



## **D2.0.4 THE ASSETS APIS**

---

### **Advanced Search Services and Enhanced Technological Solutions for the European Digital Library**

Grant Agreement Number: 250527

Funding schema: **Best Practice Network**

Deliverable D2.0.4 WP2.0

---

<Report>

V.1.1– 15 April 2011

Document. ref.: ASSETS.D2.0.4.ENG.WP2.0.V1.1

**Programme Name:** ..... ICT PSP  
**Project Number:** ..... 250527  
**Project Title:**..... ASSETS  
**Partners:**..... Coordinator: ENG (IT)  
 Contractors:  
**Document Number:** ..... D2.0.4  
**Work-Package:**..... WP2.0  
**Deliverable Type:** ..... Report  
**Contractual Date of Delivery:** ..... 28-february-2011  
**Actual Date of Delivery:** ..... 15-april-2011  
**Title of Document:** ..... The ASSETS APIs  
**Author(s):** ..... Luigi Briguglio (ENG);  
 ..... Sergiu Gordea and Andrew Lindley (AIT);  
 ..... Efstratios Tzoannos (ATC);  
 ..... Carlo Meghini, Franco Alberto Cardillo, Andrea Esuli,  
 Fabrizio Falchi, Diego Ceccarelli, Paolo Bolettieri, Nicola  
 Aloia, Cesare Concordia(CNR);  
 ..... Víctor Valdés, Fernando López, José M. Martínez, Jesús  
 Bescós, Pablo Castells, Miguel Ángel García (UAM);  
 ..... Oscar Paytuy (BMAT);  
 ..... Michalis Lazaridis (EKETA CERT);  
 ..... Abdelkrim BELOUED (INA);  
 ..... Nicolas Spyratos, Tsuyoshi Sugibuchi (UPS);  
**Approval of this report** ..... APPROVED

**Summary of this report:** ..... see Executive Summary

**History:**..... see Change History

**Keyword List:** ..... ASSETS, Models, Services, Interfaces

**Availability** ..... This report is:  
 X public  
 ### limited to ASSETS consortium distribution  
 ### limited to EU Programme distribution  
 ### restricted  
 ### internal

**Change History**

Version	Date	Status	Author	Description
0.1	21-march-2011	Draft	LB (ENG)	Revision of the contributions from D2.0.2 and System Architecture, according to sw code.
0.2	23-march-2010	Draft	LB (ENG)	Release for dev team review
0.3	31-march-2011	Draft	LB (ENG)	Revision based on dev team feedback
0.6	8-april-2011	Draft	LB (ENG)	Revision based on peer-review feedback
1.0	28-march-2011	Draft	LB (ENG)	Pre Final Release
1.1	15-april-2011	Final	SG (AIT)	Final Release



## Table of Contents

<b>1. INTRODUCTION</b>	<b>6</b>
1.1 HOW TO READ THIS DOCUMENT	7
<b>2. THE ASSETS NEEDS AND CONSTRAINTS</b>	<b>8</b>
2.1 METADATA ENRICHMENT, HETEROGENEITY REDUCTION, INFORMATION EXTRACTION AND CLASSIFICATION	8
2.2 INDEXING, RANKING AND RETRIEVAL	10
2.3 DIGITAL PRESERVATION	12
2.4 BROWSING AND CONTENT CHARACTERISATION	14
2.5 COMMUNITY	14
<b>3. THE ASSETS SERVICES</b>	<b>16</b>
3.1 THE INGESTION SERVICES	16
3.1.1 <i>The ASSETS Approach and Proposal</i>	16
3.2 INDEXING, RANKING AND RETRIEVAL SERVICES	20
3.2.1 <i>The ASSETS Approach and Proposal</i>	20
3.3 DIGITAL PRESERVATION SERVICES	23
3.3.1 <i>The ASSETS Approach and Proposal</i>	23
3.4 THE BROWSING AND CONTENT CHARACTERISATION SERVICES	25
3.4.1 <i>The ASSETS Approach and Proposal</i>	25
3.5 THE COMMUNITY SERVICES	27
3.5.1 <i>The ASSETS Approach and Proposal</i>	27
3.6 USER INTERFACE OF THE ASSETS PORTAL	28
3.6.1 <i>The ASSETS Approach and Proposal</i>	28
<b>4. THE ASSETS DATA MODELS AND INTERFACES</b>	<b>52</b>
4.1 SYSTEM ARCHITECTURE OVERVIEW	52
4.1.1 <i>System Components</i>	53
4.1.2 <i>ASSETS Software Architecture</i>	54
4.2 INTEGRATION LAYER AND INTERFACES	55
4.2.1 <i>ASSETS Data-Model Component</i>	55
4.2.2 <i>Core Data Model</i>	57
4.2.3 <i>ASSETS Common API and Common Server API Components</i>	59
4.2.4 <i>Common Server and Common Client</i>	63
4.2.5 <i>Notification and Taxonomy Models and Interfaces</i>	65
4.2.6 <i>Session Management and Identification</i>	70
4.3 THE INGESTION MODELS AND INTERFACES	71
4.3.1 <i>The Metadata Cleaning Service Models and Interfaces</i>	71
4.3.2 <i>Knowledge Extraction Models and Interfaces</i>	74
4.3.3 <i>Metadata Classification Models and Interfaces</i>	77
4.3.4 <i>Ingestion Workflow Management Models and Interfaces</i>	80
4.3.5 <i>Post-Ingestion Processing</i>	87
4.4 THE INDEXING, RANKING AND RETRIEVAL MODELS AND INTERFACES	88
4.4.1 <i>Post-query processing Models and Interfaces</i>	88
4.4.2 <i>Metadata Based Ranking Models and Interfaces</i>	89
4.4.3 <i>Text Indexing and Retrieval Service</i>	91
4.4.4 <i>Images Indexing and Retrieval Service</i>	94
4.4.5 <i>3D-Model Indexing and Retrieval Services</i>	98



4.4.6	<i>Audio Indexing and Retrieval Service</i>	105
4.4.7	<i>Video summarisation, indexing and retrieval</i>	107
4.5	THE DIGITAL PRESERVATION MODELS AND INTERFACES	110
4.5.1	<i>The Risk Management Models and Interfaces</i>	110
4.5.2	<i>The Normalisation Models and Interfaces</i>	119
4.5.3	<i>The Notification Models and Interfaces</i>	124
4.6	THE BROWSING AND CONTENT CHARACTERISATION MODELS AND INTERFACES	126
4.6.1	<i>Annotation Propagation Service Models and Interfaces</i>	127
4.6.2	<i>Training Service Models and Interfaces</i>	129
4.6.3	<i>Manual Annotation and Annotation Correction Service Models and Interfaces</i>	130
4.6.4	<i>Content Selection Service Models and Interfaces</i>	131
4.6.5	<i>Relevance feedback service Models and Interfaces</i>	133
4.6.6	<i>Log Analysis Service Models and Interfaces</i>	134
4.7	THE COMMUNITY MODELS AND INTERFACES	137
4.7.1	<i>User Generated Content Models and Interfaces</i>	138
4.7.2	<i>Taxonomy-based Notification Service Models and Interfaces</i>	148
4.7.3	<i>Personalisation service Models and Interfaces</i>	151
	<b>APPENDIX 1 - ENRICHMENT SERVICES TRAINING DATA FORMAT</b>	<b>153</b>
<b>5.</b>	<b>REFERENCES</b>	<b>154</b>

## Executive Summary

---

This is a technical document detailing the ASSETS architecture and API for each component.

It integrates and extends results of the first year mainly from T2.0.4 “Platform design and implementation guidelines” and T2.0.5 “API Specifications”, but it introduces technical aspects of all the software services defined, analysed, implemented and tested in ASSETS WP2.1, WP2.2, WP2.3, WP2.4, WP2.5 and WP3.2.

This document provides the following information regarding the Assets Services:

- The rationale behind the service choices;
- The approach and methodology of proposed solutions;
- The services description and the definition of their interfaces (APIs);
- The data models and data flows exchanged between services.

This documentation is the basis for the development activities, because identifies the components, their responsibilities data models and interfaces. Through the iteration, as yet occurred during the first year, the data models and interfaces will be refined and enriched with further details.

More, even if an improved User Interface is expected on next months, yet in this document ASSETS is able to present some anticipations with preliminary results and mock-ups, carried out during this year. Those are shown for better describing and clarifying how the ASSETS services are/will be used for improving access and usability of Europeana.

## 1. Introduction

The goal of the ASSETS projects is to improve the usability of Europeana (the European Digital Library platform) by developing, implementing and deploying robust and innovative services focusing on improvement of **search, browsing, usability and personalization functionality** of the Europeana platform. These services are designed to be reused by any digital library. The concrete enhancements proposed by ASSETS Services include the followings: i) Searching multimedia objects based on metadata and on content similarity; ii) Ranking algorithms for improved result display; iii) Browsing multimedia objects fast and comfortable navigation through semantic categories; iv) Enhanced Graphical User Interfaces specially designed for interacting with multimedia objects; v) Planning Long-term Access to digital information; vi) Ingestion of metadata, which is enriched by normalization, cleaning, knowledge extraction and mapping to a standardized format; vi) Supporting Community and the user generated contents.

Based on results documented in the D2.0.1 "Requirements Specification" [1], the Milestone 12 "System Architecture" [2], and the D2.0.2 "Interface Specifications and System Design" [7], this document provides the description, the definition of the software interfaces, the data models and data flows used by ASSETS Services. Documentation and models are synchronised with the current release of the software implementation.

All this information is collected by the activities carried out within all technical workpackages of Stream2 and Stream 3:

- Ingestion models and tools from WP2.1;
- Indexing, Ranking and Retrieval models and tools from WP2.2;
- Digital Preservation models and tools from WP2.3;
- Browsing and Content Characterisation models and tools from WP2.5;
- Community models and tools from WP3.2.

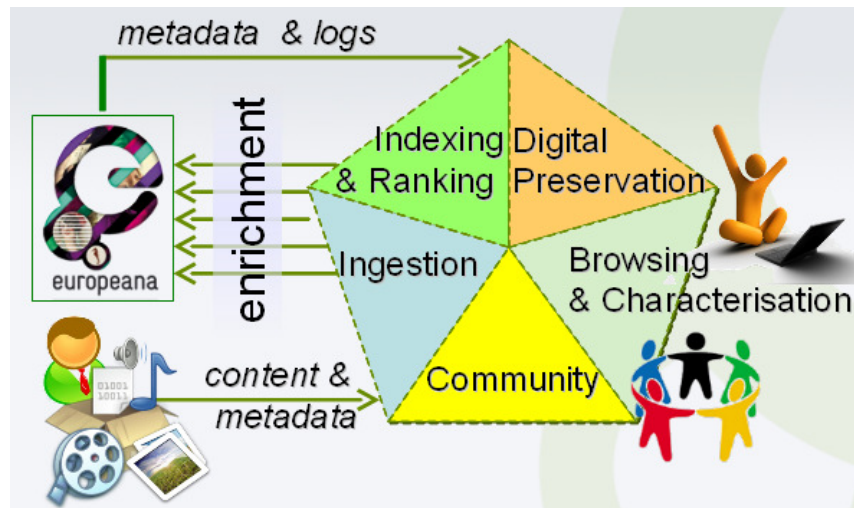


Figure 1 - The ASSETS Services

For collecting this information, a Service Specification Template has been created, based on the experience gained within the WP2.3 and in particular the Milestone 27 "Digital Preservation Service Design".

Starting from these descriptions, the current document proceeds with the identification of the needs and constraints, which are applied for characterising the ASSETS context and its technical challenges. Those latter justify the decisions behind the functionality provided by the ASSETS services and their integration. A detailed data model is defined for providing a common understanding of the key concepts used for the implementation of ASSETS Services, together with the related programming interfaces.

## 1.1 How to read this document

The document is divided in three main parts: the rationale of the services, the approach and methodology, and finally the data models.

The section 2 "The ASSETS Needs and Constraints" identifies the set of needs and constraints for ASSETS Services by grouping the issues in the five research activities of the ASSETS project. That allows describing the rationale behind the decisions for the system and its services design, the models, the constraints and the features.

This section is a brief overview of the ASSETS project and allows the reader to understand the context of the project and the issues to be addressed by each research activity. The nature of this section doesn't require a technical background for its understanding.

Section 3 "The ASSETS Services" allows the reader to understand the approach adopted by each research activity (often according to the state of the art and standards) for identifying the proposed solution to the issues described in the previous section. Moreover, preliminary results on User Interface improvement are anticipated through mockups. They allow us to indicate how will be the ASSETS Services used for improving the access and usability of Europeana portal.

Finally, by using the information available from those above parts, the last part of the document enters into the technical details by identifying and analysing the main concepts of the application domain. These concepts are modelled by using UML Class diagrams which explicitly represent their properties and their relationships.

The modelled concepts represent the so-called ASSETS Data Models. They are described in section 4 "The ASSETS Data Models and Interfaces" and drive the identification of Interfaces (i.e. APIs) for the ASSETS Services.

## 2. The ASSETS Needs and Constraints

---

ASSETS project aims at improving and extending the features of Europeana portal. Within this section we identify the set of needs and constraints, and we group them in 5 research activities:

1. Metadata Enrichment, Heterogeneity Reduction, Knowledge Extraction and Classification;
2. Indexing, Ranking and Retrieval
3. Digital Preservation
4. Browsing and Content Characterisation
5. Community Services

That allows describing the rationale behind the decisions for the system and its services, the models, the constraints and the features.

### 2.1 Metadata Enrichment, Heterogeneity Reduction, Information Extraction and Classification

The **ASSETS portal** will act as an integration system acquiring information, (i.e. metadata content) from different entities (e.g. museums, libraries, archives, etc.) which are called with the general term of Content Providers (CPs) located in different European countries. Since different CPs organize their digital archives using different formats, the metadata gathered within the ASSETS project will suffer from a great heterogeneity. The metadata submitted by CPs is characterized by the following properties:

- It is expressed in different languages. Basically, ASSETS will process metadata containing words and texts written in the most of the European languages.
- It uses different formats for the textual representation of many values, such as, for example, the dates (e.g. dd-mm-yyyy or mm-dd-yyyy) or the dimensions of a physical object (e.g. different units measurement) .
- It is formatted according to different XML schemas (e.g., Dublin core-based, encoding archival description or entirely proprietary formats).
- They are likely to contain mistakes in the textual representations and descriptions. There are several types of mistakes the metadata can suffer from. We report two example cases that we plan to study. :
  - **Spelling mistakes.** Authors can be easily misnamed if they do not have a well known name in the annotator's native language. There might be spelling errors even if the author's name (or other texts) does not change among languages. In one of the first examples received, Mozart has been written as *Mozzart*.
  - **Aging-related mistakes and problems.** The metadata datasets of cultural heritage institutions have been produced over a long period of time by different archivists. Without digital preservation actions, it is likely that different archivists used (over a long period of time) different terms and ontologies for annotating similar or related metadata records. Furthermore, metadata





annotated many years ago might refer to ontologies that are no longer used.

