

DELIVERABLE

Project Acronym: EAGLE
Grant Agreement number: 325122
Project Title: Europeana network of Ancient Greek and Latin Epigraphy

First Release of AIM Infrastructure

D4.2.1

version: 1.0

Revision: Final

Authors:

Giuseppe Amato (CNR-ISTI)
Paolo Bolettieri (CNR-ISTI)
Fabrizio Falchi (CNR-ISTI)
Paolo Manghi (CNR-ISTI)
Andrea Mannocci (CNR-ISTI)
Franco Zoppi (CNR-ISTI)

Contributors:

Vittore Casarosa (CNR-ISTI)

Reviewers:

Nicola Alfarano (GOGATE)
Miguel Angel Sicilia (UAH)

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

Revision History

Revision	Date	Author	Organisation	Description
0.1	31/01/2014	Franco Zoppi	CNR-ISTI	Structure of the document and draft content
1.0 Draft	07/03/2014	Giuseppe Amato, Paolo Bolettieri, Fabrizio Falchi, Paolo Manghi, Andrea Mannocci, Franco Zoppi	CNR-ISTI	Full content and final internal revision
1.0 Final	25/03/2014	Miguel-Angel Sicilia, Paolo Bolettieri, Andrea Mannocci, Franco Zoppi	CNR-ISTI	Curation Tools and Image Processing details added. Integration of reviewers' comments

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

TABLE OF CONTENTS

EXECUTIVE SUMMARY	4
1 THE AIM INFRASTRUCTURE IMPLEMENTATION IN RELEASE 1	5
2 METADATA AGGREGATION SYSTEM	7
2.1 EAGLE AGGREGATION WORKFLOW	7
2.2 EAGLE AGGREGATION KERNEL	7
2.3 STORE SERVICES	8
2.4 METADATA CLEANING AND CURATION SERVICES	8
2.5 INDEX SERVICE	9
2.6 EXPORT SERVICES	9
3 IMAGE RETRIEVAL SYSTEM	10
3.1 IMAGE FEATURE EXTRACTOR COMPONENT (IFE)	10
3.2 IMAGE INDEXER	10
3.3 IMAGE RECOGNIZER	10
3.4 IMAGE SEARCH ENGINE	11
3.5 IMAGE RECOGNITION API	11
3.6 SIMILARITY SEARCH API	11
4 APPENDIX A - HW AND SW REQUIREMENTS	12
5 APPENDIX B – USING THE CONTENT CHECKER	13
6 APPENDIX C – IMAGE RECOGNITION AND SIMILARITY SEARCH SERVICE AND API	17
6.1 IMAGE RECOGNITION	17
6.2 SIMILARITY SEARCH	17
6.3 EXAMPLES	18
6.3.1 <i>Java Examples</i>	18
6.3.2 <i>Testing Images</i>	19
6.3.3 <i>JSON Responses</i>	19
6.3.3.1 <i>Image Recognize JSON Example</i>	19
6.3.3.2 <i>Image Similarity Search JSON Example</i>	19

EXECUTIVE SUMMARY

This document describes the current implementation (Release 1) of the EAGLE Aggregation and Image Retrieval system (AIM) Infrastructure in terms of:

- Current implementation against the specification given in "D4.1 AIM Infrastructure Specification" (Section 1).
- Details about the Metadata Aggregation System (Section 2).
- Details about the Image Retrieval System (Section 3).
- HW & SW requirements of the AIM (Appendix A).
- Sample of the Content Checker Curation Tool (Appendix B).
- Image Recognition and Similarity Search API (Appendix C).

This document being just a Release Note produced as accompanying document of the AIM infrastructure software (D4.2.1, deliverable of type "Product"), please refer to the released document "D4.1 AIM Infrastructure Specification" for details about the full featured AIM Infrastructure.

1 THE AIM INFRASTRUCTURE IMPLEMENTATION IN RELEASE 1

This section summarizes the components implemented by the Aggregation and Image Management infrastructure within the scope of the full system as defined in "D4.1 AIM Infrastructure Specification".

Figure 1-1 shows the overall architecture of the EAGLE system as defined in D4.1 and its implementation status in Release 1. Different colors (and patterns) show respectively:

- **Green (no pattern):** components implemented in Release 1.
- **Yellow (dotted):** components that will be implemented in Release 2.
- **Grey (horizontal lines):** components whose implementation is to be confirmed yet.
- **Blue (vertical lines):** components whose responsibility is outside the scope of the AIM Infrastructure.
- **White (dotted grid):** component whose responsibility is outside the scope of the EAGLE project.

