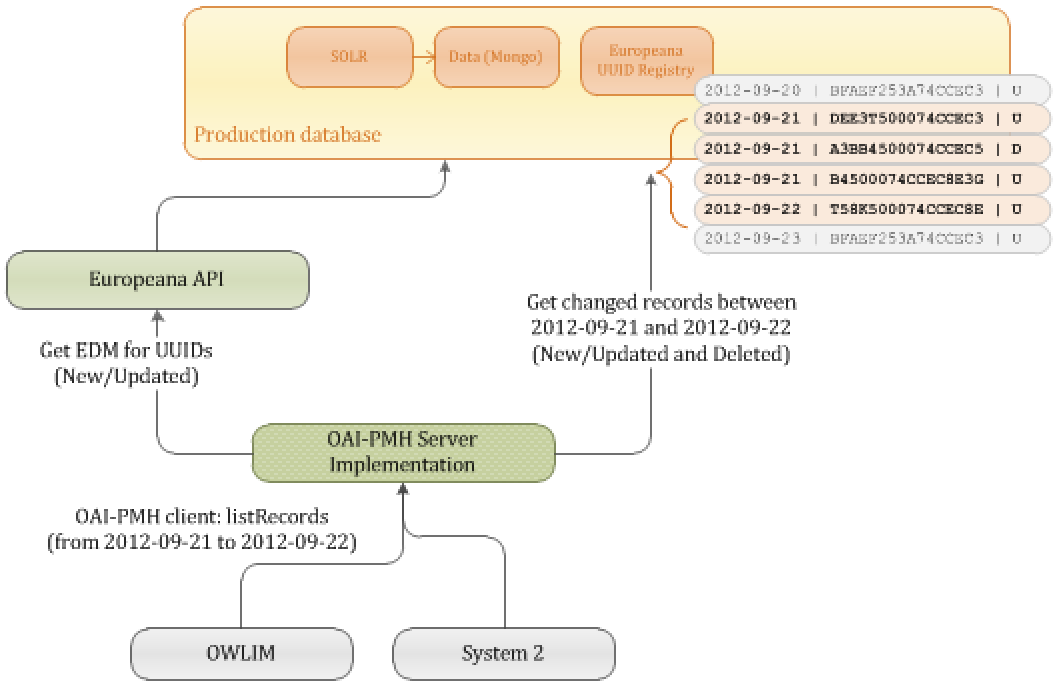

SOLR
Data (Mongo)
Europeana UUID Registry
Production database
2012-09-20 | BFABF253A74CCBC3 | U
2012-09-21 | DEE3T500074CCBC3 | U
2012-09-21 | A3BB4500074CCBC5 | D
2012-09-21 | B4500074CCBC8E3G | U
2012-09-22 | T58K500074CCBC8E | U
2012-09-23 | BFABF253A74CCBC3 | U
Europeana API
Get EDM for UUIDs (New/Updated)
Get changed records between 2012-09-21 and 2012-09-22 (New/Updated and Deleted)
OAI-PMH Server Implementation
OAI-PMH client: listRecords (from 2012-09-21 to 2012-09-22)
OWLIM
System 2

Fig. 4: OAI-PMH implementation

3.4 Deleted Records

Reporting deleted records is needed for proper synchronization with Europeana data.

There are a few things to consider here:

- Only the UUID is to be returned by the OAI-PMH server concerning the deleted records
- There is no trace of deleted records in Europeana production database
- UUIDs in deleted records are present in the Europeana UUID Registry table. Currently though there is no flag that record has been deleted there and timestamp is not updated on deletion

3.5 Architecture Changes

Unfortunately, once we implemented the approach described above (chapter 3.4) we were able to load only 13 million records into OWLIM out of the expected 30 million. In fact, Registry contained only the newest 13 million records as it was introduced with UIM and older records were processed using UIM. Several discussions were organized to deal with this issue and adopt a new approach. As a result, it was decided that the SOLR interface would be used to retrieve changed records. The deleted records tracking is not yet implemented and it is up to Europeana to decide how to maintain and provide information on them. Currently, implementation based on SOLR index of valid records is under way.

3.6 Current Status and Plans

The code of the server and reference client is available at https://github.com/europeana/OAI-PMH. Test installation is available at: http://sandbox12.isti.cnr.it:8080/oaicat/ (sample call: http://sandbox12.isti.cnr.it:8080/oaicat/OAIHandler?verb=GetRecord&identifier=http%3A%2F%2Fdata.europeana.eu%2Fitem2F15411%2Firishfilm_silent_brennan_of_the_moor_php&metadataPrefix=edm).

All records will be loaded in OWLIM shortly after all the changes related to the use SOLR instead of the Registry are completed.

During the process of downloading the records, large number of bugs in data and API were found and submitted to Europeana developers team, thus improving the overall quality of the data and provided services. Finally, handling deleted records remains an open issue.

4. Europeana Client Library

The Europeana portal is used by public users to search and browse the repository which contains more than 30 million objects3 of various types of objects including books, audio and video objects, photos, paintings, etc. These objects are collected from different domains, from fashion to natural history, from archaeology to history, from art and interpretations to science and so on. Given this diversity of objects, various possibilities of search and filtering were implemented both in the Europeana portal and the Europeana API. The overview of the elements that can be used in search queries is presented in Europeana Labs4. The API provides the same search and filtering functionalities as the portal, but the results are presented in computer readable formats (i.e. json, rdf/xml, rss) and not in a human friendly representation (which is html). The json and xml formats are used for standardizing the object serialization, but they are not the native object representations of programming languages (except javascript from which json format is derived). The Europeana-Client library was implemented in order to facilitate access to Europeana Metadata and thumbnails by remote invocation of Europeana Search API5 when using the Java programming language. This was developed by enhancing the old Europeana4J source code which implemented access to 1st version of Europeana API. Europeana Client supports the invocation of the Version 2 of the API and enhances it with additional functionality that supports:

- Easy creation of search queries
- Re-use of portal queries (URL)
- Robust mechanisms for processing large datasets
- Aggregation of user sets
- Downloading object thumbnails
- Accessing the full object representation
- Configuration of API URLs and access keys

For each method in the search and retrieval interfaces, one of more test cases are implementing which serve both for ensuring the correct functionality of the library and for providing handy examples of how to implement various queries.