Such heterogeneity represents a major difficulty in offering a satisfactory user experience on the ASSETS web site. In fact, if the metadata is ingested 'as it is', without any specific processing, the users would not benefit from the richness of such a large archive of cultural heritage digital object descriptions. The results returned to a specific query might miss some interesting results (for example, an Italian-speaking user might not receive the link to an Italian book whose metadata record is not expressed in Italian). The results might as well contain irrelevant result due to false matches among different languages or formats.



**Figure 2 - The rationale of Ingestion Service**

Europeana Foundation realised the need of integrating the tools that collectively cover the ingestion functionality into a unique framework. In particular, the need for having a unified ingestion workflow with a single access point was identified as a requirement for the Assets Project. Since this is a joint collaboration, the extended description for the design of this service will be provided within the Europeanalabs documentation.

Europeana metadata contains both structured and unstructured information. Structured information is provided by those metadata fields that identify well-specified type of information, e.g., "date", "creator", "language". Unstructured information is provided by those metadata fields that act as containers of generic information, e.g., "description".

The aim of the knowledge extraction service is to provide Europeana enrichment process with automatic knowledge extraction functionalities that enable identification of relevant information from unstructured metadata fields of collection objects.

Finally, the aim of "Metadata classification" is to develop a service for the automatic classification of metadata records according to the predefined taxonomy of semantic categories (classification schema).

The classification process consists of assigning a record to zero, one, or several categories (aka "classes", or "concepts", or "codes").

Europeana records are provided by many different content provider, which may (i) not use any classification schema for their data, (ii) use a very specific classification schema specifically tailored for particular purposes of the content provider, (iii) use a standard well-know classification schema for their data, either general-purpose (e.g., Library of the Congress Subject Headings, LCSH) or discipline-specific (e.g., Medical Subject Headings). Among these three cases the last one is certainly the preferred one for Europeana.

The metadata classification service will enable Europeana to automatically categorize the new records provided by content providers, and those already acquired by Europeana, following a set of general-purpose and/or discipline-specific classification schema.

The final goal of the task is making the searching and browsing experience on the part of the user more satisfactory; e.g.:

- user can navigate from record to concept and to other records belonging to same concept or sibling concepts;
- user can restrict search to records belonging to a specific concept;
- user can ask to group the search results according to the concepts they belong to.

## 2.2 Indexing, Ranking and Retrieval

The group of services provided by "Indexing, Ranking and Retrieval" addresses a wide variety of issues:

1. Post-query processing;
2. Metadata based ranking;
3. Text indexing and retrieval;
4. Image indexing and retrieval;
5. 3D model indexing and retrieval;
6. Audio indexing and retrieval;
7. Video summarisation, indexing and retrieval;

### Post-query processing

Users queries are often very short, ambiguous, and sometimes only exploratory. Thus, they do not suffice to describe the user *information need*. Query suggestion is fundamental for improving the search through the complex space of information provided by Europeana.

Furthermore, in the case of Europeana, identifying the possible interests of users might be beneficial for suggesting new topics that are related to the one s/he is currently exploring. For instance when searching for "Giotto" one can be interested in exploring also related topics, such as: "Cimabue", "Famous Young Artists", "Florence Art", etc. In other words, differently from what happens in the case of Search Engines, Europeana post-query processing might be helpful to enlarge somebody's vision.

### **Metadata based ranking**

Europeana indexes a massive amount of structured information. This requires special search functions, beyond full text search, that exploit this structure in order to provide better answers to users' queries.

In fact, full text search returns results (mostly) based on frequency of query terms in the collection and in the returned documents. This approach is not the most efficient one in the context of Europeana, where the indexed object is not a large document but a concise metadata record. For this reason, special search functions should have the capability for ranking metadata associated with the multi-media objects indexed by Europeana, and obviously, according to the user interests.

### **Text Indexing and Retrieval Service**

Europeana and Assets have many software components exploiting textual information. Some software components in Assets/Europeana access significant amounts of structured information to achieve their tasks. Such information include: submitted queries, popularity, recency, clicks and other aggregated data such as clustering of search queries, etc. This knowledge base is crucial for some of the Assets/Europeana components. This service will provide efficient access to such information.

Many Assets services exploit external information sources. Having this information smartly indexed, cleaned and promptly retrievable will boost the performance of search services.

### **Images Indexing and Retrieval Service**

When searching for a particular masterpiece by using text queries, the user has to know some very detailed information like title or creator of a work in order to find it in one search action. Therefore, there is a real need for being able to use images as queries for searching information about a digitized masterpiece (e.g. a snapshot of a painting taken when visiting a museum) .

### **3D--Model Indexing and Retrieval Services**

A user would like to search for 3D models geometrically similar to a query model. The query 3D model can be provided by the user or be the result of a previous search.

### **Audio Indexing and Retrieval Service**

A user would like to find the kind of audio he's looking for, but in most of the cases, the only way of accessing the audio content is via textual searches on the Dublin core metadata. This service offers the possibility to query objects by using the audio metadata, or by providing an audio file as query.

### **Video summarisation, indexing and retrieval**

Efficient and fast access to video content is often limited. Users need for condensed preview versions and advanced video search functionalities are typically unsatisfied.

The **assets** Services  
Indexing & Ranking: why?

Search service: entry point to Europeana, must guarantee:

- **Efficiency** - Response should be given in less than one second
- **Scalability** - Must serve several requests simultaneously, Must handle millions/billions documents
- **Effectiveness** - Response should contain what the user is looking for

Scalable Metadata and **similarity** based indexing and retrieval

Indexing & Ranking

updated metadata ↑  
logs ↓  
metadata & logs

Figure 3 - The rationale of Indexing Services

## 2.3 Digital Preservation

Usually Preservation is an issue which is considered “important” only for digital content which have a long lifecycle. For instance, they could be accessed after a long time since their ingestion and storage, and so a number of impacting events could occur and affect or damage their “long-term usability and/or access”:

- Changing technologies, including support for new media and data formats
- Changing of terminologies/knowledge of “designated communities”

Corrective preservation actions have to be properly planned and evaluated for avoiding side effects. Actions could affect more than the impacting events and/or carry out an irreversible process and loss forever the original content. So, it’s important to:

- Identify/Classify/Monitor potential impacting events
- Identify/Evaluate/Enact relative corrective preservation actions (plans)
- Notify impacting events and actions to the right actors involved in the preservation process
- Document the history of the digital object, as evidence for judging/auditing/certifying the goodness of “preservation archive” and the “authenticity” of archival objects.

For more clarity, it could be useful to describe a **“case in the real world”**.

The world heritage convention aims to safeguard our heritage, but sadly in one real case, we lost one cultural heritage which was not yet inscribed. The unfortunate scenario of the

explosion of the Buddhas of Bamiyan provides a real case scenario of the fragility of our heritage and the importance for its documentation and hence the preservation of that digital information for future generations.

After the explosion of the buddhas - UNESCO's partners the Institute of Geodesy and Photogrammetry - ETH Zurich, Switzerland made a 3D digital reconstruction of the buddhas with the remote sensing technique: photogrammetry. This 'digital heritage' remains the only available documentation of the buddhas to provide a true record of their original structure.

UNESCO has 3D model contents in PLY format which is mainly designed to store three dimensional data from 3D scanners of CH objects. Potentially impacting events may be:

- PLY format could become obsolete/ replaced by another format (e.g. VRML or MAYA8.x)
- Tools supporting that format could be no more available

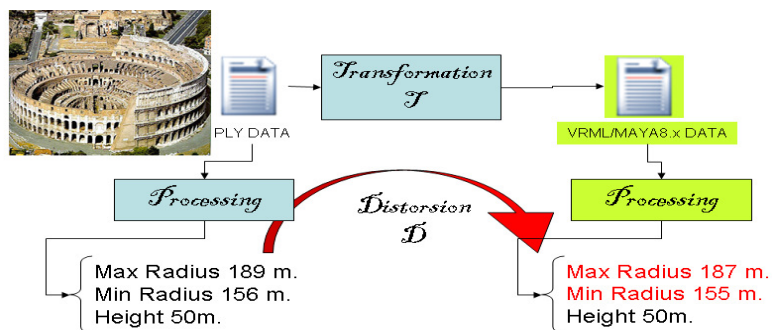


Figure 4 - Distortion in the Information introduced by a transformation of the Data

Usually, Emulation and Transformation which are applied as corrective preservation actions.

But, it's important to remark that transformation could introduce "distortions" in the information (aka Significant Properties) which may corrupt the usability of the object.

In order to evaluate the proper "transformation" we have to know which kind of information of the content we want to preserve, and the metrics for properly evaluating the process. In this perspective, who wants to preserve has to think in terms of "apply transformation  $T$  to content object  $O$  for preserving information  $I$  by accepting an error  $E$ " (e.g. preserve the main spatial information of the scanned monument).

The slide features the 'assets' logo in the top left and three green circles in the top right. The title is 'The assets Services Digital Preservation: why?'. On the left, a diagram shows a stack of binary code (0s and 1s) on a green surface. An arrow points down to a yellow document icon, then to a grey box labeled 'Transformation'. Another arrow points down to a blue document icon, which is followed by a blue starburst shape labeled 'Distorsion' (sic).

- Between the ingestion and the access a number of **impacting events could occur** and affect/damage their “long-term **usability & access**”
- For that reason it is fundamental a **continuous monitoring and maintenance** of the archived data
- **Corrective actions** have to be properly **planned** and evaluated for avoiding **side effects**
- Actions could affects more than the impacting events, and loss forever the original content

Figure 5 - The rationale of the Digital Preservation Services

## 2.4 Browsing and Content Characterisation

Many archives identify digital content by tagging and hyper-linking; however, the conventional browser model offers few possibilities for exploring the patterns that are inherent when information is classified in this way. Designers must therefore seek to enrich the user’s experience of complex data sets through:

- patterns of metadata,
- graphical visualization and iconography,

wherever possible allowing users to manipulate information according to their own interpretation.

## 2.5 Community

Traditional libraries have always served communities of users to find the information sources they need for their work; and, in turn, they enrich the library with the results of their own work. Digital Libraries are no exception. The ASSETS project will develop services aimed at supporting this information exchange. In particular, the project will address the content creation by re-use, a sophisticated form of notification based on taxonomies, and the personalization of search based on user preferences.

assets ooo

## The **assets** Services Community: why?

- Traditional libraries have always served communities of users
  - Users **find** in the library the **source** of information they need **for their work**;
  - and, in turn, they **enrich the library with the results** of their own work.
- ...Digital Libraries are **no exception**.



Figure 6 - The rationale of the Community Services

### 3. The ASSETS Services

---

This section allows the reader to understand the approach adopted by each ASSETS research activity (often according to the state of the art and standards) for identifying the proposed solution to the issues described in the previous section.

#### 3.1 The Ingestion Services

##### 3.1.1 *The ASSETS Approach and Proposal*

Most of algorithms that will be experimented and implemented will be based on a Machine Learning (ML) approach, particularly by employing supervised learning methods. Within this approach, a learning machine induces a classification model by analyzing a training set of metadata records that are considered to be correctly categorized by human users.

In particular:

- for the Metadata Cleaning service, the training set should contain examples of typical unclear or wrong constructs together with their correspondent corrections;
- for the Knowledge Extraction service, the training set should contain annotated descriptions. Basically, the relevant entities (e.g. names of persons or places) should be marked with specific tags;
- for the Metadata Classification service, the training set should contain metadata records associated to the correct categories;
- the Ingestion team has already prepared XML schemas the CPs need to use in order to provide training sets before the Europeana Data Model (EDM) is finalized. Once the EDM has been finalized, the Ingestion team will migrate the current XML schemas to the fresh data format.

ML methods have already been proven to be very effective in knowledge extraction and classification tasks. Even if some established algorithms will have to be tailored for the specific needs of the ASSETS project, it is reasonable to expect very good results if enough annotated metadata are provided by the CPs.

However, supervised ML algorithms are not the only methods that will be experimented in the Ingestion module. Possible alternatives include unsupervised learning methods (that do not require a training set) and non-adaptive methods for the simpler cases.

In general, we will limit the number of cases where a non-adaptive approach will be used. In fact, even if in some cases a rule-based system would be quite effective and much simpler to implement, it would not adapt to new datasets provided by CPs that decide to join ASSETS (or Europeana).

The ASSETS proposal for the ingestion issues is to provide implementations of advanced services with functionalities able to clean, enrich, extract knowledge, and classify the metadata records coming from the CPs involved in the ASSETS project and the metadata records currently indexed by Europeana.

The ASSETS ingestion services will:





- clean and perform a basic enrichment of the metadata (using URIs pointing to controlled vocabularies and authority files) through the **Metadata Cleaning** service;
- extract knowledge and enrich the original metadata with the new information through the **Knowledge Extraction** service;
- classify the metadata under a well-defined classification taxonomy through the **Metadata Classification** service;

### **Metadata Cleaning**

The service has the following set of goals:

1. correct part of the errors in the metadata;
2. normalize the values of specific fields by using the same textual representation in all of the datasets received by the ASSETS project;
3. perform a basic enrichment of specific elements of the metadata records.

1. In order to be able to correct generic errors affecting the metadata, the service will use algorithms based on supervised ML approaches. On simple values non-adaptive approaches will be experimented. For example, if a year has been specified as "1797,", this would be corrected by removing the comma. In particular, when the EDM is finalized and used by the CPs in submitting their metadata, the EDM semantics will be exploited to correct simple errors in a larger number of elements.

2. Two types of normalization procedures will be implemented. On simple cases, like, for example, metadata elements whose values do not vary greatly with the language used in expressing them (e.g. the size), simple rule-based systems will be used. On more complex cases, like the elements whose values vary with the language, an adaptive approach (supervised ML) is better suited.