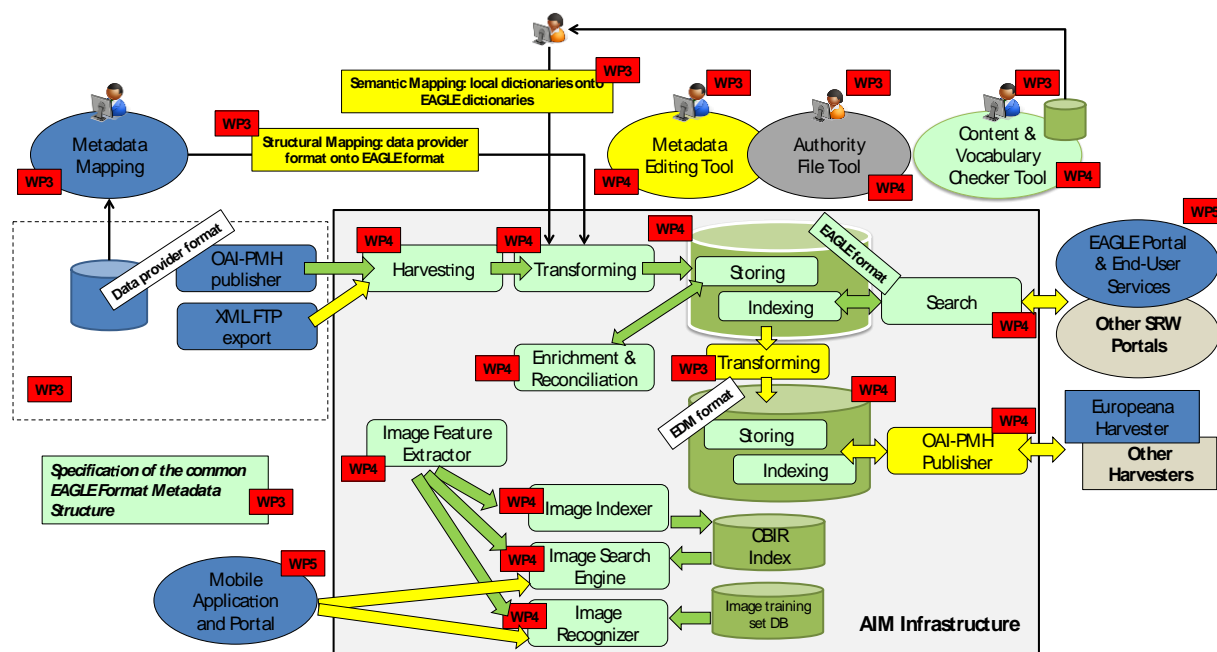


Figure 1-1 EAGLE Architecture – Global Picture – Status at Release 1

Figure 1-2 shows the global functional overview, the interaction and the flow of information between the different functional blocks and components of the AIM, i.e. the Metadata Aggregation System (MAS) and the Image Retrieval System (IRS). Here again, different colors show the different implementation status of the components.

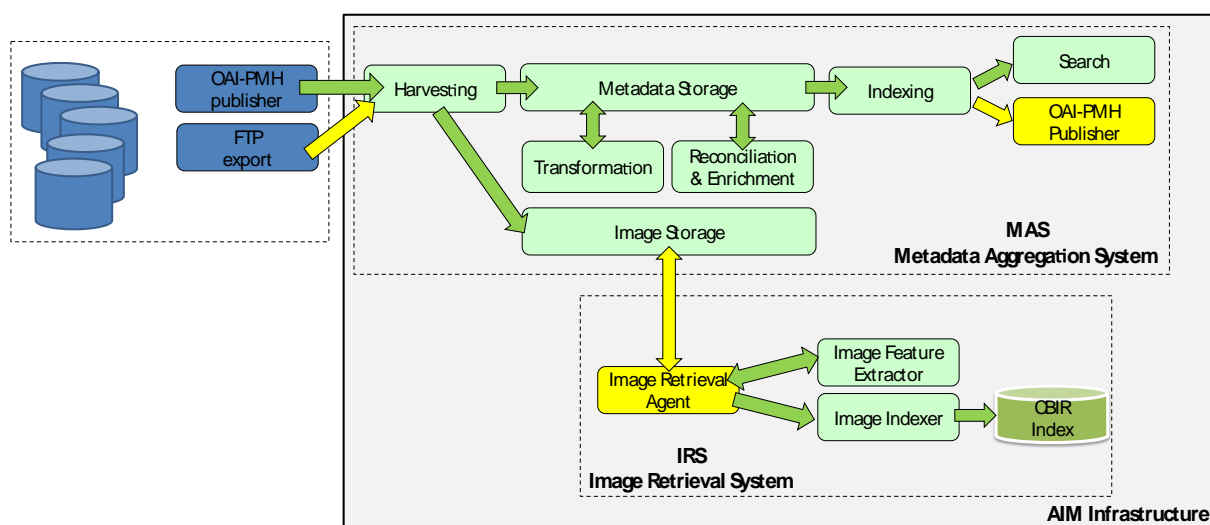


Figure 1-2 Functional AIM Infrastructure – Global Picture – Status at Release 1

The MAS will periodically harvest metadata records (Harvesting Service) made available from CPs and store them within the data infrastructure (Metadata Storage Service). Then, the MAS will start building the EAGLE Information Space step by step. Native metadata records enter a processing flow where they are structurally transformed to comply with a common metadata schema, the EAGLE Metadata Format - EMF (Transformation Service). At the same time, URLs to images pointed by metadata descriptions are visited and the relative images are persisted (Image Storage Service) in the AIM. Once metadata are transformed, metadata records are cleaned and can be further improved by curators via selective editing (Cleaning Service). Eventually they will get into their final revision and will be ready to be indexed (Indexing Service) and disseminated either to external public through the web portal and the mobile application (Search Service), or to third party software agents via OAI-PMH (OAI-PMH Publisher Service).

The IRS will periodically call the Image Storage Service via the Image Retrieval Agent to get new items to be indexed. For each new item collected, the IRS extracts its visual descriptors and encodes the visual descriptors in an efficient textual format (Image Feature Extractor Service), then puts the encoded descriptors into the CBIR¹ Index (Image Indexer Service).

¹ Content-Based Image Retrieval.

2 METADATA AGGREGATION SYSTEM

The EAGLE Metadata Aggregation System relies on the D-NET software toolkit, developed within the DRIVER project efforts and extended within the OpenAIRE, OpenAIREplus², EFG, EFG1914³ and HOPE⁴ projects. Within the EAGLE framework, D-NET is being further extended and customized in order to meet new EAGLE-specific requirements and peculiarities.

This Section describes the customizations performed on D-NET in order to satisfy the EAGLE requirements.

2.1 EAGLE AGGREGATION WORKFLOW

The overall workflow of the EAGLE infrastructure is depicted in Figure 2-1, where all the blocks are selected D-NET components.