Within the Europeana Creative project the client library is currently used for:

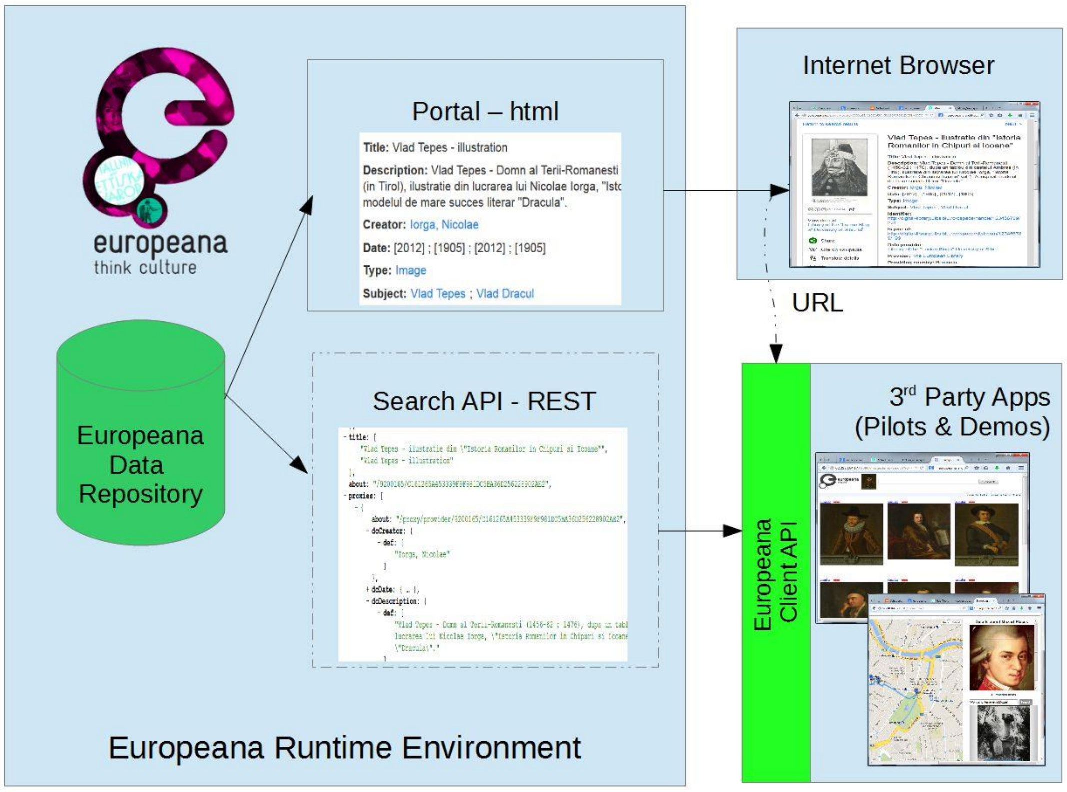
- Building datasets for image similarity service
- Retrieving relevant objects for the geo-mapping demo

[3] Official repository size at the time we edited this document.

[4] See http://labs.europeana.eu/api/data-fields/#facets, accessed on July 31, 2014.

[5] See http://pro.europeana.eu/web/guest/api, accessed on July 31, 2014.

4.1 System Architecture Overview



Title: Vlad Tepes - illustration

Description: Vlad Tepes - Domn al Terii-Romanesti (in Tirol), ilustratie din lucrarea lui Nicolae Iorga, "Isto modelul de mare succes literar "Dracula".

Creator: Iorga, Nicolae

Date: [2012] ; [1905] ; [2012] ; [1905]

Type: Image

Subject: Vlad Tepes ; Vlad Dracul

```
- title: [
    "Vlad Tepes - ilustratie din \"Istoria Romanilor in Chipuri si Icoane\"",
    "Vlad Tepes - illustration"
  ],
  about: "/9200165/C161265A453339F9F981DC5BA36D256228902AE2",
- proxies: [
  - {
        about: "/proxy/provider/9200165/C161265A453339F9F981DC5BA36D256228902AE2",
      - dcCreator: {
        - def: [
              "Iorga, Nicolae"
          ]
        },
      + dcDate: { … },
      - dcDescription: {
        - def: [
              "Vlad Tepes - Domn al Terii-Romanesti (1456-62 ; 1476), dupa un tabl
              lucrarea lui Nicolae Iorga, \"Istoria Romanilor in Chipuri si Icoane
              \"Dracula\"."
          ]
```

Fig. 5: System Architecture Overview

The figure above sketches the positioning of the client library in the Europeana platform and the relationships with the other components. The main blocks of the Europeana Runtime Environment include the Europeana Data Repository, the Portal and the Search API. The public users will typically use the internet browser to retrieve and visualize the Europeana objects. Anyway, the software developers may re-use the Portal Queries for Development purposes in Europeana Client library. The client library also facilitates the access to Europeana objects for 3rd party applications, as the Image Similarity Search and Geomapping Demo, for example. The technical documentation of the library is provided in the following section.