3. Some metadata elements related to names of persons or places will be enriched by adding a URI to controlled vocabularies and authority files. Obviously such enrichment will be performed only on those values for which those resources are available. By adding a URI it is possible to overcome part of the limitations imposed by the multilingual nature of the data and by the high variance characterizing the values in the datasets. For example:

- the Greek philosopher *Plotinos* can also be referred to as *Plotin*, *Plotinus*, *Plotino*.
- the English capital *London* can also be called *Londra* or *Londre*.
- the *Monna Lisa* is also known as *La Gioconda* or *La Joconde*. Furthermore, it has been painted by *Leonardo da Vinci*, also known as *Leonardo di ser Piero da Vinci*.

As shown in the example above, the use of URIs can greatly improve the number of relevant matches returned to a user query. For example, by using a URI pointing to Plotinus in the VIAF file ([www.viaf.org](http://www.viaf.org)), if a user specify 'Plotin' as query term, metadata records originally containing 'Plotino' or 'Plotinus' would also be included in the results.

The ingestion service will have the responsibility of **reducing the inhomogeneities** in the metadata and **enrich their description** by **extracting** from the metadata as much information as possible. The service will be organized into a processing stream composed by three steps: **metadata cleaning, knowledge extraction, and metadata classification**.

The first step **Metadata Cleaning** will correct (basic) errors and perform a basic enrichment of a well-defined set of elements. After having corrected the errors in the values, some



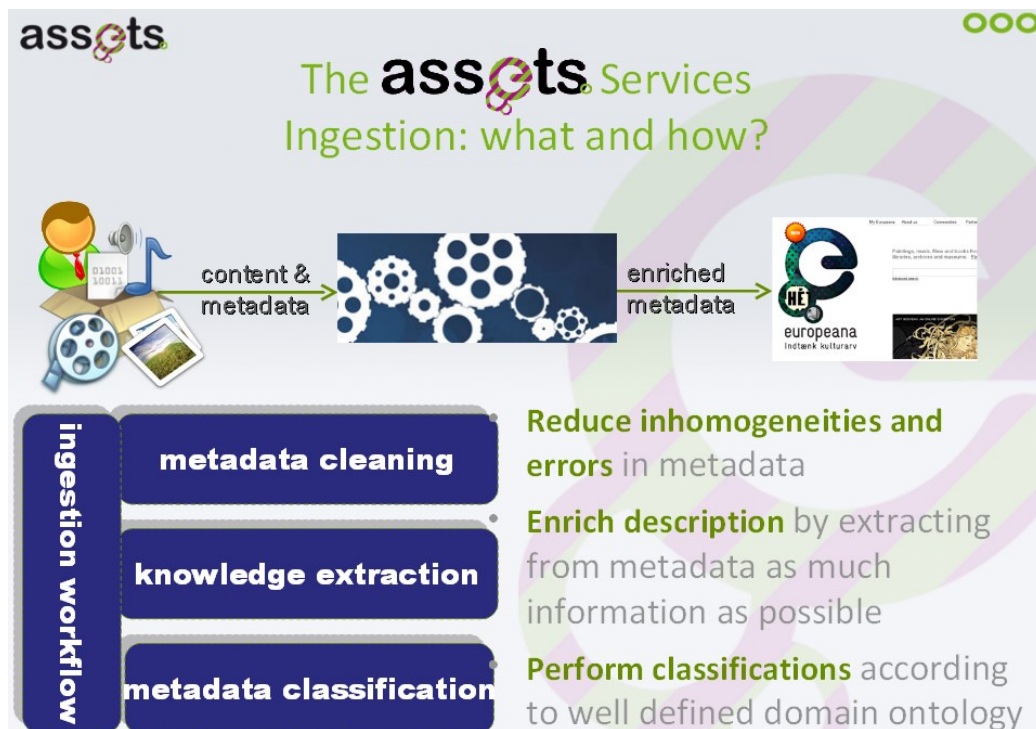
elements, like the one containing names of persons or names of places, will be "enriched" by adding a link to an entry in a controlled vocabulary or an authority files. For example, the name of an author can be "substituted" with a link (URI) to an entry in an authority file containing the author's name expressed in several languages.

**Knowledge Extraction**

Usually the metadata records have one or more descriptive fields containing free-form, unstructured textual description of a physical object. The second step **Knowledge Extraction** will extract information by such long textual description. The type of extracted information will depend on the metadata domain and cannot be specified at this moment.

**Metadata Classification**

Even after the cleaning and the extraction of knowledge have been accomplished, the metadata remain a largely disorganized set. The ASSETS users might benefit from an organization of the metadata under a well defined semantic taxonomy, like, for example, the Library of Congress Classification scheme. The third step **Metadata Classification** will associate each metadata record to one or more categories in classification schemes that are yet to be chosen.



**Figure 7 – The ASSETS Ingestion Services**

For further details on how CPs have to format their data in order to submit training data to the enrichment services of WP2.1 see Appendix 1 - Enrichment Services Training Data Format.

The ASSETS proposal for the ingestion issues is to provide implementations of advanced services with functionalities able to clean, enrich, extract knowledge, and classify the metadata records coming from the CPs involved in the ASSETS project and the metadata



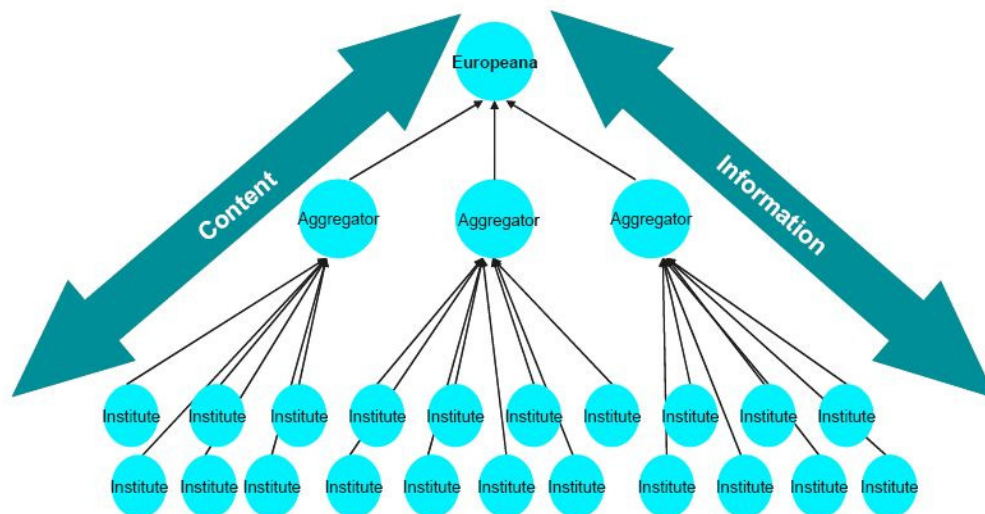
records currently indexed by Europeana.

The ASSETS ingestion services will:

- clean and perform a basic enrichment of the metadata (using URIs pointing to controlled vocabularies and authority files) through the Metadata Cleaning service.
- extract knowledge and enrich the original metadata with the new information through the Knowledge Extraction service.
- classify the metadata under a well-defined classification taxonomy through the Metadata Classification service

### ***Ingestion Workflow***

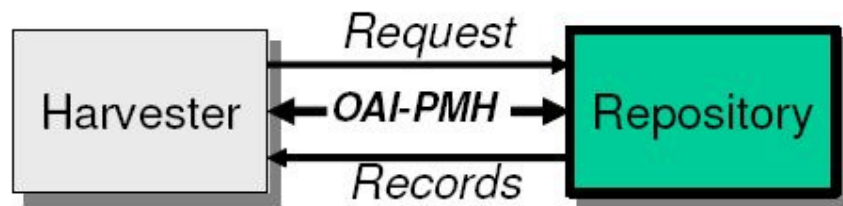
The Europeana web portal implements a search engine over the European cultural heritage. In order to provide this functionality, an index with the description of the masterpieces was created. This information is retrieved from the Content Providers (CPs). The model used for metadata aggregation is sketched in the following figure.



**Figure 8 – Aggregators in the Europeana organisation model**

Where the aggregators are organizations which integrate the data retrieved from content providers, and transform it into a representation compatible with the Europeana search index. See also Europeana Aggregator's Handbook [6].

The harvesting of the metadata is based on the Open Archives Initiative Protocol for Metadata Harvesting (for further details see <http://www.openarchives.org/pmh>). This protocol standardises the harvesting of metadata by defining a web service interface which provides descriptions of the collection objects in XML format.



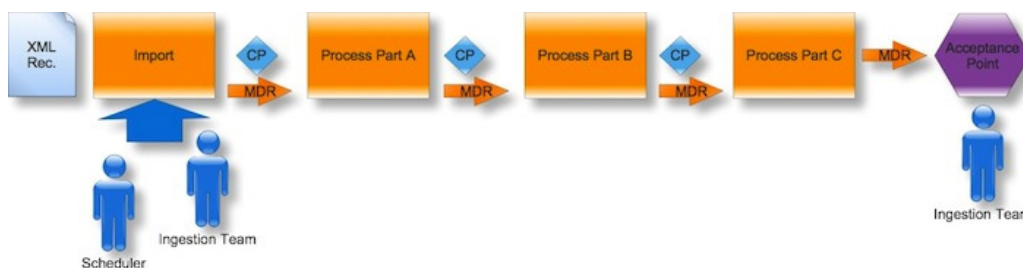
**Figure 9 - OAI-PMH Harvesting**

These XML files, retrieved through the OAI-PMH interface, are used as input for the ingestion workflow.

The ASSETS proposal for the ingestion workflow management will:

- Integrate execution of the metadata enrichment services in a standardized workflow - Ingestion Workflow Management;
- Build the multimedia index used for content based search functionality - Post-Ingestion Processing;

The ingestion framework will perform the necessary processing steps as much as possible in an autonomous way. The ingestion team is involved in the scheduling process, where the team needs/can prioritise certain collections or inform the system about a new collection which needs to be processed. After processing, the records remain in the acceptance point until a member of the ingestion team confirms them.



**Figure 10 - The Ingestion Process flow**

## 3.2 Indexing, Ranking and Retrieval Services

### 3.2.1 The ASSETS Approach and Proposal

The ASSETS proposal for the indexing and retrieval issue is to provide implementations of services with functionality such as:

- suggest queries to users mining query logs;
- enhance ranking using a ranking function designed for metadata;
- provide efficient access specifically tailored to structured information exploited by suggest and ranking services;
- enable 3D content-based indexing, search and retrieval.

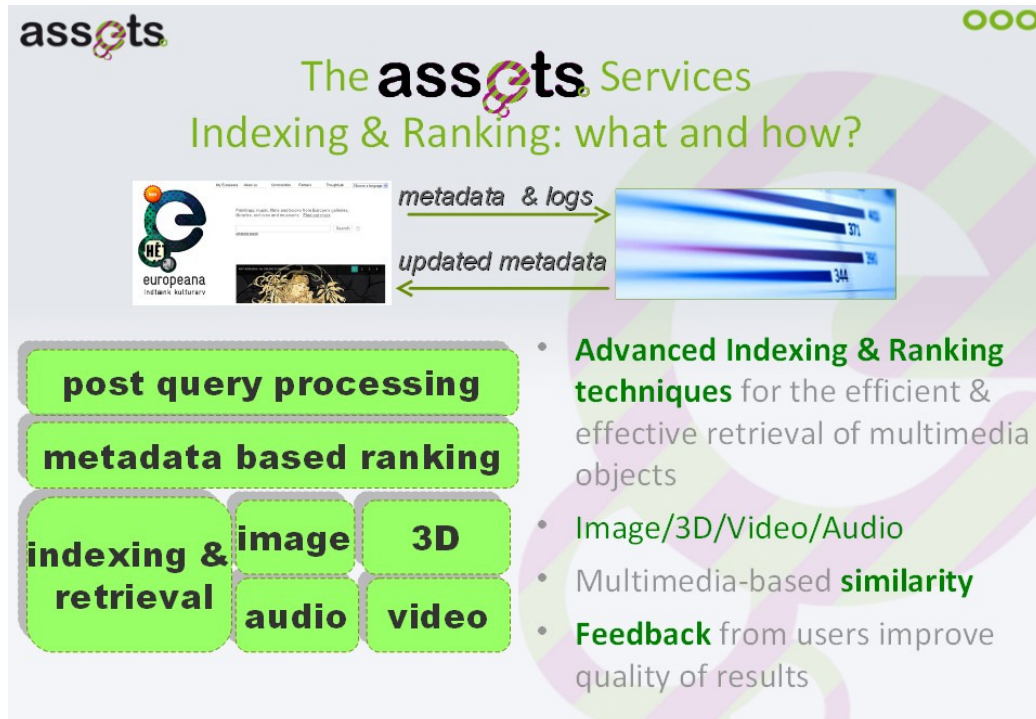


Figure 11 - The ASSETS Indexing, Ranking and Retrieval Services

As shown in section 2.2, for indexing, ranking and retrieval, there are a variety of issues which need to be addressed and focused by specific approaches and solutions. They are presented in the followings:

**Post-query processing**

The importance of query suggestion is recognized by major Web search engines, who already incorporated this tool into their portal. To this end, specific post-query processing techniques are important for providing the user with additional (or enhanced) information: query suggestion is one of such kind. The goal of query suggestion is to provide the user with a list of related queries, in addition to the usual result list.

Query suggestion helps the user to either better specify the information he/she is looking for, or help him in browsing semantically related concepts. For instance, given the query "Pablo Picasso", possible interesting suggestions are "Pablo Picasso blu's period" (specification) or "cubism" (related concept - diversification).

We aim at producing a novel software product, specifically tailored for the information indexed by Europeana, and its users. In particular, we plan to exploit sources of implicit feedback from Europeana users, e.g. query logs.

**Metadata Based Ranking**

The goal of this service is to provide a better ranking of the result list returned after a user query.

The full-text based retrieval/ranking does not best fit the Europeana content representation,



and for that reason, a ranking model being aware of the structure present in Europeana documents is mandatory.

In this perspective, we aim at producing a novel software product, specifically tailored for the information indexed by Europeana, and its users. In particular, we plan to exploit sources of implicit feedback from Europeana users, e.g. query logs. From this knowledge it is possible to properly weigh the various fields of the Europeana Metadata Records.

This service will affect the result list presented on the Europeana portal. The service needs historical information regarding users interaction with Europeana, and in particular, query logs and click stream data should be made accessible. Also the service needs to access the full collection of metadata records for statistical analysis and feature extraction.