Figure 2-1 EAGLE Aggregation Workflow

The XML metadata records provided by the Content Providers are gathered (by harvesting, file transfer or other ad-hoc means) and stored into the Native Metadata Store. After that, metadata in native format are transformed into the EAGLE Metadata Format, making them structurally uniform, and stored in a Transformed Metadata Store. At this point the workflow forks into two: one branch deals with pure metadata enhancement, curation and indexing. The other branch is instead in charge of fetching images pointed to by collected metadata and storing them in an Object Store for later usage by the Image Retrieval System (see Section 3).

It is crucial to provide a staged metadata storage so that processing and elaborations late in the workflow can take place without repeating all the steps from the beginning. Thus, each Metadata Store can be considered as a “safe-point” in data processing.

2.2 EAGLE AGGREGATION KERNEL

Metadata Harvester Service

The Metadata Harvester Service has been deployed and tested against collection from OAI-PMH, remote FTP (hosted by a CP in our case) and local file system (at the aggregation system). Another extension is currently in progress in order to collect metadata from a EAGLE-dedicated shared workspace in which CPs can upload their content. This solution was designed in order to meet the needs of some CPs which are unable to make their data available for remote collection in a programmatic way.

²<http://www.openaire.eu/>

³<http://www.europeanfilmgateway.eu/it>

⁴<http://www.peoplesheritage.eu/>

Metadata Transformation Service

The Metadata Transformation Service is operational. The actual transformation step is based on an XSLT mapping which maps the properties of metadata records in input into one or more EAGLE entity in output, being them conforming to EAGLE metadata format as described in Deliverable 3.1 and its eventual amendments. A test phase has started in order to refine the mapping from eventual error.

Image Harvester Service

The actual implementation of the Image Harvester Service i) scans all the Visual Representation entities (ref. D3.1) present in the Information Space, ii) follows the URL pointing to the external file with picture extension (e.g. .jpg), and iii) collects the digital object into the Image Store Service (see below) for convenience of the IR subsystem.

2.3 STORE SERVICES

Metadata Stores

All the Metadata Stores and relative services are configured and integrated in the workflow. MongoDB is the back-end on which the metadata storage service relies.

Image Store

The Image Store Service has been implemented and deployed. It implements a REST interface for CRUD operations and tagging of collected items (pictures in this case). To date, the full integration between the MAS and IRS (thank to this component) is still to be completed and will be soon tested and integrated. MongoDB-GridFS is the back-end on which the image storage service relies.

2.4 METADATA CLEANING AND CURATION SERVICES

Cleaner Service

The Cleaner Service component has been deployed and integrated within the workflow. However, in its actual implementation no cleaning policy is in place. As a matter of facts, CPs made clear that they want to take care of the process of alignment of vocabularies right at the sources. However, the component is in place in order to anticipate just in case any future need.

Curation tools

According to the DoW, the tools promised were three: i) Metadata Editor, ii) Content Checker and iii) Vocabulary Checker.

As emerged during debates and plenary meetings, Content Providers currently chose not to edit their metadata in the aggregated information space using the Metadata Editor, as they always prefer to hold the most authoritative version of the information flowing in the infrastructure. Any modification or update of records thus will happen right at the CP repository; any change occurred is reflected in the aggregation system at the next scheduled harvesting process.

The Content Checker is deployed, running and available at <http://node0.d.eagle.research-infrastructures.eu:8080/is/mvc/lightui/index.do>. It offers a low-level portal with basic Search& Browse features through which a CP might check its own data post-transformation and explore the XMLs of the entities generated from its initial XML records.

The Vocabulary Editor and Checker are deployed. Nevertheless their actual use is still an open point as the CPs chose not to use the vocabulary tool natively offered by the D-NET toolkit. In fact, a Tematres

endpoint, the controlled vocabulary system chosen within EAGLE, can be found at <http://www.eagle-network.eu/voc/material/> (for the vocabulary about materials, one example out of the seven vocabularies defined).

Sample screenshots of the Content Checker are shown in "Appendix B – Using the Content Checker".

2.5 INDEX SERVICE

The Index Service is fully operational in Release 1. The SOLR index holds XML documents being complaint to Eagle Aggregation Metadata Schema as described in Deliverable 3.1.

2.6 EXPORT SERVICES

Search Service

The SOLR index is publicly reachable thanks to Search Service at http://node0.d.eagle.research-infrastructures.eu:8080/is/mvc/index/EMF-index-cleaned/solr.do/select?q=* and replies to queries compliant with SOLR syntax.

Its counterpart accepting CQL queries can be found at http://node0.d.eagle.research-infrastructures.eu:8080/is/mvc/index/EMF-index-cleaned/cql.do/select?q=*.

The aforementioned Content Checker is based on results returned by the Search Service.

OAI-publisher

The OAI-publisher service has been deployed but the configuration must still be completed. These records are used in order to troubleshoot the ingestion process to Europeana and assess the quantity and quality of the records.

3 IMAGE RETRIEVAL SYSTEM

The Image Retrieval System provides two modes of recognition of an image provided in input as a query image. In the first mode, called Similarity Search, the query result will be a list of images contained in the data base, ranked in order of similarity to the query image. In the second mode, called Recognition Mode, the result of the query will be the information associated with the recognized image, or a message indicating that the image was not recognized. In this second mode it is possible for an epigraph to appear in the query image in any position, and also as part of a more general picture (e.g. the picture of an archeological site, or the scanned image of the page of a book).

In the following paragraphs we describe the component developed in the first release of the AIM Infrastructure.