4.2 Technical Documentation

The technical documentation of the Europeana client library is intended to support developers that make use of Europeana Search API in 3rd party Java applications. It includes an overview of the main classes available in the package in form of class and interaction diagrams, source code documentation, system requirements and hyperlinks to the code repository and packaged artefacts (i.e. official releases).

4.2.1 Class Diagram

An overview of the main classes available for querying the Europeana API and retrieving the Europeana Objects is available in the following figure. The facades of the library are implemented by the *EuropeanaConnection* and *EuropeanaApi2Client* which implement the querying of the version 1 and version 2 of the Europeana API, respectively. They make use of the *HttpConnector* class, which facilitates reliable communication over the http protocol. The parameters used for configuring the access to the API are defined in the *europeana-client.properties* file and can be read in a standardized manner through the *ClienConfiguration* class.

The Europeana API provides a large number of query and filtering parameters as one can find in the official documentation6. The client library supports the usage of all these parameters and provides handy method for building queries with the most frequently used ones (see *EuropeanaQuery* and *API2Query* classes). *Api2QueryBuilder* can be used also for instantiating common search queries.

The message returned by the Europeana server response is deserialized to the *EuropeanaApiResults* representation, which may provide appropriate error messages or the Europeana objects (i.e. *EuropeanaApi2Item*) in case of successful service invocation.

[6] See http://labs.europeana.eu/api/data-fields/#facets, accessed on July 31, 2014.