### ***Text Indexing and Retrieval***

The goal of this service is to provide efficient access specifically tailored to structured information exploited by those services. The most popular approach for text indexing is the usage of inverted document representation. We will adopt the same or similar technologies, and provide additional functionalities specifically tailored for Assets services. In particular, this service will focus on the analysis, the cleaning and the processing of the query logs. The output of the service will allow obtaining useful statistics on the user's usage and will feed the Post-query Processing and Metadata Based Ranking tasks.

### ***Images Indexing and Retrieval Service***

We aim at developing a scalable content based similarity search engine that will allow searching between Europeana images using an image as query. The goal is to offer more powerful yet easy to use search functions to our portal end-users. This type of search is also useful in a mobile context.

### ***3D-Model Indexing and Retrieval***

The 3D Model Indexing and Retrieval Services allow users to search for 3D models similar to a query 3D model. The 3D search interface will allow 3 types of queries: Models uploaded from the user, models returned from a previous search and hand-drawn sketches.

### ***Audio Indexing and Retrieval***

The audio indexing and retrieval service provides advanced music search and recommendation functionalities. It enhances Europeana by enabling end users to discover audio media based not only in textual similarity but based on audio content and context similarities. This service allows searching for similar songs, albums or artists based on a given song, album or artist; also being able to find tracks by mood, tempo amongst other music descriptors.

### ***Video summarization, indexing and retrieval***

Existing internet video providers don't provide neither advanced video summarization nor content based search (only text/tag based search). This service aims at enhancing the functionalities of Europeana for searching, browsing, pre-visualizing and accessing video content. The search of video content is many times difficult due to the high amount of information this kind of media contains.

- Provide the users with reduced-length versions of the original videos (video summaries) in order to be able to overview the original video content without



downloading/watching the complete original video. The personalization of video summaries will be possible in terms of summary length as well as tempo vs. coverage of the generated summaries which can be modified via generation parameters

- Offer solutions to allow the user to search for videos with enhanced mechanisms (e.g. visual similarity) apart from existing tag-based search approaches.

### 3.3 Digital Preservation Services

#### 3.3.1 The ASSETS Approach and Proposal

OAIS (Open Archival Information System) Reference Model is the ISO standard (14721:2003) adopted for addressing the digital preservation problem, which identifies three main components: i) information model, ii) archive responsibilities, iii) functional model.

For addressing the digital preservation issue it's important to:

- Identify and characterise events potentially impacting the long-term usability and access of digital information - This deals with the evaluation of potential accessibility RISKS;
- Monitor occurring events which potentially impact the long-term usability and access of digital information;
- Identify, characterise and evaluate relative corrective actions/plans for mitigating impacts (preservation plans) - This deals with the plans for reducing impacts of risks;
- Communicate/Notify impacting events and actions to the right actors involved in the preservation process - This deals with the notification/communication of RISKS to the actors responsible for taking the right decisions and enacting corrective actions;
- Enact, monitor and control relative corrective actions - This deals with the enactment/reaction for mitigating impacts of risks;
- Track, report and document occurring events/actions/changes within the digital archive, as evidence for judging/auditing/certifying the quality of "preservation archive" and the "authenticity/provenance/integrity" of archival objects - This deals with the reporting aspect, which is useful for tracking changes.

OAIS Preservation Planning and Administration components provide the functionality described above, as shown in the following table.

OAIS Responsibilities	
PLANNING	ADMINISTRATION
<ul style="list-style-type: none"> <li>• <b>Monitoring OAIS Environment;</b></li> <li>• <b>Detect changes/impacts in DCKB (Designated Community Knowledge Base);</b></li> <li>• <b>Mapping out Preservation Strategy;</b></li> <li>• <b>Provide Recommendations.</b></li> </ul>	<ul style="list-style-type: none"> <li>• <b>Manage submission agreements;</b></li> <li>• <b>Audit submission;</b></li> <li>• <b>Maintain configuration management;</b></li> <li>• <b>Monitor archive operations;</b></li> <li>• <b>Inventory archive content;</b></li> <li>• <b>Report on archive content;</b></li> <li>• <b>Migrate/update archive content;</b></li> </ul>

	<ul style="list-style-type: none"> <li>• Manage archive standards/policies.</li> </ul>
--	--

The ASSETS proposal for the digital preservation issue is to provide implementations of services with functionality defined for the OAIS Preservation Planning and the OAIS Administration components. More concrete, the ASSETS Digital Preservation Services will:

- Estimate preservation risk for data through the Risk Management Service
- Track and report/notify occurring events through the Notification Service
- Enact preservation plans through the Normalisation Service



Figure 12 - The ASSETS Digital Preservation Services

### **Risk Management**

The services aims at mitigating the risk of digital obsolescence by providing risk management reports to content providers - which is done by analyzing the contributed content. The service will perform object inspection and statistical analysis of the content formats at hand to categorize them based on their preservation risk. The service needs to be robust in terms of reliably classifying the underlying data on basis of available metadata and giving solid preservation recommendations. The component beyond this addresses the topics of technology watch and (semi)automated preservation policies. It will make use of available preservation community resources such as technical registries (like PRONOM<sup>1</sup> and

<sup>1</sup> <http://www.nationalarchives.gov.uk/PRONOM/Default.aspx>



UDFR<sup>2</sup> – Unified Digital Formats Registry) for policy extraction and the Assets Normalisation Service for object identification and policy execution.

The service addresses the problem of: format obsolescence and limited support for proprietary formats; unstructured and unknown digital collections; automated collection profiling and recommendation of fine-tuned preservation actions.

### ***Normalisation Service***

The Assets Normalization Preservation Service offers a wide range of tools (e.g. Droid, JHove, ImageMagick, etc.) and is responsible for deploying and exposing their functionality through a well defined set of standardized preservation operations. These include for example the identification, characterisation, migration and validation of digital objects. The provided API will allow for example to migrate a digital object from its original representation into open and preservation-friendly archival formats, such as e.g. PDF/A for documents or TIFF for images - or to profile digital collections. Based on the results of the risk management analysis, the service can automatically perform a normalization strategy on the provider's collection.

### ***Notification Service***

Notification is one of the digital preservation services, which guarantees the adequate communication and management procedures in reaction to events that could impact long term preservation, within the digital archive. As actions, appropriate messages are dispatched according to event types, well defined rules, roles of the entities involved in the digital environment (i.e. curator, preserver, holder). The alerted entity (i.e. human actor and/or automatic tool) is able to enact corrective action according to established preservation plans.

## **3.4 The Browsing and Content Characterisation Services**

### ***3.4.1 The ASSETS Approach and Proposal***

Most existing Information Retrieval systems either annotate all the objects in the database (manual tagging) or annotate a subset of the database manually selected (partial annotation). In the case of large-scale digital repositories, full annotation is increasingly difficult because of the manual effort required. Partial annotation is relatively affordable and trims down the heavy manual labour. Leveraging this information to automatically tag items that have not been tagged, e.g. newly entered items can be done effectively using recent advances in active learning frameworks, where most salient features and items are automatically selected for human verification and generalisation.

A dynamic and adaptive knowledge representation scheme - e.g. a vector space model or a probabilistic model - used for representing the semantics of collection items can be designed to allow generalisation and fuzzy matching of items within the collection and across collections. Together with a learning approach, this will allow for the construction of meaningful representations, which evolve over time and automatically assign associative links between items that are related by virtue of their usage (recommendation systems) or their content representations. Technologies that make this feasible have been deployed for

---

<sup>2</sup> <http://www.udfr.org/>

innovative text search systems but have not yet been put in place for multimedia, except in laboratory experiments - the technology is now mature for application.

The ASSETS project will adopt the Semantic Web paradigm in representing knowledge and content in order to achieve a semantically rich representation of content, user interests, ideas and social interactions. The project will augment the capabilities of multimedia information retrieval by providing the mechanisms for the establishment of semantic links between pieces of information presented in different media. This will be realised through the development of a service based on semantic cross-linking of multimedia content. The service will allow users who have little experience with the collection to gain access to it with a low learning threshold. This is expected to significantly improve searching, browsing and retrieving information, especially for large scale Digital Libraries such as Europeana.

The ASSETS project will also develop graphical user interfaces specially designed for searching and browsing in large scale multimedia archives. The objective is to develop an interface that provides intelligent access to a wide range of media content within an rich functionality web portal. This will support the processing of different types of queries, which enhances the user's experience and allows a customization of search responses according to user's information needs and preferences . Rich user interface components will be designed and implemented to support the ASSETS Platform and its value-added services. The ASSETS web portal will allow complex query formulation and results presentation, ensuring native integration and interoperability with EUROPEANA. It will also implement GUI components specially designed for improving the usability of the searching and browsing functionality of EUROPEANA platform.

The ASSETS proposal for the browsing and characterisation issue is enhance the application with functionality supporting Semantic cross-linking, the Query Log Analysis and the Graphical User Interface improvements for displaying a wide range a multimedia objects (e.g. different text, image, audio, video, and 3D formats). In particular, the ASSETS Browsing and Characterisation Services will:

- establish semantic links between pieces of information presented in different media, through the Annotation Propagation Service;
- appropriately train the Annotation Propagation Service, through the Training Service;
- allow to manually correct and complete the ingested metadata and provide training examples to the Training Service, through the Manual Annotation and Annotation Correction Service;
- select the most "informative" examples for manual annotation through the Content Selection Service;
- display and play media content provided by Assets content providers
- exploit user feedback information in order to improve the quality of search results, through the Relevance Feedback Service, and
- perform query log analysis through the Query Log Analysis Service.

The **assets** Services  
Browsing & Characterisation: what and how?

- Adoption of **Semantic Web paradigm**: establish semantic links between pieces of information presented in different media
- Exploit **user feedback information** in order to improve the quality of search results
- Browsing **Similarity and 3D**

content selection

annotation

manual

correction

propagation

query log analysis

relevance feedback

Figure 13 - The ASSETS Browsing and Characterisation Services

### 3.5 The Community Services

#### 3.5.1 The ASSETS Approach and Proposal

The objective of this WP is to develop services addressing the needs of digital libraries communities. A community is a group of users who share some interests, such as the carrying out of a common activity or the membership to some social network. This WP will focus on the following services:

- Content creation by re-use. The availability of content in digital form makes it possible to create new content by extracting and recombining in various forms existing digital objects or parts thereof. This service aims at supporting the composition of new complex objects by combining the information available in simpler digital objects;
- Taxonomy-based notification. This is a service that enables the selective dissemination of events, in an asynchronous manner, to communities that have expressed interest in those events. This service aims at developing a sophisticated form of notification, based on a taxonomy between the events;
- Personalization services. This service aims at customizing the functionalities of a digital library to specific communities by reacting in different ways to one and the same request, depending on the community issuing the request.

The **assets** Services  
Community: what and how?

- Community services aimed at supporting this **information exchange**
- **Composition of new content** by extracting and recombining existing digital objects
- **Selective dissemination** of events to communities that have expressed interest in those
- **Personalisation** of search results, based on the specific user/community preferences.

**content creation by re-use**

**taxonomy-based notification**

**personalisation**

Figure 14 - The ASSETS Community Services

### 3.6 User Interface of the ASSETS Portal

#### Premise

Users evaluating the Europeana portal have reported that the look & feel of the portal is good, although the purpose of the web site could be unclear to users at first. They have also demanded the ability to get more clear and precise information about specific items. Some functionality and usability drawbacks could discourage users from future use of Europeana. Also, some difficulties experienced in navigation, query creation and using filters to refine searches indicates that the Europeana Portal should to be improved by offering access to its functionality in a more user friendly environment.

#### 3.6.1 The ASSETS Approach and Proposal

The ASSETS proposal for the user interface improvement issue supports the enhancements of GUIs with components specially designed for searching and browsing large scale multimedia archives.

Rich user interface components will be designed and implemented to support the ASSETS Platform and its value-added services. We aim at providing user friendly and intuitive interaction with the Europeana platform and its contents. Consequently, this will make the portal more attractive and will increase its popularity for the wide public. Particularly, the ASSETS UI Improvements will provide support for the end-user services such as:

- Scalable metadata and Similarity based Indexing and Retrieval;



- Semantic cross linking browsing;
- Media display, play and preview;
- Intuitive execution of text and content based similarity search actions;

A complete release of User Interface improvements is expected due to next months, but yet we are able to present some prototypes with preliminary results and mock-ups, which were developed during this year. Those are shown for better describing and clarifying how the ASSETS services are/will be used for improving access and usability.