3.1 IMAGE FEATURE EXTRACTOR COMPONENT (IFE)

This is the component in charge of analyze the visual content of an image (to be added to the database or being provided as a query), to generate its visual descriptors. The IFE has a multi-threaded architecture for fast extraction of features and to take advantage of multicore processors. It has a plug-in architecture, so that it is easy to add or delete the mathematical libraries supporting the extraction algorithms of many different features. In the first release we leveraged SIFT features, for their effectiveness in the image recognition context.

3.2 IMAGE INDEXER

The Image indexer module leverages the functionality of the Melampo CBIR System⁵.

The SIFT⁶ features extracted by IFE are transformed into strings of text suitable to be indexed and searched by a standard full-text search engine. The search engine used in the current implementation is Apache Lucene.

We used the so called *Bag of Features* approach to transforms the visual features to text. With this technique a textual vocabulary of visual words (the textual tags) is created starting from all the local descriptors of the whole dataset.

In EAGLE, we defined a specific vocabulary of visual words, based on the characteristics of the epigraph collections of *The Epigraphic Database Rome (EDR)*.

Once the vocabulary has been created, each epigraph has been described by a set of “words” in the vocabulary and indexed into Apache Lucene.

In this first release, ~17.000 images of the archive of *The Epigraphic Database Rome* were indexed.

3.3 IMAGE RECOGNIZER

The image recognizer, given a query image, uses the image training database to decide if the epigraph contained in the query image belongs to one of the stored training sets. In this release it allows to recognize images of epigraphs contained in the *EDR* archive.

⁵ <https://github.com/claudiogennaro/Melampo>

⁶ Scale-Invariant Feature Transform.

3.4 IMAGE SEARCH ENGINE

The Similarity Search components, given in input a query image, returns a list of images “similar” to the query image, ranked in decreasing order of similarity. In this release the image search engine is fully operational, but it will be exploited by the end-users only when the Portal and/or the Mobile Application will be available.

3.5 IMAGE RECOGNITION API

The Image Recognition Service provides a REST API to query the index. It provides methods that accept in input an image and return a JSON response containing either the identifier of the recognized epigraph or a “NOT IDENTIFIED” message.

This API will be used by the EAGLE user interface whenever it receives from a user (either through a browser or through the mobile application) an image to be recognized.

See Appendix C for details.

3.6 SIMILARITY SEARCH API

The Similarity Search API provides a REST API that accepts in input either an image or the ID of an image already in the CBIR index and returns a JSON response containing a list of the most similar images to the query ranked in decreasing order of similarity.

In the first case, the user will search similar images by uploading an image or by providing an URL to an image (this is known as Query by Example); in the second case the user will search similar images by providing the ID of an epigraph already in the index; this feature can be useful to find epigraphs similar to the one already obtained as the result of a metadata search. In either case the result will be a list of image IDs, ordered with respect to the decreasing value of the similarity.

See Appendix C for details.

4 APPENDIX A - HW AND SW REQUIREMENTS

The hardware minimal requirements of a virtual machine to satisfy to run Release 1 of the production EAGLE Infrastructure are:

- Open Source XEN (no Citrix XEN Server) and lvm volumes (no raw files)
- CPUs: 4 (currently the production virtual machine has AMD Opteron(tm) Processor 6176, MHz 2300 CPUs)
- RAM: 8GB
- HD: 200GB (but it must be possible to extend it based on the size and number of input records)

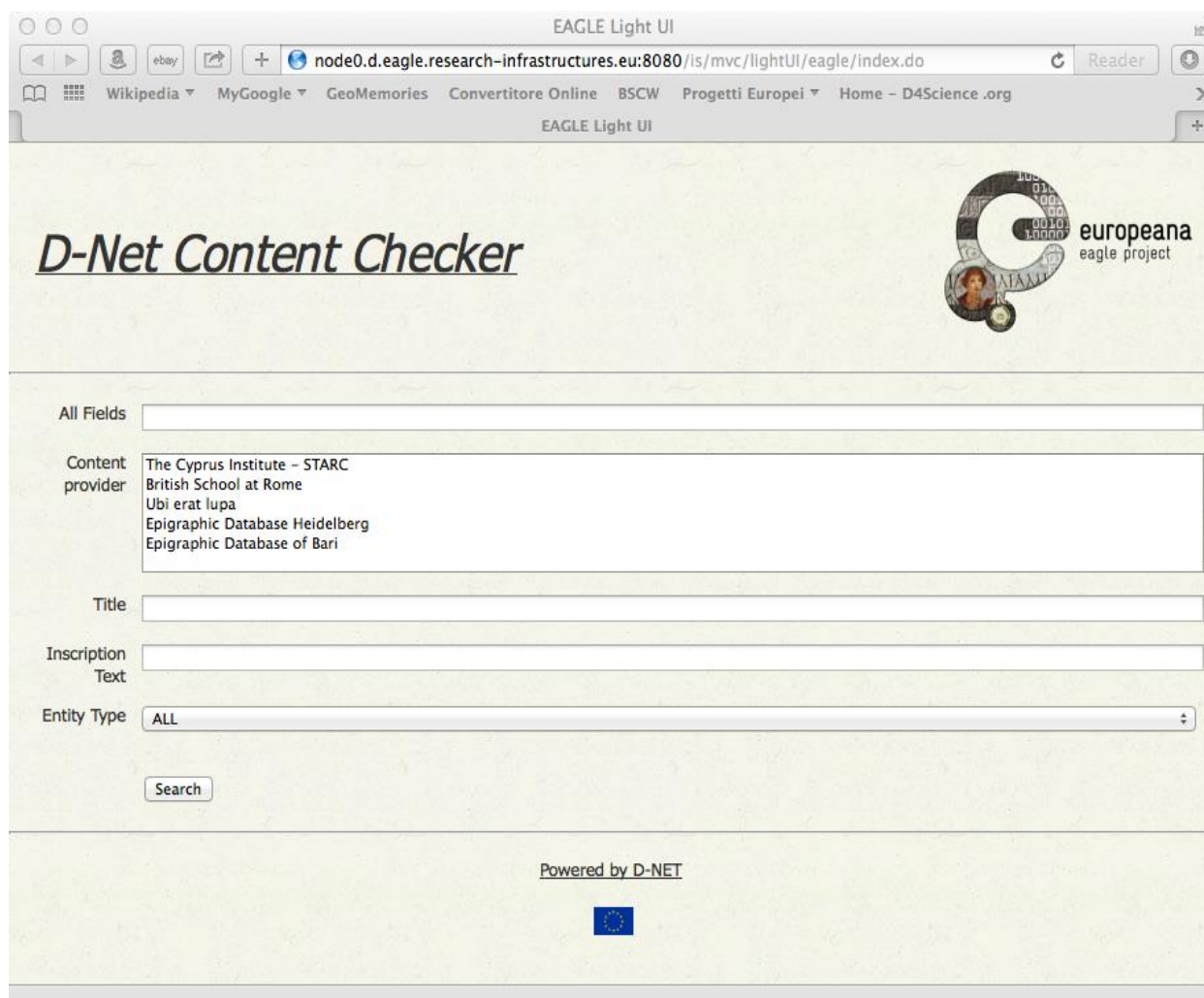
The software requirements are instead:

- OS: 64-bit Ubuntu 12.04 LTS
- Standard Packages and Libraries:
 - MongoDB
 - Apache2
 - Apache Tomcat 6
 - Apache Lucene 3.6
 - Jetty 6.1.11 (currently upgrading to Tomcat 7)
 - Java 6 (currently upgrading to 7)
 - OpenIMAJ (<http://www.openimaj.org/>)
- Specific Libraries:
 - Melampo (<https://github.com/claudiogennaro/Melampo>)
 - VIR (<https://github.com/ffalchi/it.cnr.isti.vir>)

5 APPENDIX B – USING THE CONTENT CHECKER

This appendix shows some screenshots captured in a sample session of the Content Checker, i.e. the curation tool that will be used to check the both the quantitative results of the aggregation process and the quality of the aggregated metadata.

First, the tool shows the initial page allowing to express a query on the content of the aggregated information space (Figure 5-1 Content Checker - Starting page); a query on this returns a set of records which satisfy the expressed condition (Figure 5-2 Content Checker – Retrieving a set of results). This first result is then refined to restrict the number of records of interest (Figure 5-3 Content Checker – Refining a query) and finally a single record is accessed to check its quality (Figure 5-4 Content Checker – Accessing a record).



The screenshot shows a web browser window titled "EAGLE Light UI" with the URL "node0.d.eagle.research-infrastructures.eu:8080/is/mvc/lightUI/eagle/index.do". The browser's address bar and tabs are visible, showing various search engines and services. The main content area of the browser displays the "D-Net Content Checker" interface. The page has a light beige background and features the Europeana eagle project logo in the top right corner. The title "D-Net Content Checker" is prominently displayed in a large, italicized font. Below the title, there is a search form with several input fields: "All Fields" (empty), "Content provider" (containing a list of providers: "The Cyprus Institute - STARC", "British School at Rome", "Ubi erat lupa", "Epigraphic Database Heidelberg", and "Epigraphic Database of Bari"), "Title" (empty), "Inscription Text" (empty), and "Entity Type" (set to "ALL"). A "Search" button is located below the "Entity Type" field. At the bottom of the page, there is a footer that reads "Powered by D-NET" and includes the European Union flag logo.


Figure 5-1 Content Checker - Starting page

EAGLE Light UI

node0.d.eagle.research-infrastructures.eu:8080/is/mvc/lightUI/eagle/results.do?_reposit

Wikipedia MyGoogle GeoMemories Convertitore Online BSCW Progetti Europei Home - D4Science .org

D-Net Content Checker



Query: *


Documents found (5450)

Pages: [1] 2 3 4 5 6 7 8 9 10 >> (next)


inscriptiontype	
Grabinschrift	531
Dedication	213
Weihschrift	170
dedication	81
Funerary	78
Honours	71
more	

material	
marble	314
Kalkstein	227
grey limestone	215
Marmor	203
Granit	85
brown limestone	75
more	


entitytype	
Artifact	2020
Documental manifestation	2020
Visual representation	1410

 **Entity type:** Artifact
Title: HD000144
Description:


[\[View the Record\]](#)

 **Entity type:** Documental manifestation
Title: HD000144
Description:

[\[View the Record\]](#)

 **Entity type:** Artifact
Title: HD000288
Description:

[\[View the Record\]](#)

 **Entity type:** Documental manifestation
Title: HD000288
Description:

[\[View the Record\]](#)