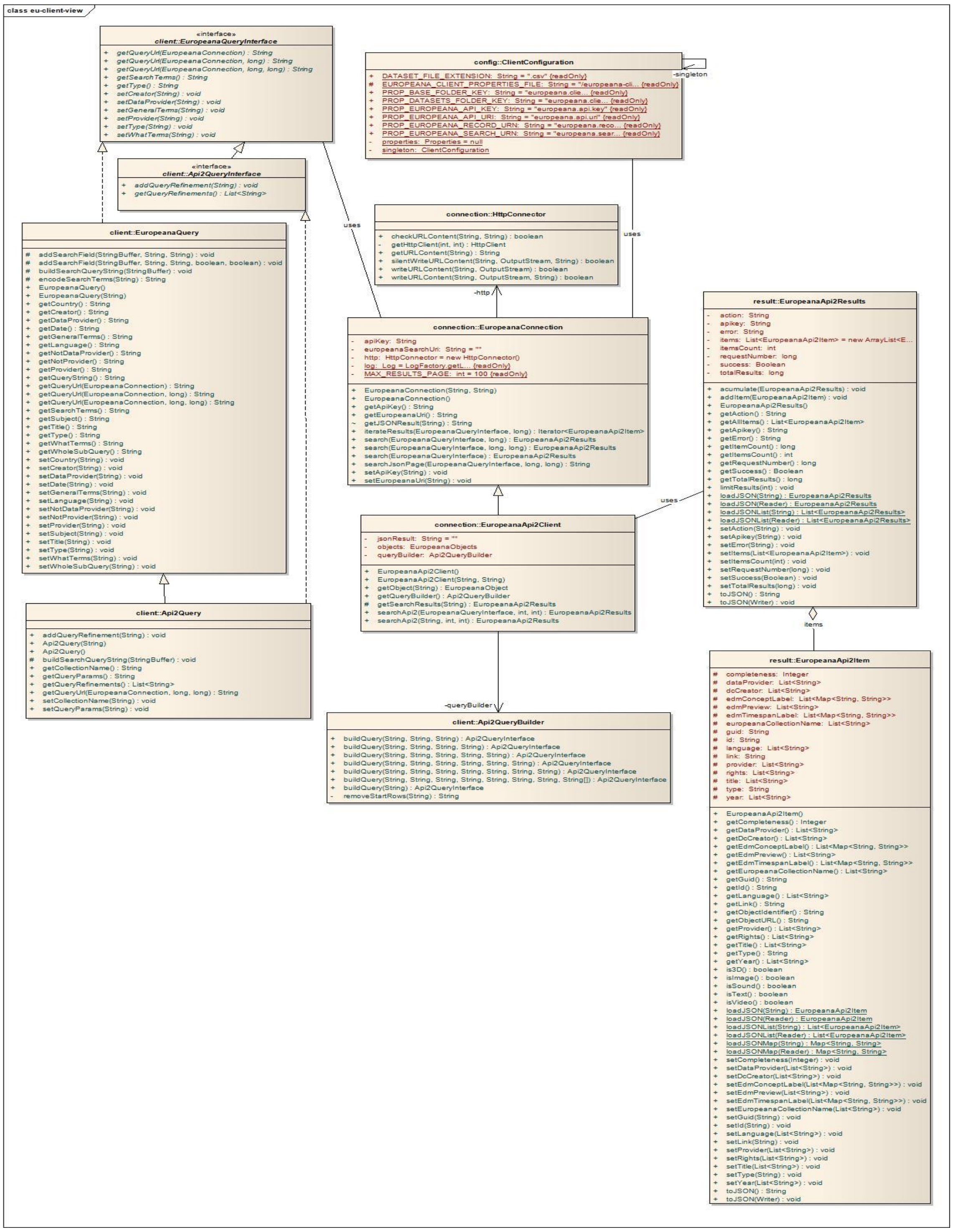

class eu-client-view
«interface»
client::EuropeanaQueryInterface
+ getQueryUrl(EuropeanaConnection) : String
+ getQueryUrl(EuropeanaConnection, long) : String
+ getQueryUrl(EuropeanaConnection, long, long) : String
+ getSearchTerms() : String
+ getType() : String
+ setCreator(String) : void
+ setDataProvider(String) : void
+ setGeneralTerms(String) : void
+ setProvider(String) : void
+ setType(String) : void
+ setWhatTerms(String) : void
«interface»
client::Api2QueryInterface
+ addQueryRefinement(String) : void
+ getQueryRefinements() : List<String>
config::ClientConfiguration
+ DATASET_FILE_EXTENSION: String = ".csv" {readOnly}
# EUROPEANA_CLIENT_PROPERTIES_FILE: String = "/europeana-cli... {readOnly}
+ PROP_BASE_FOLDER_KEY: String = "europeana.clie... {readOnly}
+ PROP_DATASETS_FOLDER_KEY: String = "europeana.clie... {readOnly}
+ PROP_EUROPEANA_API_KEY: String = "europeana.api.key" {readOnly}
+ PROP_EUROPEANA_API_URI: String = "europeana.api.uri" {readOnly}
+ PROP_EUROPEANA_RECORD_URN: String = "europeana.reco... {readOnly}
+ PROP_EUROPEANA_SEARCH_URN: String = "europeana.sear... {readOnly}
- properties: Properties = null
- singleton: ClientConfiguration
-singleton
uses
uses
connection::HttpConnector
+ checkURLContent(String, String) : boolean
- getHttpClient(int, int) : HttpClient
+ getURLContent(String) : String
+ silentWriteURLContent(String, OutputStream, String) : boolean
+ writeURLContent(String, OutputStream) : boolean
+ writeURLContent(String, OutputStream, String) : boolean
-http
client::EuropeanaQuery
# addSearchField(StringBuffer, String, String) : void
# addSearchField(StringBuffer, String, String, boolean, boolean) : void
# buildSearchQueryString(StringBuffer) : void
# encodeSearchTerms(String) : String
+ EuropeanaQuery()
+ EuropeanaQuery(String)
+ getCountry() : String
+ getCreator() : String
+ getDataProvider() : String
+ getDate() : String
+ getGeneralTerms() : String
+ getLanguage() : String
+ getNotDataProvider() : String
+ getNotProvider() : String
+ getProvider() : String
+ getQueryString() : String
+ getQueryUrl(EuropeanaConnection) : String
+ getQueryUrl(EuropeanaConnection, long) : String
+ getQueryUrl(EuropeanaConnection, long, long) : String
+ getSearchTerms() : String
+ getSubject() : String
+ getTitle() : String
+ getType() : String
+ getWhatTerms() : String
+ getWholeSubQuery() : String
+ setCountry(String) : void
+ setCreator(String) : void
+ setDataProvider(String) : void
+ setDate(String) : void
+ setGeneralTerms(String) : void
+ setLanguage(String) : void
+ setNotDataProvider(String) : void
+ setNotProvider(String) : void
+ setProvider(String) : void
+ setSubject(String) : void
+ setTitle(String) : void
+ setType(String) : void
+ setWhatTerms(String) : void
+ setWholeSubQuery(String) : void
connection::EuropeanaConnection
- apiKey: String
- europeanaSearchUri: String = ""
- http: HttpConnector = new HttpConnector()
- log: Log = LogFactory.getL... {readOnly}
- MAX_RESULTS_PAGE: int = 100 {readOnly}
+ EuropeanaConnection(String, String)
+ EuropeanaConnection()
+ getApiKey() : String
+ getEuropeanaUri() : String
~ getJSONResult(String) : String
+ iterateResults(EuropeanaQueryInterface, long) : Iterator<EuropeanaApi2Item>
+ search(EuropeanaQueryInterface, long) : EuropeanaApi2Results
+ search(EuropeanaQueryInterface, long, long) : EuropeanaApi2Results
+ search(EuropeanaQueryInterface) : EuropeanaApi2Results
+ searchJsonPage(EuropeanaQueryInterface, long, long) : String
+ setApiKey(String) : void
+ setEuropeanaUri(String) : void
result::EuropeanaApi2Results
- action: String
- apikey: String
- error: String
- items: List<EuropeanaApi2Item> = new ArrayList<E...
- itemsCount: int
- requestNumber: long
- success: Boolean
- totalResults: long
+ acumulate(EuropeanaApi2Results) : void
+ addItem(EuropeanaApi2Item) : void
+ EuropeanaApi2Results()
+ getAction() : String
+ getAllItems() : List<EuropeanaApi2Item>
+ getApikey() : String
+ getError() : String
+ getItemCount() : long