User Interface Name	<b>Query Suggestion UI</b>
Objective	Help the user to either better specify the information s/he is looking for, or help him in browsing semantically related concepts
Basics	<ul style="list-style-type: none"> <li>• A list of related links will be shown below the search box and/or at the bottom of the result search page.</li> <li>• This list should be easily recognizable but not intrusive</li> </ul>
Dependencies	Query Suggestion Service
Mock-ups	
Next figure shows the suggestions returned to the user after submitting a query.	

pablo picasso









[Refine Search](#) [Advanced search](#)

**Related seaches:**  
[pablo picasso obras](#) [pablo picasso cuadros](#) [escuela pablo picasso](#)

Matches for: pablo picasso

All **Texts (22)** Images (9,518) Videos (35) Sounds (5)

Results 1 - 12 of 9,580 Page: 1 2 3 4 5 6 7 8 9 10 →

 <p><b>Pablo Picasso</b> Deutsche Fotothek</p>	 <p><b>Pablo Picasso</b> Deutsche Fotothek</p>	 <p><b>Pablo Picasso</b> Deutsche Fotothek</p>	 <p><b>Pablo Picasso</b> Deutsche Fotothek</p>
 <p><b>Pablo Picasso</b> Deutsche Fotothek</p>	 <p><b>Pablo Picasso</b> Deutsche Fotothek</p>	 <p><b>Pablo Picasso</b> Deutsche Fotothek</p>	 <p><b>Pablo Picasso</b> Anonyme , Culture.fr/collections</p>

Results 1 - 12 of 9,580 Page: 1 2 3 4 5 6 7 8 9 10 →

**Related seaches:**  
[pablo picasso obras](#) [pablo picasso cuadros](#) [escuela pablo picasso](#)

When the user clicks the "arrow" button, further suggestions will be returned:

pablo picasso

[Refine Search](#) [Advanced search](#)

**Related seaches:**

[pablo picasso obras](#)  
 [pablo picasso cuadros](#)  
 [pablo picasso obras](#)  
 [pablo picasso cuadros](#)  
 [escuela pablo picasso](#)  
 [pablo picasso obras](#)  
 [pablo picasso cuadros](#)

Matches for: pablo picasso

Results 1 - 12 of 9,580   Page: 1 2 3 4 5 6 7 8 9 10 →

Results 1 - 12 of 9,580   Page: 1 2 3 4 5 6 7 8 9 10 →

**Related seaches:**

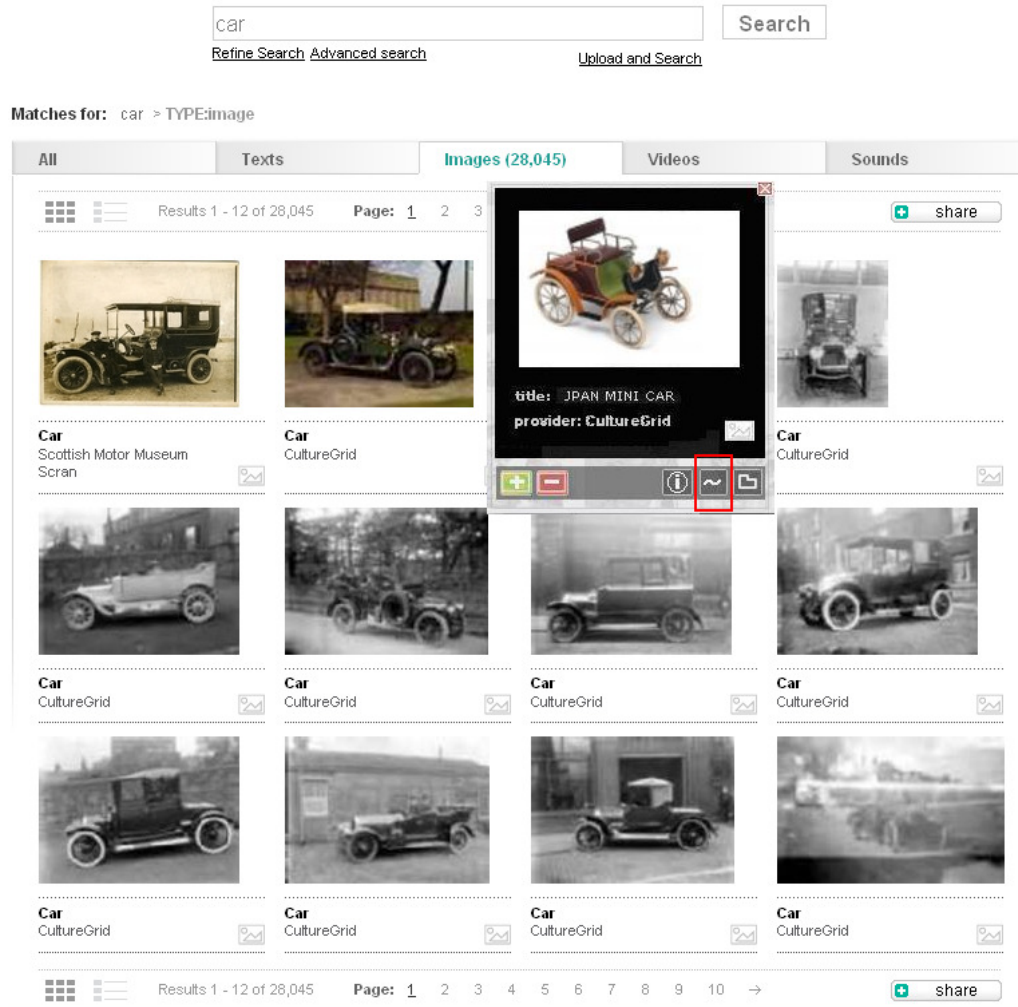
[pablo picasso obras](#)  
 [pablo picasso cuadros](#)  
 [escuela pablo picasso](#)

User Interface Name	<b>Similar Search Images UI</b>
Objective	Search for visually similar images, allowing a user to find images based on other images.
Basics	<ul style="list-style-type: none"> <li>• A link for searching similar images should be added near each image returned by the Europeana, independently from the type of query. In particular, the button or link for similarity searching should be showed only for those images for which it will possible to perform the content based similarity search.</li> <li>• The UI should allow similarity search in a self-explanatory and easy way. No settings, weights or specifications of what is the intention of the user searching for similar images should be required.</li> <li>• In the results list of similar images it could be added a text reporting the similarity score of the results images. However, this is not mandatory.</li> </ul>



Mock-ups

When the mouse is over an image thumbnail a mini-zoom window will be displayed. The toolbar at the bottom of this window includes a button labelled "~" that allows the user to make a similar search based on the selected image.



When the user press the "~" button the system returns as result the list of similar images. The image used for the query is displayed at the top of the results.




Search

[Refine Search](#)
[Advanced search](#)
[Upload and Search](#)


Matches for:

All
Texts
Images (28,045)
Videos
Sounds


Results 1 - 12 of 28,045
Page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) →
+ share




**Car**  
Scottish Motor Museum  
Scran




**Car**  
CultureGrid




**Baby Carriage**  
CultureGrid




**Argyll Tourer Type G Motor**  
CultureGrid




**Car**  
CultureGrid




**Model of a Two-Wheel**  
CultureGrid




**Model of a Two-Wheel**  
CultureGrid




**Spinal Carriage**  
CultureGrid




**Horse drawn ralli car, by A.D.**  
CultureGrid



**Arrol-Johnston '18' Open**  
CultureGrid



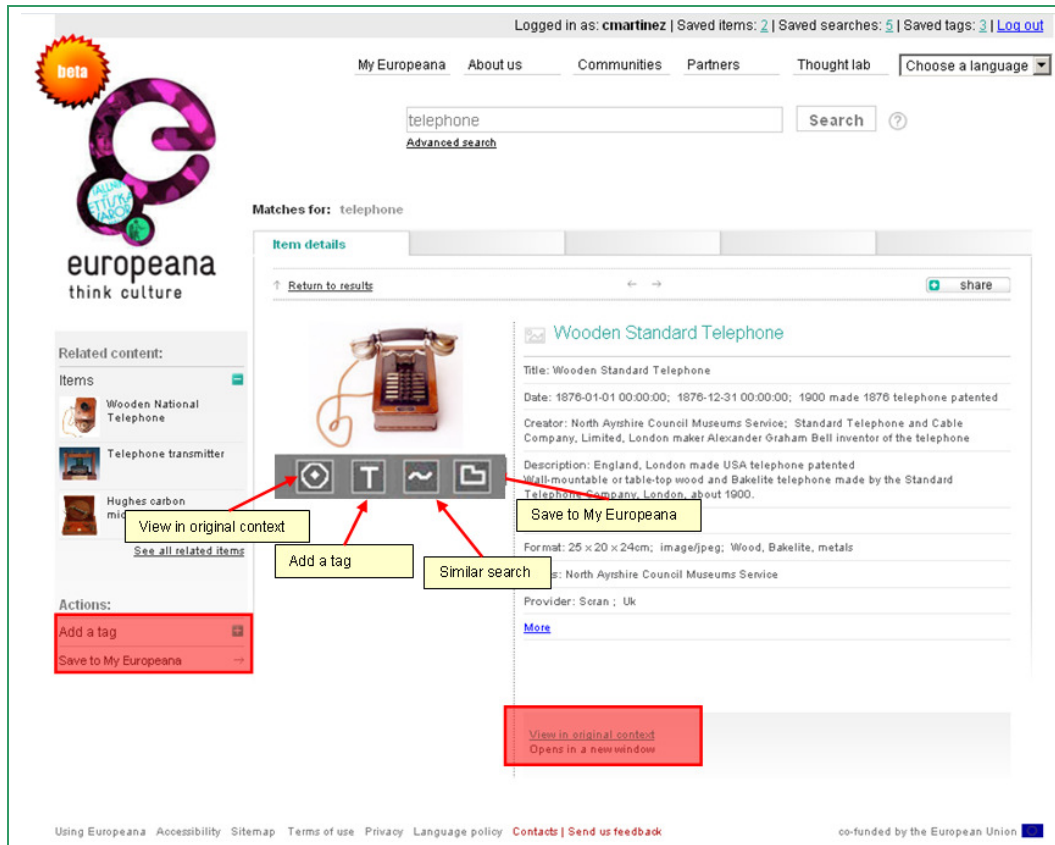
**Car**  
CultureGrid



**Car**  
CultureGrid

Results 1 - 12 of 28,045
Page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) →
+ share

A similar search can also be launched in the Details window, by clicking the "~" button at the toolbar.




User Interface Name	<b>Uploading &amp; Search Images UI</b>
Objective	Upload a locally available image or refer an image on the web, and make a search (e.g. to identify an image whose motif is unknown to the user)
Basics	<ul style="list-style-type: none"> <li>• Identity an image to base a search by uploading a locally available image or referring an image on the web.</li> <li>• A link should be created just below the text box for searching. This link will allow uploading an image for searching on the basis of the content of the given image</li> </ul>
Dependencies	Images Indexing and Retrieval Service
Mock-ups	A link labelled "Upload and Search" will be placed below the search box.

[Refine Search](#)
[Advanced search](#)
Upload and Search


Matches for: car > TYPE:image

All
Texts
Images (28,045)
Videos
Sounds


Results 1 - 12 of 28,045
Page: 1
2
3
4
5
6
7
8
9
10
→
[share](#)




**Car**  
Scottish Motor Museum  
Scran




**Car**  
CultureGrid




**Car**  
CultureGrid




**Car**  
CultureGrid




**Car**  
CultureGrid




**Car**  
CultureGrid




**Car**  
CultureGrid




**Car**  
CultureGrid




**Car**  
CultureGrid



**Car**  
CultureGrid



**Car**  
CultureGrid



**Car**  
CultureGrid

Results 1 - 12 of 28,045
Page: 1
2
3
4
5
6
7
8
9
10
→
[share](#)

When the user clicks this link a window will be opened where the user can either select a file or enter a URL address. After clicking the "Search" button, the image is uploaded and the search similar function returns a list of similar images.

[Hide upload and search](#) ✕

**Select the image to upload**

**File:**

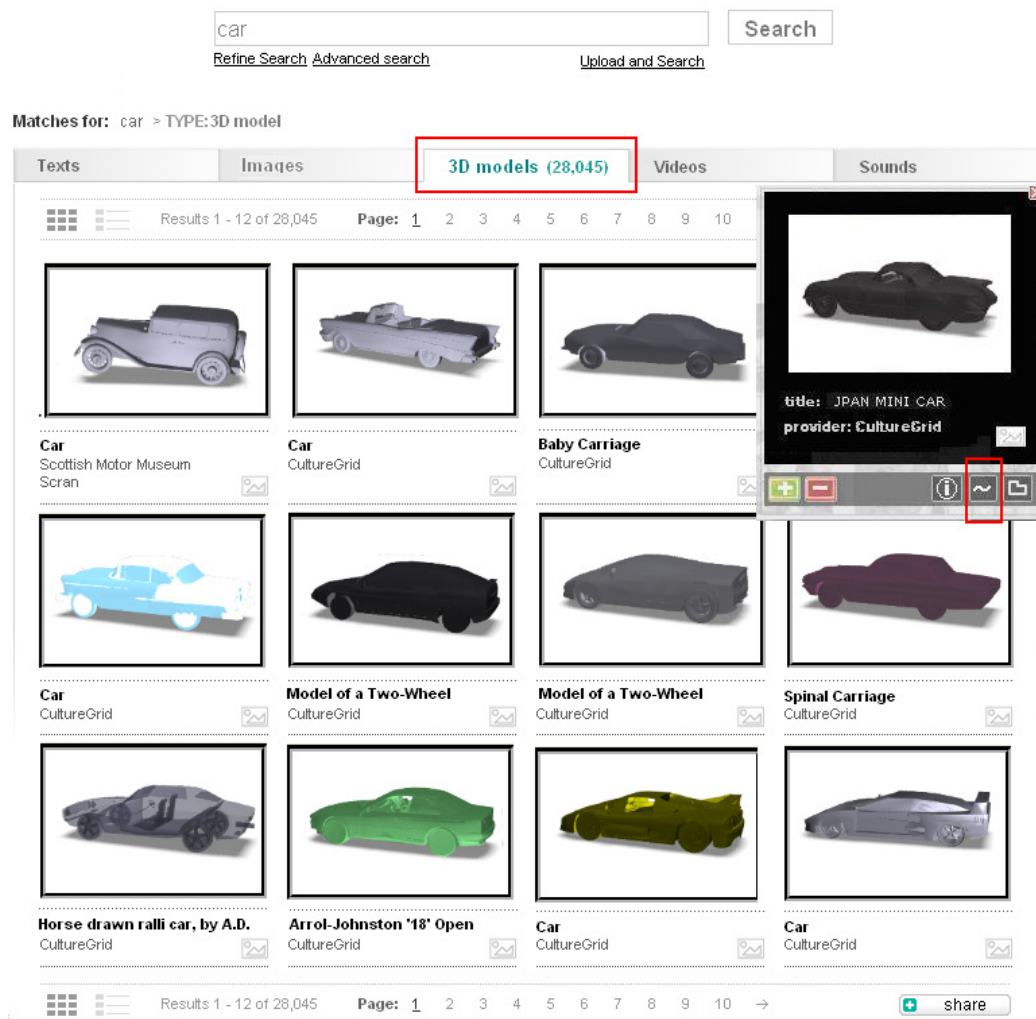
**url:**

Search

User Interface Name	<b>Similar Search 3D UI</b>
Objective	Select one of the existing 3D models and search for similar models.
Basics	<ul style="list-style-type: none"> <li>The user accesses the interface by clicking on a "Search similar" link, while browsing through the existing models. The search similar function returns a list of similar results.</li> <li>The service should be self explanatory and easy to use even from a non-expert user.</li> </ul>
Dependencies	3D Search and Retrieval service

**Mock-ups**

When the mouse is over a 3D model thumbnail a mini-zoom window will be displayed. The toolbar at the bottom of this window includes a button labelled "~" that allows the user to make a similar search based on the selected 3D model.



When the user press the "~" button the system returns as result the list of similar models. The model used for the query is displayed at the top of the results.