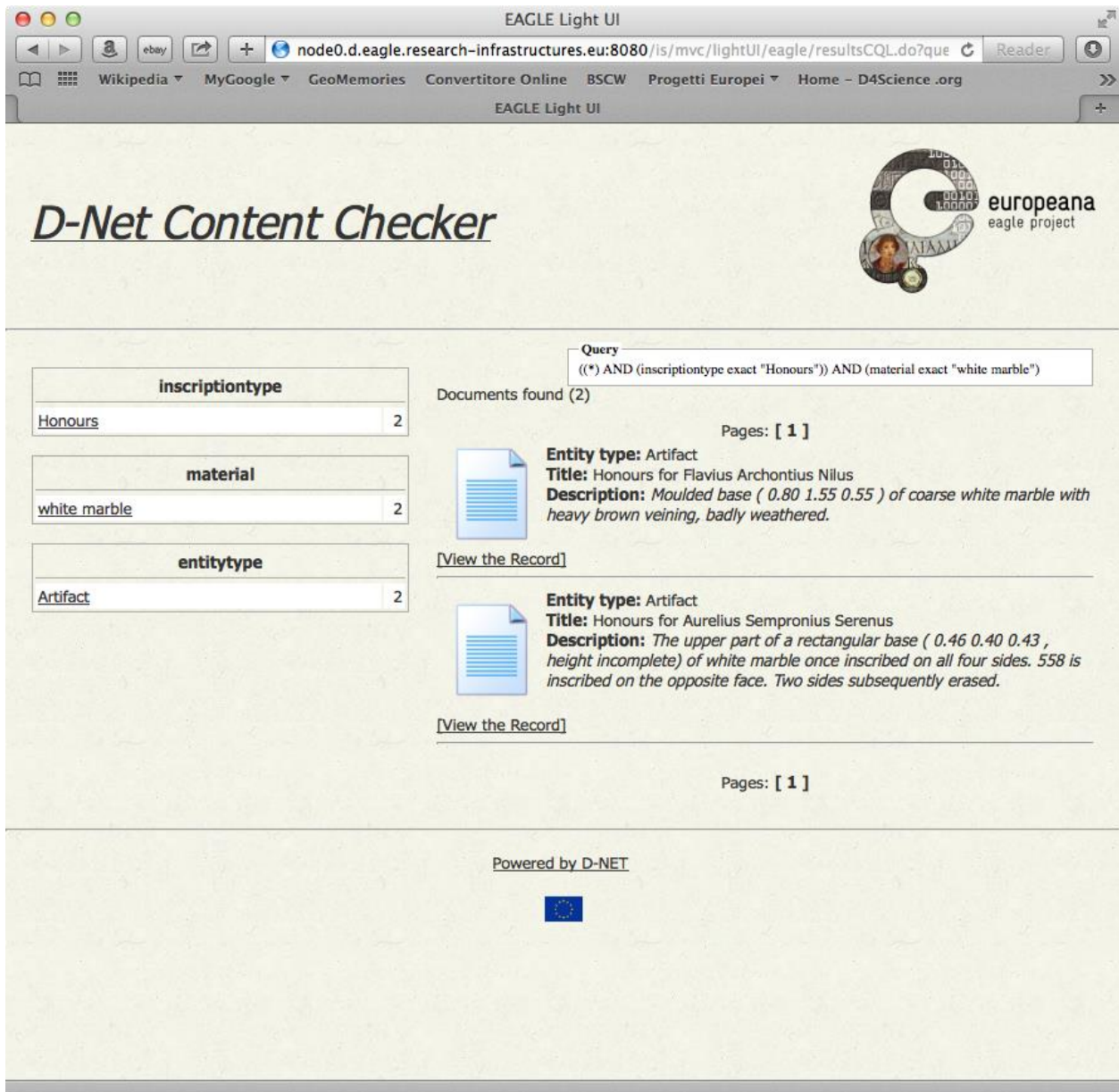
 **Entity type:** Artifact
Title: HD000034

Figure 5-2 Content Checker – Retrieving a set of results



inscriptiontype
Honours 2

material
white marble 2

entitytype
Artifact 2

Query
((*) AND (inscriptiontype exact "Honours")) AND (material exact "white marble")

Documents found (2)

Pages: [1]

Entity type: Artifact
Title: Honours for Flavius Archontius Nilus
Description: Moulded base (0.80 1.55 0.55) of coarse white marble with heavy brown veining, badly weathered.

[\[View the Record\]](#)

Entity type: Artifact
Title: Honours for Aurelius Sempronius Serenus
Description: The upper part of a rectangular base (0.46 0.40 0.43 , height incomplete) of white marble once inscribed on all four sides. 558 is inscribed on the opposite face. Two sides subsequently erased.

[\[View the Record\]](#)

Pages: [1]