+ getItemsCount() : int
+ getRequestNumber() : long
+ getSuccess() : Boolean
+ getTotalResults() : long
+ limitResults(int) : void
+ loadJSON(String) : EuropeanaApi2Results
+ loadJSON(Reader) : EuropeanaApi2Results
+ loadJSONList(String) : List<EuropeanaApi2Results>
+ loadJSONList(Reader) : List<EuropeanaApi2Results>
+ setAction(String) : void
+ setApikey(String) : void
+ setError(String) : void
+ setItems(List<EuropeanaApi2Item>) : void
+ setItemsCount(int) : void
+ setRequestNumber(long) : void
+ setSuccess(Boolean) : void
+ setTotalResults(long) : void
+ toJSON() : String
+ toJSON(Writer) : void
uses
connection::EuropeanaApi2Client
- jsonResult: String = ""
- objects: EuropeanaObjects
- queryBuilder: Api2QueryBuilder
+ EuropeanaApi2Client()
+ EuropeanaApi2Client(String, String)
+ getObject(String) : EuropeanaObject
+ getQueryBuilder() : Api2QueryBuilder
# getSearchResults(String) : EuropeanaApi2Results
+ searchApi2(EuropeanaQueryInterface, int, int) : EuropeanaApi2Results
+ searchApi2(String, int, int) : EuropeanaApi2Results
client::Api2Query
+ addQueryRefinement(String) : void
+ Api2Query(String)
+ Api2Query()
# buildSearchQueryString(StringBuffer) : void
+ getCollectionName() : String
+ getQueryParams() : String
+ getQueryRefinements() : List<String>
+ getQueryUrl(EuropeanaConnection, long, long) : String
+ setCollectionName(String) : void
+ setQueryParams(String) : void
items
-queryBuilder
client::Api2QueryBuilder
+ buildQuery(String, String, String) : Api2QueryInterface
+ buildQuery(String, String, String, String) : Api2QueryInterface
+ buildQuery(String, String, String, String, String) : Api2QueryInterface
+ buildQuery(String, String, String, String, String, String) : Api2QueryInterface
+ buildQuery(String, String, String, String, String, String, String) : Api2QueryInterface
+ buildQuery(String, String, String, String, String, String, String, String[]) : Api2QueryInterface
+ buildQuery(String) : Api2QueryInterface
- removeStartRows(String) : String
result::EuropeanaApi2Item
# completeness: Integer
# dataProvider: List<String>
# dcCreator: List<String>
# edmConceptLabel: List<Map<String, String>>
# edmPreview: List<String>
# edmTimespanLabel: List<Map<String, String>>
# europeanaCollectionName: List<String>
# guid: String
# id: String
# language: List<String>
# link: String
# provider: List<String>
# rights: List<String>
# title: List<String>
# type: String
# year: List<String>
+ EuropeanaApi2Item()
+ getCompleteness() : Integer
+ getDataProvider() : List<String>
+ getDcCreator() : List<String>
+ getEdmConceptLabel() : List<Map<String, String>>
+ getEdmPreview() : List<String>
+ getEdmTimespanLabel() : List<Map<String, String>>
+ getEuropeanaCollectionName() : List<String>
+ getGuid() : String
+ getId() : String
+ getLanguage() : List<String>
+ getLink() : String
+ getObjectIdentifier() : String
+ getObjectURL() : String
+ getProvider() : List<String>
+ getRights() : List<String>
+ getTitle() : List<String>
+ getType() : String
+ getYear() : List<String>
+ is3D() : boolean
+ isImage() : boolean
+ isSound() : boolean
+ isText() : boolean
+ isVideo() : boolean
+ loadJSON(String) : EuropeanaApi2Item
+ loadJSON(Reader) : EuropeanaApi2Item
+ loadJSONList(String) : List<EuropeanaApi2Item>
+ loadJSONList(Reader) : List<EuropeanaApi2Item>
+ loadJSONMap(String) : Map<String, String>
+ loadJSONMap(Reader) : Map<String, String>
+ setCompleteness(Integer) : void
+ setDataProvider(List<String>) : void
+ setDcCreator(List<String>) : void
+ setEdmConceptLabel(List<Map<String, String>>) : void
+ setEdmPreview(List<String>) : void
+ setEdmTimespanLabel(List<Map<String, String>>) : void
+ setEuropeanaCollectionName(List<String>) : void
+ setGuid(String) : void
+ setId(String) : void
+ setLanguage(List<String>) : void
+ setLink(String) : void
+ setProvider(List<String>) : void
+ setRights(List<String>) : void
+ setTitle(List<String>) : void
+ setType(String) : void
+ setYear(List<String>) : void
+ toJSON() : String
+ toJSON(Writer) : void