[Refine Search](#) [Advanced search](#) [Upload and Search](#)

Matches for:

[Texts](#) [Images](#) [3D models \(28,045\)](#) [Videos](#) [Sounds](#)

Results 1 - 12 of 28,045 Page: 1 2 3 4 5 6 7 8 9 10 →

<b>Car</b> Scottish Motor Museum Scran	<b>Car</b> CultureGrid	<b>Baby Carriage</b> CultureGrid	<b>Argyll Tourer Type G Motor</b> CultureGrid
<b>Car</b> CultureGrid	<b>Model of a Two-Wheel</b> CultureGrid	<b>Model of a Two-Wheel</b> CultureGrid	<b>Spinal Carriage</b> CultureGrid
<b>Horse drawn ralli car, by A.D.</b> CultureGrid	<b>Arrol-Johnston '18' Open</b> CultureGrid	<b>Car</b> CultureGrid	<b>Car</b> CultureGrid

Results 1 - 12 of 28,045 Page: 1 2 3 4 5 6 7 8 9 10 →

A similar search can also be launched in the Details window, by clicking the "~" button at the toolbar.

User Interface Name	<b>Upload &amp; Draw &amp; Search 3D UI</b>
Objective	Upload a 2D model or create a sketch and search for similar models
Basics	<ul style="list-style-type: none"> <li>The user accesses the interface by selecting the appropriate tab. He or she either draws a sketch or uploads a 2D image by clicking a "Browse" button and sketches the contour of the object of interest. The search similar function returns a list of similar results.</li> </ul>
Dependencies	3D Search and Retrieval service
Mock-ups	
A link labelled "Upload and Search" will be placed below the search box.	

car

[Refine Search](#) [Advanced search](#) [Upload and Search](#)

Matches for: car > TYPE:3D model

[Texts](#)
[Images](#)
[3D models \(28,045\)](#)
[Videos](#)
[Sounds](#)

Results 1 - 12 of 28,045 Page: 1 2 3 4 5 6 7 8 9 10 → [share](#)

<b>Car</b> Scottish Motor Museum Scran	<b>Car</b> CultureGrid	<b>Baby Carriage</b> CultureGrid	<b>Argyll Tourer Type G Motor</b> CultureGrid
<b>Car</b> CultureGrid	<b>Model of a Two-Wheel</b> CultureGrid	<b>Model of a Two-Wheel</b> CultureGrid	<b>Spinal Carriage</b> CultureGrid
<b>Horse drawn ralli car, by A.D.</b> CultureGrid	<b>Arrol-Johnston '18' Open</b> CultureGrid	<b>Car</b> CultureGrid	<b>Car</b> CultureGrid

Results 1 - 12 of 28,045 Page: 1 2 3 4 5 6 7 8 9 10 → [share](#)

When the user clicks this link a window will be opened where the user can select the "Draw" option. The window will then expand to allow the user either to select a base image (and draw a contour) or to make a sketch from scratch. After clicking the "Search" button, the model is uploaded and the search similar function returns a list of similar models.


[Hide upload and search](#) ✕

**Select the 3D model to upload**

File:

Draw

Base image:



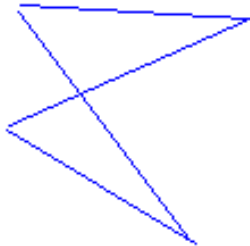
[Hide upload and search](#) ✕

**Select the 3D model to upload**

File:

Draw

Base image:

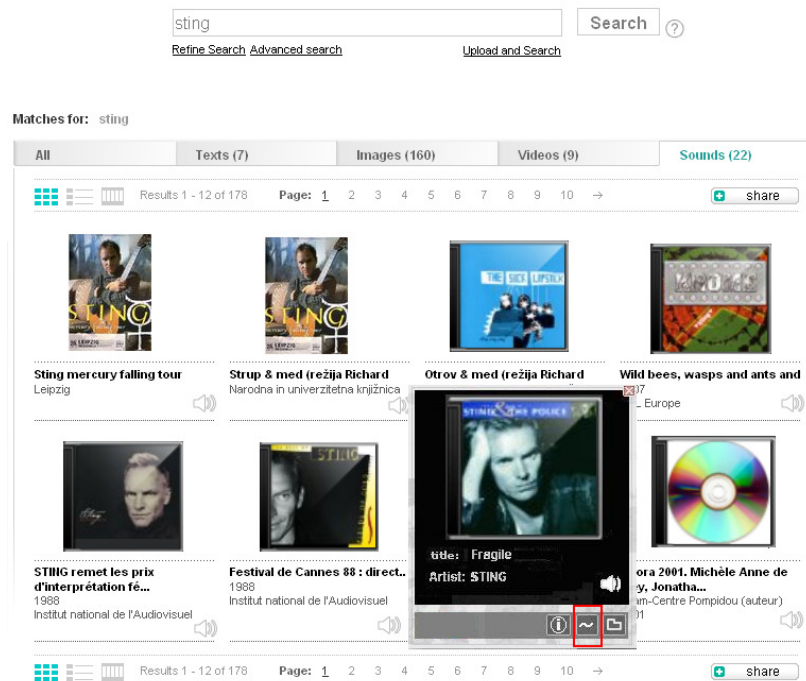


User Interface Name	<b>Similar Search Audio UI</b>
Objective	Similarity search by prior search result: allowing searching for similar songs, albums or artists based on a given song, album or artist
Basics	<ul style="list-style-type: none"> <li>A link for searching similar audio content should be added near each audio content returned by Europeana in the web user interface independently from the type of query performed. In particular, the button or link for similarity searching should be showed only for those audio contents for which it will possible to perform the similarity search.</li> <li>In the results list of similar songs it could be added a text reporting the similarity score of the results songs. However, this is not mandatory</li> </ul>
Dependencies	Audio Indexing and Retrieval service

**Mock-ups**

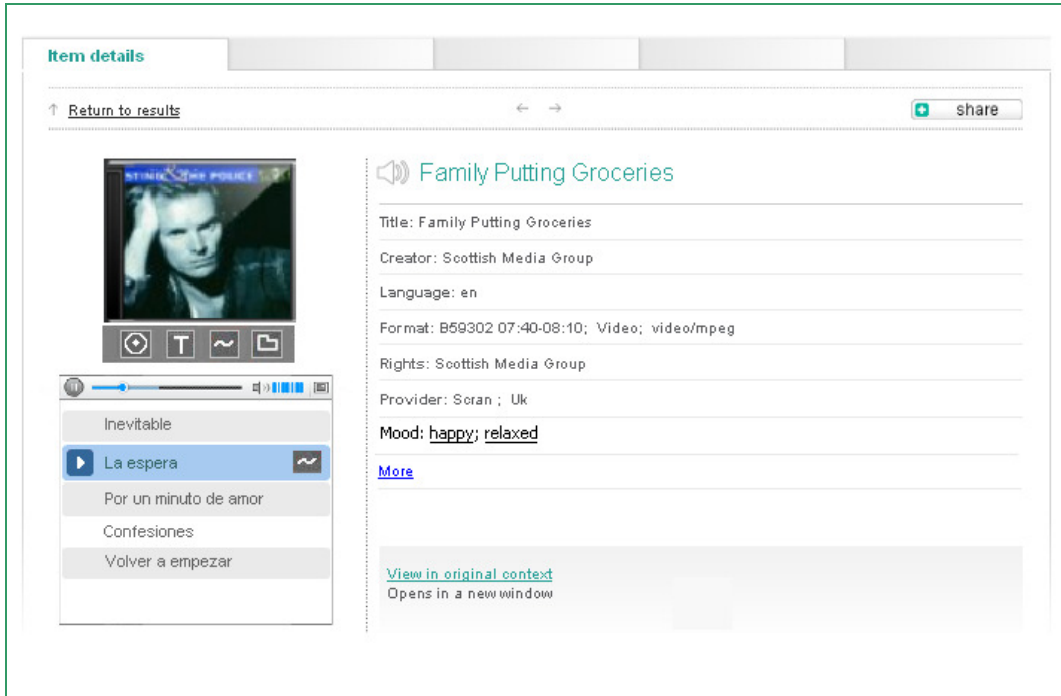
When the mouse is over an audio thumbnail a mini-zoom window will be displayed and the track will start to play automatically. The toolbar at the bottom of this window includes a button labelled "~" that allows the user to make a similar search based on the selected music.

When the user press the "~" button the system returns the list of similar tracks.

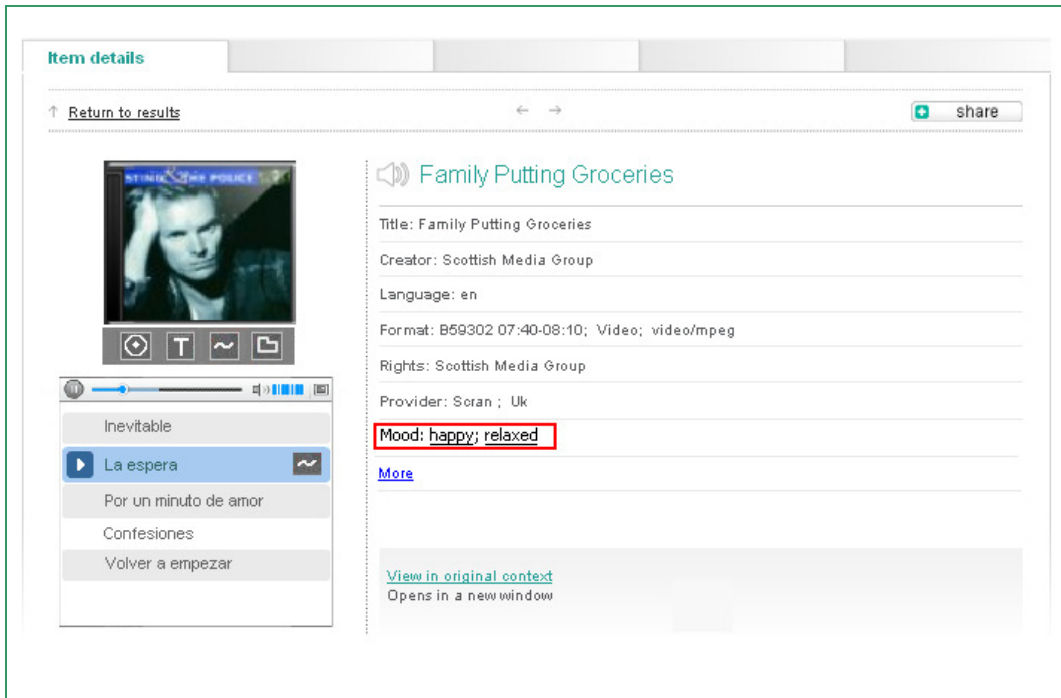


A similar search can also be launched in the Details window, by clicking the "~" button at the toolbar. When the item is an album the details window will show the list of tracks within the album. The user will be able to play a track or to make a similar search based on it.





User Interface Name	Descriptor Search Audio UI
Objective	Find tracks by mood and other music descriptors
Basics	<ul style="list-style-type: none"> <li>A user searching for classical music in Europeana finds a very sad and powerful piece of music. He then clicks the link "Find more sad music" and Europeana returns audio objects that have been indexed and tagged as "sad"</li> </ul>
Dependencies	Audio Indexing and Retrieval service
Mock-ups	<p>When the user opens the <b>Details window</b> of a music item, its music descriptors tags will be shown on the right hand side of the window. The user can click on any of them to launch a new search based on that value.</p>



User Interface Name	Similar Search Video UI
Objective	Search for visually similar videos, i.e. provide solutions to allow the user to carry out searches of video content based in extracted video features (e.g. allow the user to search for videos containing similar video shots)
Basics	<ul style="list-style-type: none"> <li>A link for searching similar video content should be added for video objects displayed in Europeana portal.</li> </ul>
Dependencies	Video Indexing and Retrieval service
Mock-ups	<p>When the mouse is over a video thumbnail a mini-zoom window will be displayed. The toolbar at the bottom of this window includes a button labelled "~" that allows the user to make a similar search based on the selected video.</p> <p>When the user presses the "~" button, the system will performs a content based search and will return a list of similar videos.</p>

car  Search ?

[Refine Search](#) [Advanced search](#) [Upload and Search](#)

Matches for: car > TYPE:VIDEO

All Texts Images **Videos (804)** Sounds

Results 1 - 12 of 804 Page: 1 2 3 4 5 6 7 8 9 10 → [share](#)

**Championnat d'Europe de stock cars..**  
1957  
Institut national de l'Audiovisuel

**Cars interdits Montmartre..**  
1997  
Institut national de l'Audiovisuel

**A Car Journey: Educational film (video c...**  
Scottish Screen

**Board game**  
National Museums of Scotland - Scottish Life Archive  
Scran

**Accès à Montmartre interdit aux cars de ...**  
1985  
Institut national de l'Audiovisuel

**Man in Car with Mobile Phone and Laptop ...**  
Scottish Media Group  
Scran

**Title: Family Groceries**  
Artist: STING

**Asian Women Learning Car Maintenance (si...**  
Scottish Media Group  
Scran

Results 1 - 12 of 804 Page: 1 2 3 4 5 6 7 8 9 10 → [share](#)

A similar search can also be launched in the **Details window**, by clicking the "~" button at the toolbar. In this window the Storyboard of the video will be shown. The user will be able to perform a similar search for a selected keyframe.

**Item details**

[Return to results](#) [share](#)

Video player controls: S, play, T, ~, share

Storyboard: 00:00, 00:10, 00:20, 00:30, 00:40, 00:50, 01:00

**Family Putting Groceries in Car (silent video clip)**

Title: Family Putting Groceries in Car (silent video clip)

Creator: Scottish Media Group

Language: en

Format: B59302 07:40-08:10; Video; video/mpeg

Rights: Scottish Media Group

Provider: Scran ; Uk

[More](#)

[View in original context](#)  
Opens in a new window

User Interface Name	<b>Video Summarization UI</b>
Objective	Provide condensed versions of complete videos (video summaries or video abstracts) for easing the users' browsing process
Basics	<ul style="list-style-type: none"> <li>Video summaries will allow the user to overview the original video content without downloading/watching the complete original video</li> </ul>
Dependencies	Video Indexing and Retrieval service; Video Summarization service

**Mock-ups**

Video summaries will be displayed on the **Mini-zoom window** and on the **Details window**.  
 A button toolbar labelled "S" will allow the user to start this functionality on the selected video.

The screenshot shows a search interface for the term "car". At the top, there is a search bar containing "car", a "Search" button, and a help icon. Below the search bar are links for "Refine Search", "Advanced search", and "Upload and Search".

The results section is titled "Matches for: car > TYPE:VIDEO". It features a navigation bar with tabs for "All", "Texts", "Images", "Videos (804)", and "Sounds". The "Videos (804)" tab is selected.