Powered by D-NET


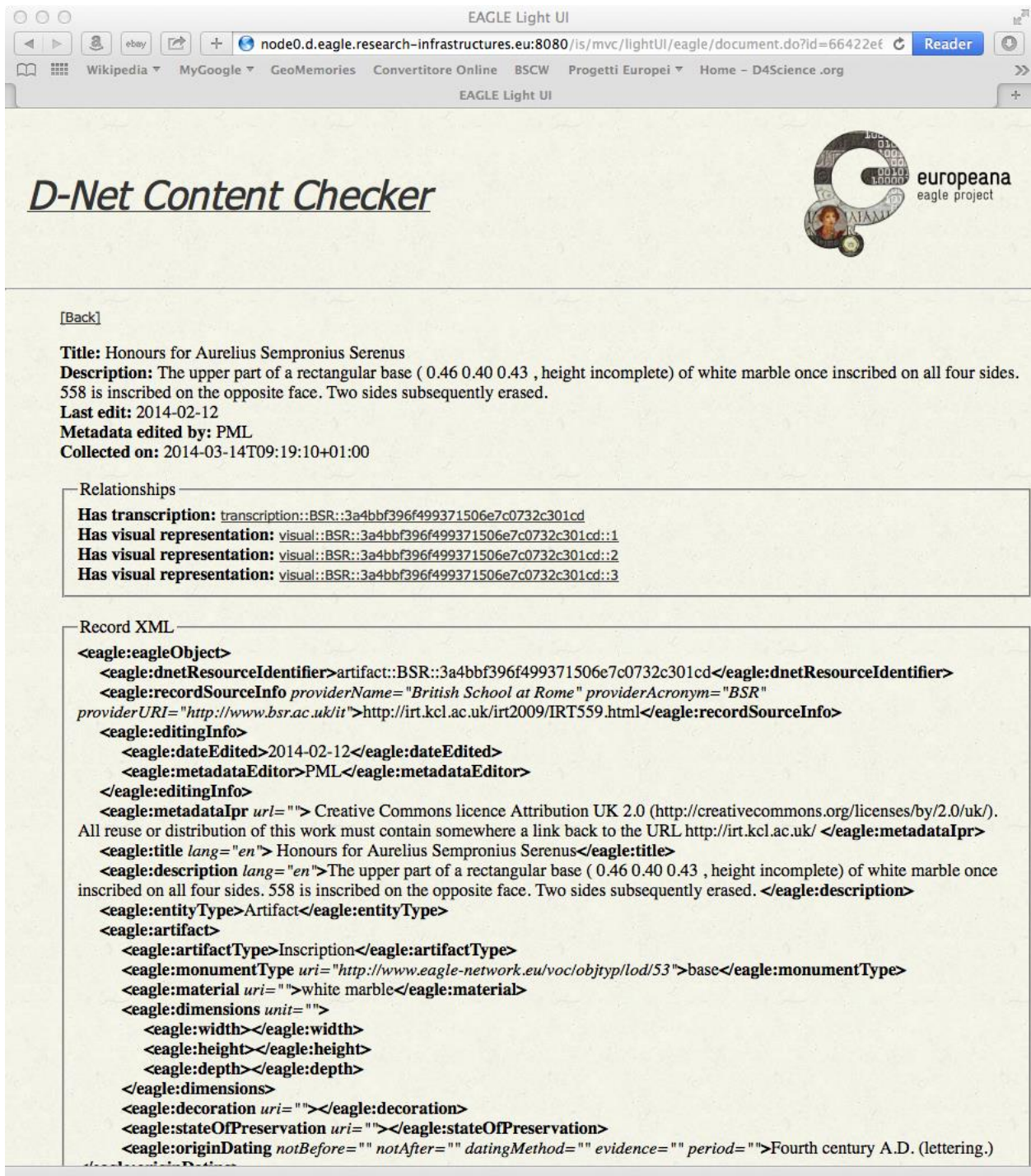


Figure 5-3 Content Checker – Refining a query



D-Net Content Checker

[Back]

Title: Honours for Aurelius Sempronius Serenus
Description: The upper part of a rectangular base (0.46 0.40 0.43 , height incomplete) of white marble once inscribed on all four sides. 558 is inscribed on the opposite face. Two sides subsequently erased.
Last edit: 2014-02-12
Metadata edited by: PML
Collected on: 2014-03-14T09:19:10+01:00

Relationships

Has transcription: [transcription::BSR::3a4bbf396f499371506e7c0732c301cd](#)
Has visual representation: [visual::BSR::3a4bbf396f499371506e7c0732c301cd::1](#)
Has visual representation: [visual::BSR::3a4bbf396f499371506e7c0732c301cd::2](#)
Has visual representation: [visual::BSR::3a4bbf396f499371506e7c0732c301cd::3](#)

Record XML

```
<eagle:eagleObject>
  <eagle:dnetResourceIdentifier>artifact::BSR::3a4bbf396f499371506e7c0732c301cd</eagle:dnetResourceIdentifier>
  <eagle:recordSourceInfo providerName="British School at Rome" providerAcronym="BSR"
  providerURI="http://www.bsr.ac.uk/it">http://irt.kcl.ac.uk/irt2009/IRT559.html</eagle:recordSourceInfo>
  <eagle:editingInfo>
    <eagle:dateEdited>2014-02-12</eagle:dateEdited>
    <eagle:metadataEditor>PML</eagle:metadataEditor>
  </eagle:editingInfo>
  <eagle:metadataIpr url=""> Creative Commons licence Attribution UK 2.0 (http://creativecommons.org/licenses/by/2.0/uk/).
  All reuse or distribution of this work must contain somewhere a link back to the URL http://irt.kcl.ac.uk/ </eagle:metadataIpr>
  <eagle:title lang="en"> Honours for Aurelius Sempronius Serenus</eagle:title>
  <eagle:description lang="en">The upper part of a rectangular base ( 0.46 0.40 0.43 , height incomplete) of white marble once
  inscribed on all four sides. 558 is inscribed on the opposite face. Two sides subsequently erased. </eagle:description>
  <eagle:entityType>Artifact</eagle:entityType>
  <eagle:artifact>
    <eagle:artifactType>Inscription</eagle:artifactType>
    <eagle:monumentType uri="http://www.eagle-network.eu/voc/objtyp/lod/53">base</eagle:monumentType>
    <eagle:material uri="">white marble</eagle:material>
    <eagle:dimensions unit="">
      <eagle:width></eagle:width>
      <eagle:height></eagle:height>
      <eagle:depth></eagle:depth>
    </eagle:dimensions>
    <eagle:decoration uri=""></eagle:decoration>
    <eagle:stateOfPreservation uri=""></eagle:stateOfPreservation>
    <eagle:originDating notBefore="" notAfter="" datingMethod="" evidence="" period="">Fourth century A.D. (lettering.)
  </eagle:artifact>
</eagle:eagleObject>
```

Figure 5-4 Content Checker – Accessing a record

6 APPENDIX C – IMAGE RECOGNITION AND SIMILARITY SEARCH SERVICE AND API

6.1 IMAGE RECOGNITION

Service address: <http://virserv101.isti.cnr.it/fma/services/IRServices/recognize>

Please note that the service address is temporary, and it will change when the services are deployed on Eagle servers.

Supported images formats: **JPG, PNG**

Image encoding: either **binary** or **Base64** encoding (by Apache Commons Codec library).

Parameters:

- **img** *inputstream* of an image (mandatory).
- **correlationId** optional

any other parameters will be ignored (in this release).