Fig. 6: Class Diagram: An overview of Main Classes

4.2.2 Sequence Diagrams

All methods in the Façade classes are tested by using the junit toolset, and the implemented unit tests represent good examples for using the Europeana client library. The following interaction diagrams represent the steps performed to implement the most useful scenarios: the simple search and the processing of large result sets.

The *SimpleSearchTest* example presents the sequence of operations which are necessary to be performed for querying the API and printing the retrieved objects in the java console. In the first step the Api2Query object is instantiated and the search criteria are set. Secondly, the Façade object (*EuropeanaApi2Client*) is instantiated by using the default configurations and the appropriate search method is invoked. This will fire the remote invocation of the search API and will make use of the Gson library to parse the Json result into the Java object. Finally, the search results are printed in the java console in the case of successful text execution.

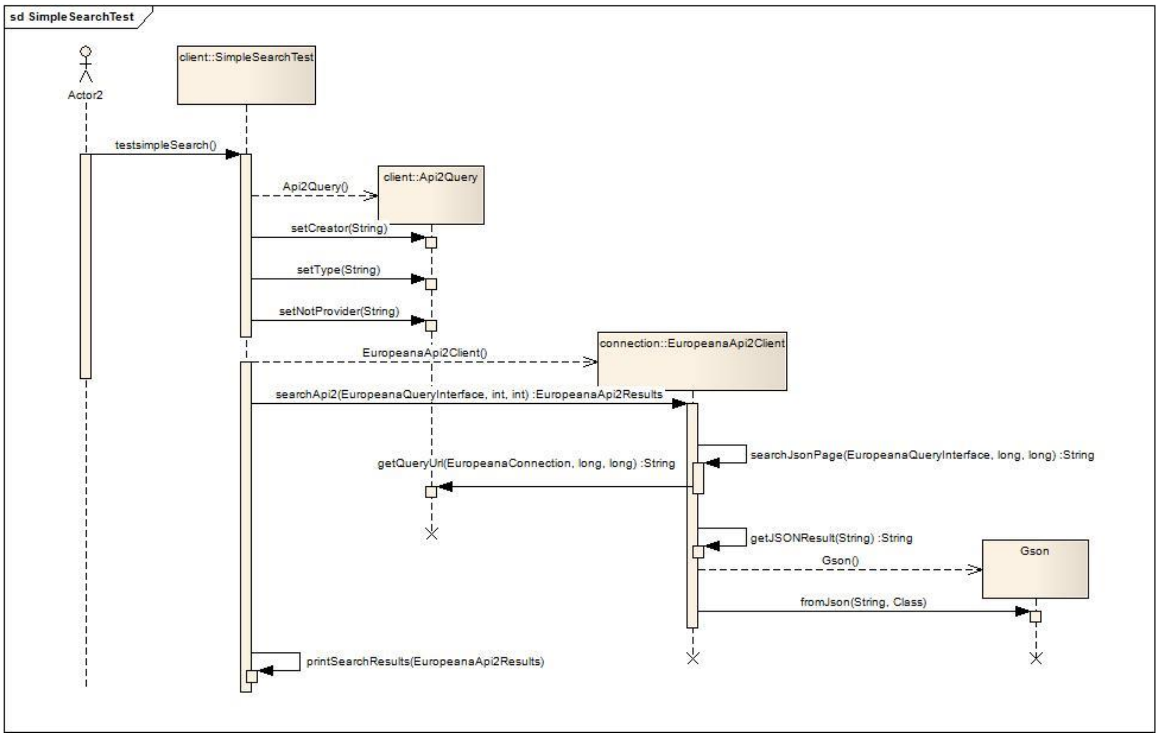

sd SimpleSearchTest
Actor2
client::SimpleSearchTest
testsimpleSearch()
client::Api2Query
Api2Query()
setCreator(String)
setType(String)
setNotProvider(String)
connection::EuropeanaApi2Client
EuropeanaApi2Client()
searchApi2(EuropeanaQueryInterface, int, int) :EuropeanaApi2Results
searchJsonPage(EuropeanaQueryInterface, long, long) :String
getQueryUrl(EuropeanaConnection, long, long) :String
getJSONResult(String) :String
Gson
Gson()
fromJson(String, Class)
printSearchResults(EuropeanaApi2Results)

Fig. 7: Sequence diagram: SimpleSearchTest

The *DownloadThumbnailsTest* presents an example which uses the client library for accessing large scale datasets through the Europeana API and processing (parts of) this information. In this concrete test case, the ThumbnailURLs are extracted from the search results and the associated media files are downloaded from the internet. The implementation makes use of the Observer pattern which is implemented by using the *LargeThumbnailsetProcessing* class. The Europeana Object Ids are provided as input from a text file, which was previously generated by using the search API. The LargeThumbnailsetProcessing class implements the Observable interface and notifies the Observer (ThumbnailDownloader) as soon as a new bunch of Thumbnail URLs are available for download. The *update()* method is responsible for implementing this functionality. The implementation follows the best effort paradigm and it doesn't stop when the thumbnail downloading fails from one or other reason. Instead, it warns about the error and keeps track of the number of objects that couldn't be processed successfully.