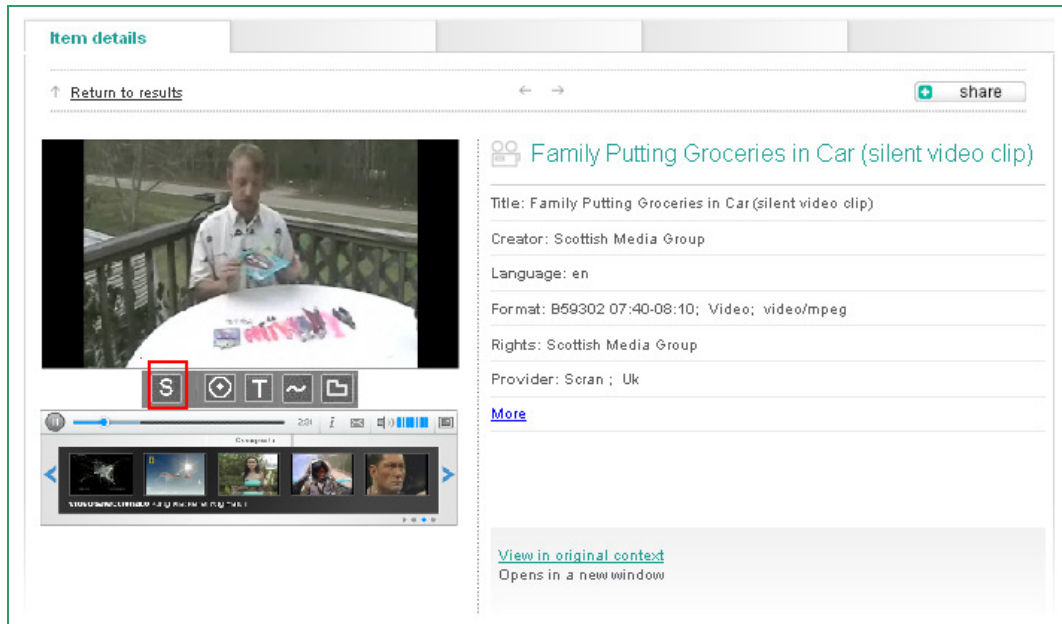
Below the navigation bar, there are pagination controls showing "Results 1 - 12 of 804" and "Page: 1" with a right arrow. A "share" button is also present.

The main content area displays a grid of video thumbnails. Each thumbnail includes a video player preview, a title, a year, and the source. The thumbnails shown are:

- Championnat d'Europe de stock cars..** (1957, Institut national de l'Audiovisuel)
- Cars interdits Montmartre..** (1997, Institut national de l'Audiovisuel)
- A Car Journey: Educational film (video c...** (Scottish Screen)
- Board game** (National Museums of Scotland - Scottish Life Archive Scran)
- Accès à Montmartre interdit aux cars de ...** (1985, Institut national de l'Audiovisuel)
- Man in Car with Mobile Phone and Laptop ...** (Scottish Media Group Scran)
- Family Groceries** (Artist: STING)
- Asian Women Learning Car Maintenance (sl...** (Scottish Media Group Scran)

The video player for "Family Groceries" is highlighted with a red box. It features a yellow "Video Summary" button and a toolbar with an "S" button, which is also highlighted with a red box. Other icons in the toolbar include a magnifying glass, a refresh icon, and a share icon.

At the bottom of the results section, there are more pagination controls and a "share" button.





User Interface Name	Similar Category UI
Objective	Search for media of the same category as a selected media file (image, 3D model)
Basics	<ul style="list-style-type: none"> <li>The database media files are classified in several categories. The user accesses the functionality by clicking on the category of a selected media file. Results of the same category are returned.</li> <li>This service will work on 3D models and images, as those types of media use low-level feature vectors for their description.</li> </ul>
Dependencies	Relevance Feedback service
Mock-ups	<p>The user initiates a search, then selects one of the results and opens the <b>Details window</b>. A list of the object Semantic categories will be displayed on the right hand side of the window. Each category is a link.</p> <p>By accessing the a new search for objects from the selected category will be performed..</p>

Matches for: telephone

Item details

[Return to results](#) [share](#)

**Wooden Standard Telephone**

---

Title: Wooden Standard Telephone

Date: 1876-01-01 00:00:00; 1876-12-31 00:00:00; 1900 made 1876 telephone patented

Creator: North Ayrshire Council Museums Service; Standard Telephone and Cable Company, Limited, London maker Alexander Graham Bell inventor of the telephone

Description: England, London made USA telephone patented Wall-mountable or table-top wood and Bakelite telephone made by the Standard Telephone Company, London, about 1900.

Language: en

Format: 25 x 20 x 24cm; image/jpeg; Wood, Bakelite, metals

Rights: North Ayrshire Council Museums Service

Provider: Scran ; Uk

**Semantic Labels: telephone; square; communication**

[More](#)

---

[View in original context](#)  
Opens in a new window

User Interface Name	<b>Relevance Feedback UI</b>
Objective	Refine the results already retrieved during the last search session
Basics	<ul style="list-style-type: none"> <li>The user performs a search and inspects the results. The user marks some of the results as more relevant (or less relevant) than the rest to the object that s/he has in mind. This feedback is used to refine the search and return more similar results. Refined results are presented to the user. The refinement can be repeated until the user is satisfied.</li> </ul>
Dependencies	Relevance Feedback service
Mock-ups	
The user initiates a search and judges (gives scores to) the relevance of the retrieved results by clicking on the mini-zoom "+" "-" toolbar buttons.	

car

[Refine Search](#) [Advanced search](#)

Matches for: car > TYPE:image

All   Texts   **Images (28,045)**   Videos   Sounds

Results 1 - 12 of 28,045   Page: 1 2 3

**Car**  
Scottish Motor Museum  
Scran

**Car**  
CultureGrid

**Car**  
CultureGrid

**Car**  
CultureGrid

**Car**  
CultureGrid

**Car**  
CultureGrid

**Car**  
CultureGrid

**Car**  
CultureGrid

**Car**  
CultureGrid

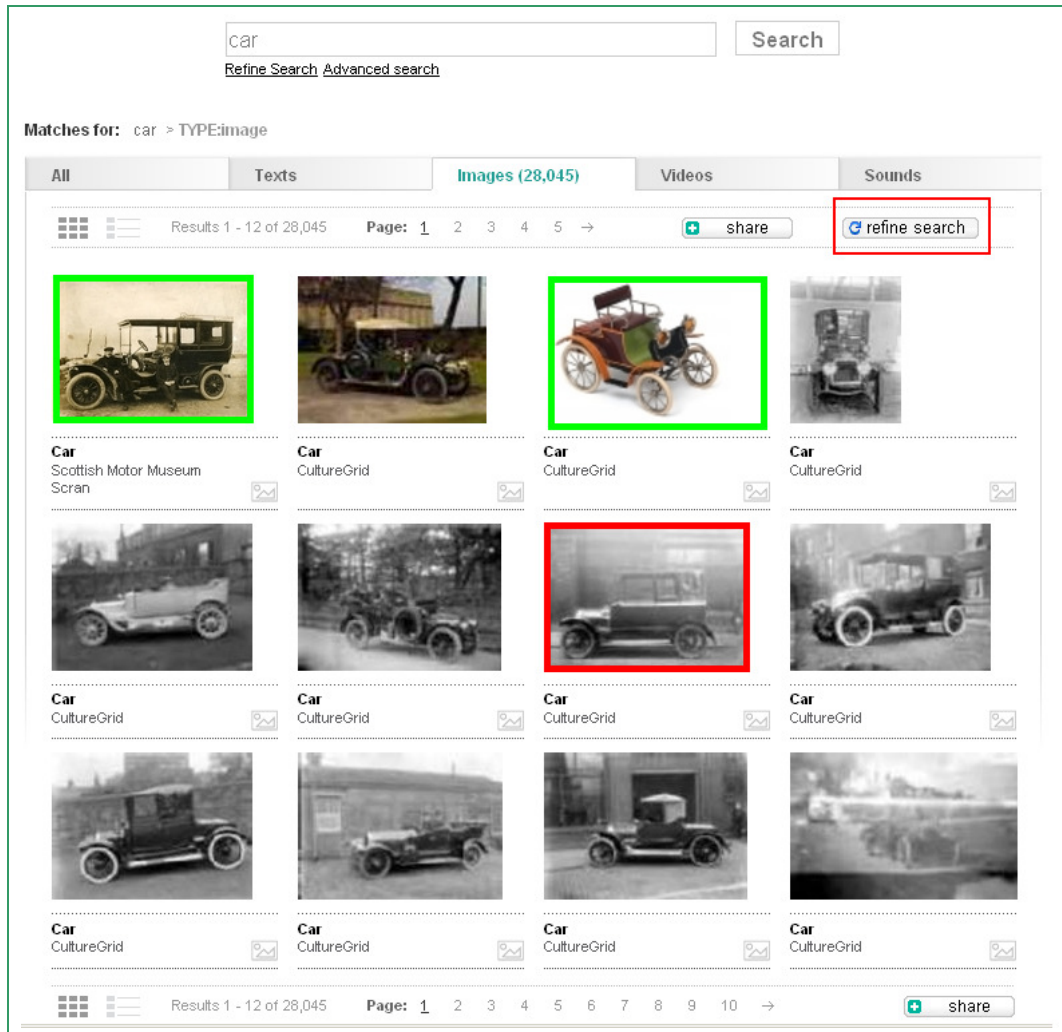
**Car**  
CultureGrid

**Car**  
CultureGrid

**Car**  
CultureGrid

Results 1 - 12 of 28,045   Page: 1 2 3 4 5 6 7 8 9 10 →

Those results that have been marked as "+" or "-" by the user will be shown on the interface surrounded by a "green" or "red" frame respectively. The "Refine" button will allow to launch a new search and receive new results which are more relevant to the object(s) that s/he is looking for.



User Interface Name	Improvements On Results_UI
Objective	Enhance the user experience in the interaction through the Europeana portal, providing search, navigation and access to documents in a user-friendly way
Basics	<ul style="list-style-type: none"> <li>• Option to sort the results by different criteria (e.g. by rank, by title, by creator, by provider, by date, etc.).</li> <li>• Option to define the desired number of results to display.</li> <li>• <b>Compact table view:</b> to display just the title below each thumbnail and the rest of the information on mouse-over.</li> <li>• <b>Mini-zoom window</b> displayed on mouse-over a result: Show a bigger size thumbnail -if available-, main metadata values and a toolbar.</li> <li>• <b>Buttons Toolbar:</b> icons to launch main operations on the selected result: "Save to My Europeana", "Similar search",</li> </ul>



“Open details”. This toolbar will be included in the Mini-zoom window and in the Details page.

Dependencies

Mock-ups

Next figure shows a mock-up of the proposed "Compact view" and the mini-zoom on a result.

The screenshot shows the Europeana search results page for the query "telephone". The page is in compact view, indicated by the grid view icon in the top left. A search bar at the top contains the text "telephone". Below the search bar, there are tabs for "All", "Texts (78)", "Images (1,493)", "Videos (73)", and "Sounds (11)". The "All" tab is selected. The search results are displayed in a grid. A red box highlights the grid view icon in the top left. Another red box highlights the mini-zoom toolbar over a search result titled "JAPAN MINI CAR" with provider "Cultura eGrid". The toolbar includes icons for zoom, pan, and other image manipulation functions. The page also features a sidebar with filters for "Refine your search" (By provider, language, country, date, type) and "Actions" (Save this search, Login, Register). At the bottom, there are links for "Using Europeana", "Accessibility", "Sitemap", "Terms of use", "Privacy", "Language policy", "Contacts", and "Send us feedback".

Next figures show mock-ups of the details window for images, audio and video respectively.

Logged in as: **cmartinez** | Saved items: 2 | Saved searches: 5 | Saved tags: 3 | [Log out](#)

My Europeana About us Communities Partners Thought lab Choose a language ▾

telephone   ?  
[Advanced search](#)

Matches for: telephone

**Related content:**

Items

- Wooden National Telephone
- Telephone transmitter
- Hughes carbon microphone


[See all related items](#)

**Actions:**

- Add a tag
- Save to My Europeana

**Item details**

[Return to results](#) [share](#)



**Wooden Standard Telephone**

Title: Wooden Standard Telephone

Date: 1876-01-01 00:00:00; 1876-12-31 00:00:00; 1900 made 1876 telephone patented

Creator: North Ayrshire Council Museums Service; Standard Telephone and Cable Company, Limited, London maker Alexander Graham Bell inventor of the telephone

Description: England, London made USA telephone patented Wall-mountable or table-top wood and Bakelite telephone made by the Standard Telephone Company, London, about 1900.

Format: 25 x 20 x 24cm; image/jpeg; Wood, Bakelite, metals

Provider: North Ayrshire Council Museums Service

Provider: Scran ; Uk

[More](#)

View in original context

Add a tag

Similar search

Save to My Europeana


View in original context  
Opens in a new window

Using Europeana [Accessibility](#) [Sitemap](#) [Terms of use](#) [Privacy](#) [Language policy](#) [Contacts](#) | [Send us feedback](#) co-funded by the European Union

---

**Item details**

[Return to results](#) [share](#)



**Family Putting Groceries**

Title: Family Putting Groceries

Creator: Scottish Media Group

Language: en

Format: B59302 07:40-08:10; Video; video/mpeg

Rights: Scottish Media Group

Provider: Scran ; Uk

**Mood: happy; relaxed**

[More](#)

View in original context  
Opens in a new window



**Item details**

[Return to results](#) [share](#)

**Family Putting Groceries in Car (silent video clip)**

Title: Family Putting Groceries in Car (silent video clip)

Creator: Scottish Media Group

Language: en

Format: B59302 07:40:08:10; Video: video/mpeg

Rights: Scottish Media Group

Provider: Soran ; Uk

[More](#)

[View in original context](#)  
Opens in a new window

<b>User Interface Name</b>	<b>Improvements On MyEuropeana UI</b>
<b>Objective</b>	Introduce new facilities in 'My Europeana' page
<b>Basics</b>	<ul style="list-style-type: none"> <li>Option to allow the user sorting stored items by different available criteria (e.g. by type, by title, by creator, by date,...).</li> <li>Option to delete all items at once.</li> <li>Mini-zoom window to be displayed on mouse-over.</li> </ul>
<b>Dependencies</b>	

**Mock-ups**

Next figure shows a mock-up of the proposed improvements for "My Europeana" page.