Returns:

- JSON response (see *JSON Responses*)

HTML Call Example:

```
<form method="POST" enctype="multipart/form-data" name="test" action="
http://virserv101.isti.cnr.it/fma/services/IRServices/recognize">
  Query <input name="img" type="file">
  <input type="submit" value="Search" name="submit">
</form>
```

HTML Testing page:

<http://virserv101.isti.cnr.it/fma/recognizeTesting.html>

6.2 SIMILARITY SEARCH

Service address: <http://virserv101.isti.cnr.it/fma/services/IRServices/searchSimilar>

Please note that the service address is temporary, and it will change when the services are deployed on Eagle servers.

Supported images formats: **JPG, PNG**

Image encoding: either **binary** or **Base64** encoding (by Apache Commons Codec library).

Parameters:

- **img** *inputstream* of an image (mandatory)
- **correlationId** optional

- **nResults** optional, number of results (default 30)

any other parameters will be ignored (in this release).

Returns:

- JSON response (see *JSON Responses*).

HTML Call Example:

```
<form method="POST" enctype="multipart/form-data" name="test" action="
http://virserv101.isti.cnr.it/fma/services/IRServices/searchSimilar">
  Query <input name="img" type="file">
  Num of Results <input name="nResults" type="text" size="5">
    <input type="submit" value="Search" name="submit">
</form>
```

HTML Testing page:

<http://virserv101.isti.cnr.it/fma/similarityTesting.html>

6.3 EXAMPLES

6.3.1 Java Examples

Eclipse project: **FMAclient**

This project contains some Java code and images to test the FMA services.

It includes the classes to wrap the JSON responses into Java code. It leverages *Google Gson* library to deserialize JSON messages. The code should work also on Android platforms.

- **ResponseCodes** contains the following response codes:
 - *RESPONSE_OK* = 200
 - *RESPONSE_NO_MATCH_FOUND* = 300
 - *RESPONSE_SERVER_ERROR* = 400
- **RecognizerResponse** contains the response of an image recognition query. It includes the result set as list of *RecognizerResult*.
- **RecognizerResult** contains the result set of an image recognition query.
- **SimilaritySearchResponse** contains the response of an image similarity search. It includes the result set as list of *SimilaritySearchResults*.
- **SimilaritySearchResults** contains the result set of an image similarity search.
- **IRClient** is just a simple class to show how to call FMA Services.
- **RecognizerExample** is a simple image recognition example.

- **SimilaritySearchExample** is a simple image similarity search example.

6.3.2 Testing Images

The `EDR_Images` folder contains a test collection of 100 images from EDR archives.

6.3.3 JSON Responses

6.3.3.1 Image Recognize JSON Example

JSON response complies with the requirements in *EAGLE_Adv_Architecture*.

The only relevant fields are

- **responseCode**
- **score**
- **id**
- **linkToCP**

The other ones have dummy values (at the moment)

```
{
  "responseCode":200,
  "recognizerResult":[
    {
      "score":0.892,
      "id":"000034",
      "title":"Lorem ipsum dolor sit amet",
      "inscriptionType":"Lorem ipsum dolor sit amet",
      "objectType":"Lorem ipsum dolor sit amet",
      "material":"Lorem ipsum dolor sit amet",
      "findSpotRegion":"Lorem ipsum dolor sit amet",
      "findSpotCity":"Lorem ipsum dolor sit amet",
      "date":"Lorem ipsum dolor sit amet",
      "text":"Lorem ipsum dolor sit amet",
      "translation":"Lorem ipsum dolor sit amet",
      "linkToCP":"http://www.edr-
edr.it/edr_programmi/res_complex_comune.php?do\u003dbook\u0026id_nr\u003dEDR000034"
    }
  ]
}
```

6.3.3.2 Image Similarity Search JSON Example

JSON response complies with the requirements in *EAGLE_Adv_Architecture*.

The only relevant fields are

- **responseCode**
- **score**
- **id**
- **thumbnail**

The other ones have dummy values (at the moment)

```
{  
  "responseCode":200,  
  "searchResults":[  
    {  
      "score":0.868,  
      "id":"000035",  
      "title":"Lorem ipsum dolor sit amet",  
      "thumbnail":"http://virserv101.isti.cnr.it/eagle-images/000035.jpg",  
      "text":"Lorem ipsum dolor sit amet",  
      "translation":"Lorem ipsum dolor sit amet",  
      "findSpotRegion":"Lorem ipsum dolor sit amet",  
      "findSpotCity":"Lorem ipsum dolor sit amet",  
      "date":"Lorem ipsum dolor sit amet"  
    },  
    {  
      "score":0.868,  
      "id":"110477",  
      "title":"Lorem ipsum dolor sit amet",  
      "thumbnail":"http://virserv101.isti.cnr.it/eagle-images/110477.jpg",  
      "text":"Lorem ipsum dolor sit amet",  
      "translation":"Lorem ipsum dolor sit amet",  
      "findSpotRegion":"Lorem ipsum dolor sit amet",  
      "findSpotCity":"Lorem ipsum dolor sit amet",  
      "date":"Lorem ipsum dolor sit amet"  
    },  
    {  
      "score":0.868,  
      "id":"005070",  
      "title":"Lorem ipsum dolor sit amet",  
      "thumbnail":"http://virserv101.isti.cnr.it/eagle-images/005070.jpg",  
      "text":"Lorem ipsum dolor sit amet",  
      "translation":"Lorem ipsum dolor sit amet",  
      "findSpotRegion":"Lorem ipsum dolor sit amet",  
      "findSpotCity":"Lorem ipsum dolor sit amet",  
      "date":"Lorem ipsum dolor sit amet"  
    }  
  ]  
}
```