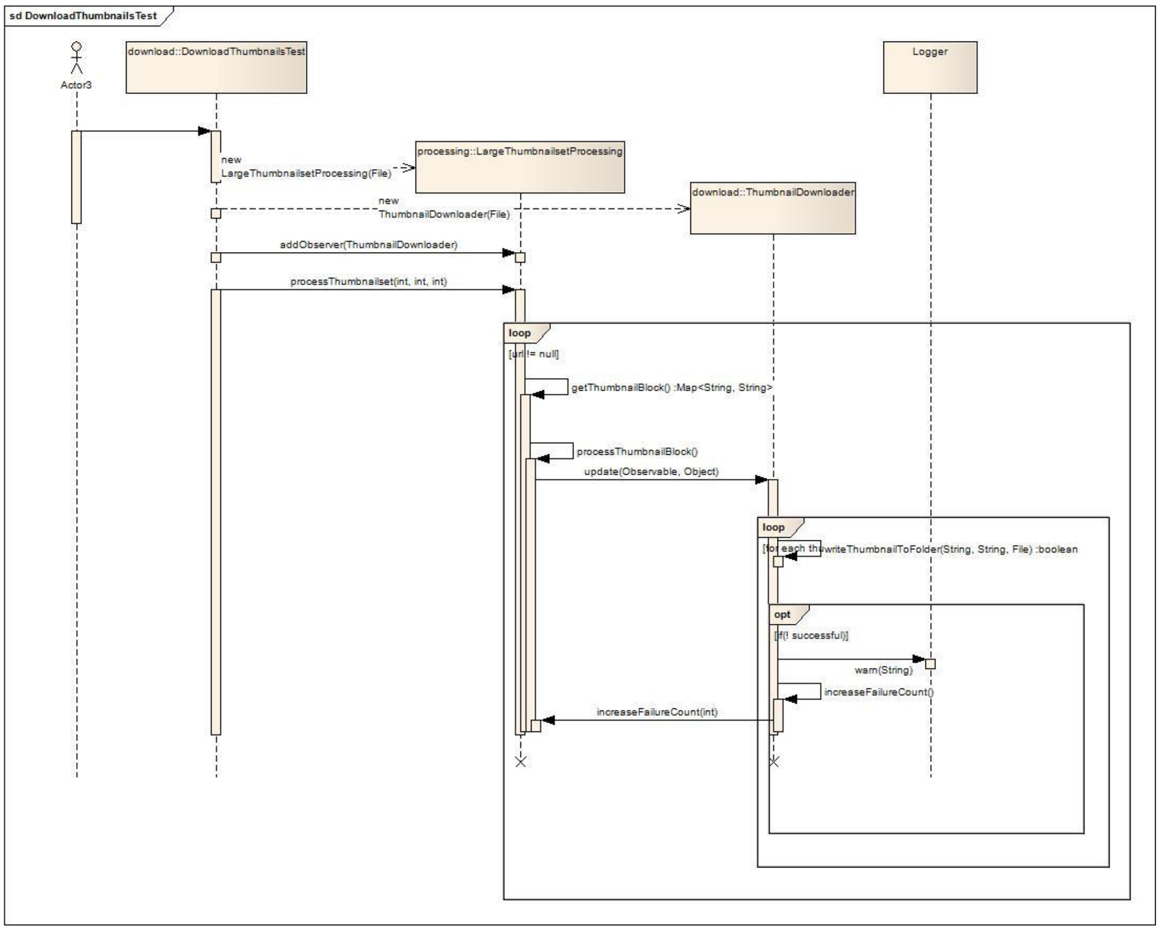

sd DownloadThumbnailsTest
Actor3
download::DownloadThumbnailsTest
processing::LargeThumbnailsetProcessing
download::ThumbnailDownloader
Logger
new LargeThumbnailsetProcessing(File)
new ThumbnailDownloader(File)
addObserver(ThumbnailDownloader)
processThumbnailset(int, int, int)
loop
[url != null]
getThumbnailBlock() :Map<String, String>
processThumbnailBlock()
update(Observable, Object)
loop
[for each thu
writeThumbnailToFolder(String, String, File) :boolean
opt
[if(! successful)]
warn(String)
increaseFailureCount()
increaseFailureCount(int)

Fig. 8: Sequence Diagram: DownloadThumbnailsTest

4.2.3 Javadocs

The following tables present the documentation of the Façade class EuropeanaApi2Client, the QueryInterface and the classes used for accessing the Europeana Thumbnails.

Class: EuropeanaApi2Client	
The main class used for accessing the Europeana Search API - V2	
Method	Api2QueryBuilder getQueryBuilder()
	Method for creation of an Api2QueryBuilder `@return Api2QueryBuilder: object for generation of a new query.`
Method	searchApi2(EuropeanaQueryInterface query, int limit, int start)

	`Method for remote invocation of Europeana Search API, Version 2`
	
	`This method takes a query interface object and the limit and start of a query to generate a URL and use it to get search results from the search API by calling the {@link #getSearchResults(String)} method.`
	
	`@param query @see Api2Query: a query object representing the expected results.`
	`@param limit: the limit for the amount of results retrieved.`
	`@param start: the starting index for the query results.`
	`@return The return value is a EuropeanaApi2Results object containing the search results.`
	`@throws IOException`
	`@throws EuropeanaApiProblem`

Method	EuropeanaApi2Results getSearchResults(String url)

	Method for search results from Europeana Search API, Version 2
	
	The method retrieves a json result from the provided URL and generates a EuropeanaApi2Results object based on the json data.
	
	`@param url: URL for the search request.`
	`@return generated object from json data.`
	`@throws IOException`
	`@throws EuropeanaApiProblem`

Method	EuropeanaApi2Results searchApi2(String portalSearchUrl, int limit, int start)

	Method for remote invocation of Europeana Search API, Version 2
	
	This method takes a portal search URL string and the limit and start of a query to generate a query interface object and URL and use them to get search results from the search API by calling the {@link #getSearchResults(String)} method.
	
	`@param portalSearchUrl: search URL for the portal search.`
	`@param limit: the limit for the amount of results retrieved.`
	`@param start: the starting index for the query results.`
	`@return The return value is a EuropeanaApi2Results object containing the search results.`
	`@throws IOException`

	`@throws EuropeanaApiProblem`
Method	EuropeanaObject getObject(String id)

	Method for retrieval of a EuropeanaObject
	
	The method uses the client configuration to generate a URL and retrieve json results. The json results are used to generate a full EuropeanaObject which is then return by the method.
	
	`@param id: id of the new object.`
	`@return EuropeanaObject which represents a full result object of the search.`
	`@throws IOException`

Class: ThumbnailAccessor	
A ThumbnailsAccesor is a tool that makes easier the handling of thumbnails of the Europeana items.	
Method	List<String> copyThumbnails(EuropeanaQueryInterface search, File dir, int maxResults)

	The copyThumbnails method extracts a thumbnail image from the search results and stores it in the provided directory.
	
	`@param search: EuropeanaQueryInterface object representing the search.`
	`@param dir: A directory where the thumbnail will be saved.`
	`@param maxResults: The maximal amount of result retrieved by the search.`

	`@return a list of ids from items which provided valid thumbnails.`
	`@throws IOException`
	`@throws EuropeanaApiProblem`

Method	boolean writeThumbnailToFolder(String id, String thumbnailUrl, File imageFolder)

	The method writeThumbnailToFolder creates an output stream and writes the binary content of the thumbnail to a new file in a specified folder.
	
	`@param id: ID string of the image file.`
	`@param thumbnailUrl: URL of the thumbnail.`
	`@param imageFolder: Folder where the resulting images are to be saved.`
	`@return boolean indicating whether the operation has been successful.`

Method	FileOutputStream createOutputStream(File imageFolder, String id)

	Helper method for the creation of a file output stream.
	
	`@param imageFolder: folder where images are to be saved.`
	`@param id: id of specific image file.`
	`@return FileOutputStream object prepared to store images.`
	`@throws FileNotFoundException`

Method	File getImageFile(File dir, String id)

	Helper method to get an image file in a certain directory.
	
	`@param dir: directory where to search the image file.`
	`@param id: id of the image file.`
	`@return File object representing the found image.`

Method	List<String> copyThumbnails(Map<String, String> thumbnailsMap, File imageFolder)

	The copyThumbnails method iterates the provided thumbnails map and filters the items that are skipped due to failed write operations in the specified folder.
	
	`@param thumbnailsMap: map of thumbnails to be searched.`
	`@param imageFolder: folder where to save found thumbnails.`
	`@return the list of items that were not saved on disk.`
	`@throws IOException`
	`@throws FileNotFoundException`

Class: ThumbnailsForCollectionAccessor	
A ThumbnailsAccesor is a tool that makes easier the handling of thumbnails of the Europeana items.	
Method	Map<String, String> getThumbnailsForCollection(int start, int limit, int errorHandlingPolicy)
	This method extracts the map of <thumbnailId,

	thumbnailURL> by invoking the search API. If available the URL of the LARGE version of thumbnails is returned, otherwise the first available If <code>limit > DEFAULT_BLOCKSIZE</code>, the thumbnails will be be fetched iteratively in DEFAULT_BLOCKSIZE chunks
	
	`@param start - first position in results. if smaller than 0, this parameter will default to 1`
	`@param limit - the number or returned results. If <code>start + limit > totalResults</code>, the (last) available result starting with the start position will be returned`
	`@return`
	`@throws EuropeanaApiProblem`

Method	void fetchBlock(int blockStartPosition, int blockLimit, int errorHandlingPolicy)

	Helper method to fetch a block in a thumbnail.
	
	`@param blockStartPosition: starting position for block fetching.`
	`@param blockLimit: limit of blocks to be fetched.`
	`@param errorHandlingPolicy: policy for error handling.`

Method	void fetchNextBlock(int start, int limit)

	Helper method to fetch the next block in a thumbnail.
	
	`@param start: starting position for block fetching.`
	`@param limit: limit of blocks to be fetched.`

Method	String getLargestThumbnail(EuropeanaApi2Item item)
	Helper method to retrieve the largest thumbnail in a Europeana

	item.
	
	`@param item: Europeana item where the thumbnail is searched.`
	`@return string containing a uri of the largest thumbnail.`

Method	int getBlockSize()
	Method to get the block size. `@return int representing the block size.`
Method	void setBlockSize(int blockSize)
	Method to set the block size. `@param blockSize int representing the block size.`
Method	void setQuery(Api2QueryInterface query)
	Method to set the query. `@param query: Api2QueryInterface object representing the query to be set.`
Method	Api2QueryInterface getQuery()
	Method to retrieve the query. `@return Api2QueryInterface object representing the retrieved query.`

Class: Api2QueryInterface	
Interface definition for standardized Search API calls.	
Method	void addQueryRefinement(String qf)
	Registers a query refinement Similar to the "add keyword" functionality in the portal `@param qf: string to define the query refinement`
Method	List<String> getQueryRefinements()
	retrieves the list of registered query refinements `@return list of registered query refinements`

Class: EuropeanaQueryInterface	
Standard interface for Europeana queries.	
Method	String getSearchTerms()
	Returns the Europeana search terms query string `@return the searchTerms argument of the URL`
Method	String getQueryUrl(EuropeanaConnection connection)
	Returns the full Europeana query URL

	`@param connection: defines the used Europeana connection object.` `@return string representing the query URL of a search.` `@throws UnsupportedEncodingException`
Method	String getQueryUrl(EuropeanaConnection connection, long offset)
	Returns the full Europeana query URL `@param connection: defines the used Europeana connection object.` `@param offset: The item in the search results to start with; first item is 1 [default = 1].` `@return string representing the query URL of a search.` `@throws UnsupportedEncodingException`
Method	String getQueryUrl(EuropeanaConnection connection, long limit, long offset)
	Returns the full Europeana query URL `@param connection: defines the used Europeana connection object.` `@param limit: The number of records to return; the maximum value is 100 [default = 12].` `@param offset: The item in the search results to start with; first item is 1 [default = 1].` `@return string representing the query URL of a search.` `@throws UnsupportedEncodingException`

Method	String getType()
	Retrieves the used query type. `@return used query type.`
Method	void setType(String type)
	Sets the used query type. `@param type: used query type.`
Method	void setWhatTerms(String what)
	Sets the terms for a query. `@param what: terms for the query.`
Method	void setGeneralTerms(String generalTerms)
	Sets general query terms. `@param generalTerms: string containing general terms.`
Method	void setCreator(String creator)

	Sets the creator of the query. `@param creator: string containing query creator.`
Method	void setProvider(String provider)
	Sets the provider of a query. `@param provider: string containing the provider.`
Method	void setDataProvider(String dataProvider)
	Sets the data provider of the query. `@param dataProvider: string containing the data provider.`

Software Location and Technical Requirements

Link to software	https://github.com/europeana/europeana-client/
Releases	https://github.com/europeana/europeana-client/releases
Log-in information	Open Source – Publicly Available
Development environment	Java
Programming	Java 6

language used	
Application Type	Java Library
Operating system requirements	Any OS able to run JDK1.6 or upper versions
Port requirements / default ports used	Remote invocation of API V2 using the port 80.
Interface	Java
Licensing conditions	Open Source - EUPL

5. Conclusion

This deliverable is a technical documentation concerning the services that have been implemented in the Europeana-Creative project in order to store and retrieve metadata. In particular, we presented the OWLIM platform that has been deployed for data storage and retrieval, the OAI-PHM server that has been integrated in the storage system to enable synchronisation with the Europeana content and finally the Europeana Client library that has been integrated to enhance the access to Europeana repository for developers.