The screenshot shows the 'My Europeana' user interface. At the top, there is a navigation bar with 'My Europeana', 'About us', 'Communities', 'Partners', 'Thought lab', and a language selector. Below this is the 'My Europeana' header. A sub-header contains 'User information', 'Saved items', 'Saved searches', and 'Saved tags'. Under 'Saved items', there is a dropdown menu for 'Sort results by:' with 'Latest' selected, and a 'Delete all' button. The main content area displays a list of saved items, each with a thumbnail, title, creator, date saved, and a 'share' button. Two items are visible: 'Vas bitronconic cu două tor Åxi' and 'Bronzezeitliche Schleif- oder Karrenspuren (car ru...'. The footer contains various links like 'Using Europeana', 'Accessibility', 'Sitemap', 'Terms of use', 'Privacy', 'Language policy', 'Contacts', and 'Send us feedback', along with a note 'co-funded by the European Union'.



## 4. The ASSETS Data Models and Interfaces

The information available from the above sections allows us to depict the context and the application scenarios for the ASSETS services, and consequently, this allows us to identify concepts which are presented below. These concepts are modelled as UML diagrams which emphasize the relationships between and the properties of objects, which represent the ASSETS Data Models.

Obviously, each group of services (i.e. Ingestion, Indexing/Ranking/Retrieval, Digital Preservation, Browsing/Characterisation, Community) will model and use a specific set of concepts. By now, during the above sections, common concepts and needs have been identified. For that reason, this section introduces the Common Data Model, and afterwards it and describes the Service Specific Data Models.

The ASSETS Data Models drive the identification of Java Interfaces for the ASSETS Services, which are the basis for the implementation of the ASSETS Services API.

For that reason, in this document we present the most important aspects related to the assets service design such as: their interfaces, their responsibility, their supported operations and the key concepts modelled within the service.

The following tables have been used as template for collecting this information.

Service Name	<i>The name of the service</i>
Responsibility	<i>List of items for the responsibility of the service</i>
Provided Interfaces	<i>List of the interfaces through whom the service provides its features and manages key concepts</i>
Dependencies	<i>List of dependencies with other ASSETS services, if any. If this information is not available, provides the expected key concepts which represent inputs for the service from other ASSETS services</i>

Interface Name	<i>The name of the service interface</i>
Key Concepts	<i>Identification of the key concepts ( data model) managed by the interface</i>
Operations	<i>List of Item for the operations of the interface</i>

It is important to remark that ASSETS project is adopting an iterative and incremental development process. For that reason, the interfaces and models presented here are a picture of the current development phase, where many components are still under definition and will provide a more detailed specification on deliverables such as D2.1.1 and D2.2.1.

### 4.1 System Architecture Overview

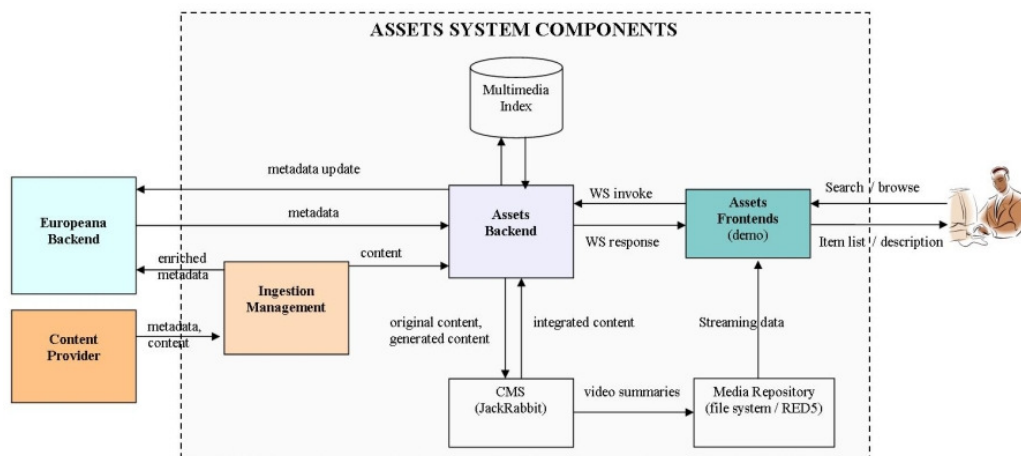
Differently from Europeana project, which stores exclusively the items' metadata within its database, the ASSETS services will need to index and store multimedia content, too. Moreover, the "Video summarisation, adaptation, indexing and retrieval" service will



generate video summaries which need to be made available to the end user. These requirements enforce the enhancement of ASSETS architecture with the usage of “heavyweight-technologies” in comparison to Europeana architecture.

Anyway, one of our project goals is to implement high quality services and to integrate as many as possible into the Europeana portal. Therefore, the ASSETS architecture needs to follow as much as possible the Europeana architecture, technologies and implementation guidelines.

The proposed system architecture is sketched in the following figure.



**Figure 15 – Overview of the ASSETS System Architecture**

The dashed line marks the border of the ASSETS system and its external interfaces. The named arrows represent the dataflows exchanged between the ASSETS system components (internally or with the outer world).

The assets services needs to communicate with:

- Content Providers Portal: which needs to provide an OAI-PMH interface for metadata harvesting and an URI for content harvesting;
- Europeana Backend: which will be accessed through its Web API for updating the Europeana metamodel and metadata with the one created by ASSETS services;
- End Users: access the ASSETS search (and browsing) services from their browser.

#### 4.1.1 System Components

The Assets internal architecture is composed from 3 main software modules. First of them has the role of collecting the metadata information and the content from the content providers and submitting it for storage into the Assets&Europeana databases (Ingestion Management). The second one implements the business functionality (Assets Backend) and makes it available on Internet through a Web API. The third component implements the Graphical User Interface (Assets Frontend) which offers a rich set of browsing and searching functionality for end users. Further information regarding these components is available in the sections: Ingestion Management, Assets Backend and Assets Frontends

If the servlet container (on which the frontend application will be deployed) doesn't handle correctly the http pseudostreaming requests, RED5 will be used for streaming the media

content. See <http://red5.org/>

Apache Jackrabbit will be used internally for the storing and accessing the multimedia content (binary files). “Apache Jackrabbit is a fully conforming implementation of the Content Repository for Java Technology API (JCR, specified in JSR 170 and 283).” See: <http://jackrabbit.apache.org/>

The Assets services will extract metadata and different features from the media content (i.e. text, images, 3D models, audio and video files) and will store this information into the multimedia index.

#### 4.1.2 ASSETS Software Architecture

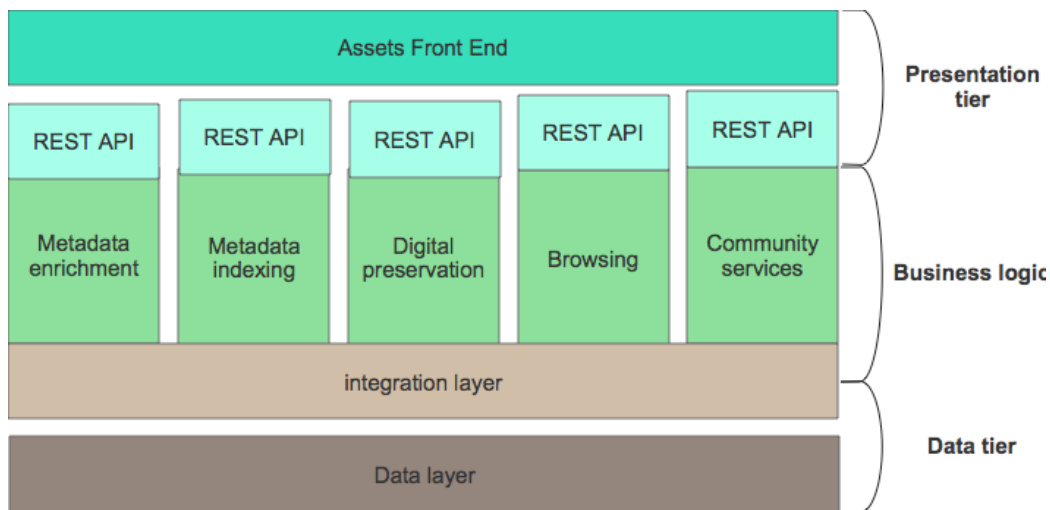


Figure 16 - ASSETS Three-Tier Architecture

Considering the “three-tier architecture” paradigm typically adopted to describe software systems, the ASSETS system can be described as follows:

**Presentation tier:** the ASSETS system will provide RESTful APIs enabling external applications to use its functionalities and a set of GUIs for managing specific functionalities such as data ingestion and data access and browsing.

**Business logic:** a set of autonomous software components implementing the following functionalities:

- Ingestion: Metadata Enrichment, Heterogeneity Reduction, Information Extraction and Classification;
- Indexing, Ranking and Retrieval;
- Digital Preservation;
- Browsing and Content Characterisation;
- Community services: tagging, user generated content, etc.

A middleware, implementing integration functionalities, enables interaction between components.

**Data tier:** enables software components to interact in a transparent way with data repositories. The ASSETS system will need to manage different kind of information objects: multimedia objects, structured data, text etc. For this reason there will be several different kinds of data management stores.

The main goal in adopting the described architecture is to obtain a loosely coupled system: each service component has no (or little) knowledge of other components and also is not supposed to know which tools are actually used to store or manage data, this kind of knowledge is implemented in the integration layer module.

## 4.2 Integration layer and Interfaces

The integration layer was designed for implementing:

- functionalities that enable ASSETS services to interact each others;
- functionalities implementing interactions with the data layer;
- functionalities shared between service components.

These functionalities include: the access to the information stored into the Europeana database and Solr index, the unified concept for application configuration, the common data-model used by ASSETS components, the ORM framework for data storage based on MongoDB.

The Assets integration layer is implemented by 5 components:

- Assets data-model
- Assets common-api
- Assets common-server-api
- Assets common-server
- Assets common-client

### 4.2.1 ASSETS Data-Model Component

The main goal of the common data-model is to offer a common representation for the information exchanged between Assets services. The component implements an AbstractFactory pattern for the instantiation of the domain objects.

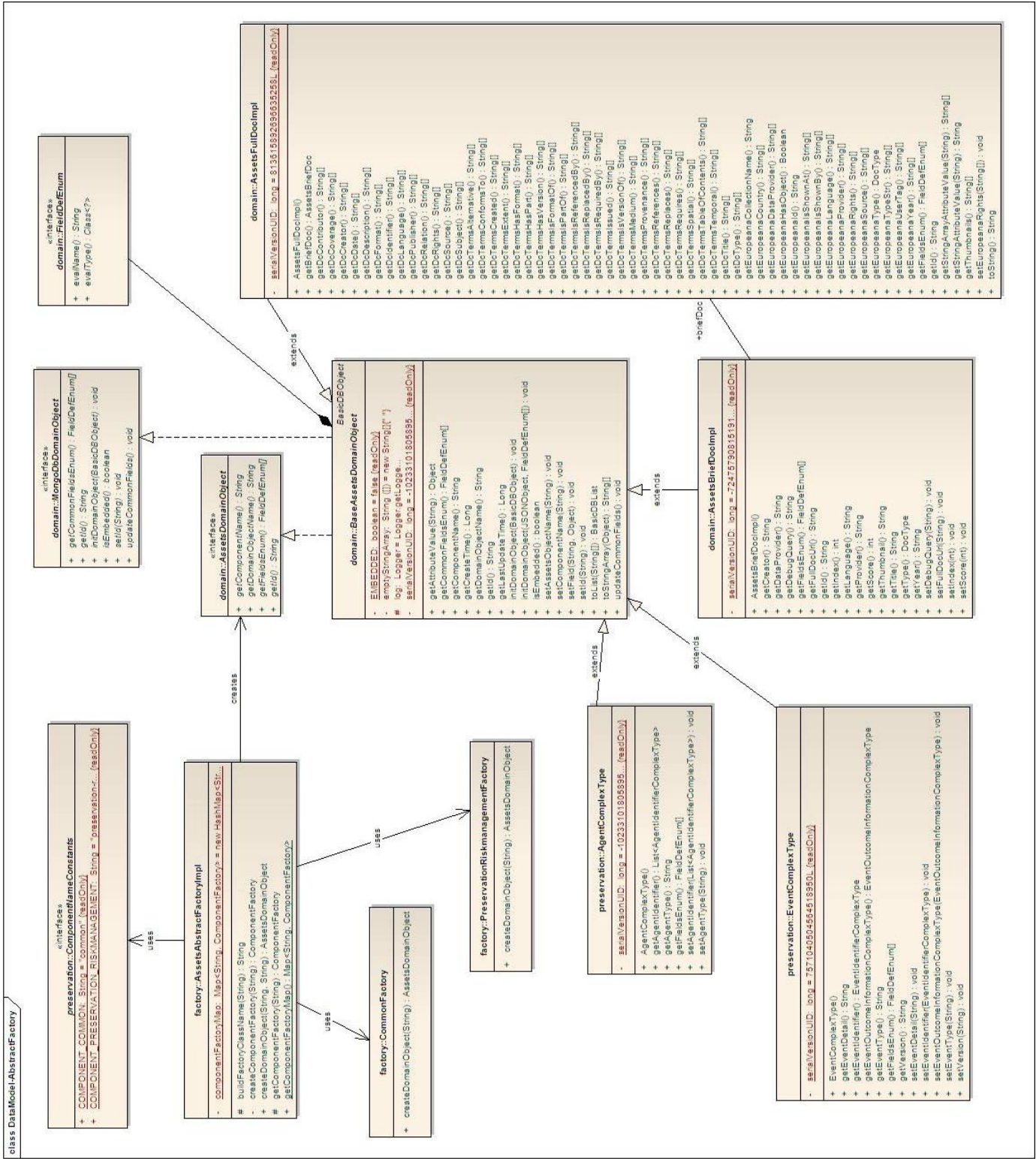


Figure 17 - Abstract Factory Implementation





#### 4.2.2 Core Data Model

ASSETS integration layer offers access to the information managed by the Europeana application. This information is organized in collections provided by individual content providers or provider aggregators (EuropeanaCollection objects), and the metadata containing the descriptions of the masterpieces (FullDoc objects). Each object in the collection is identified by a set of properties which are grouped in Europeanald objects. In order to be able to expose these objects over the Rest interface, the Assets adapter classes enhance the core application objects by adding JAXB serialization annotations (EuropeanaCollectionAdapter, EuropeanaldAdapter, FullDocAdapter). The metadata descriptions are stored into the Solr index for providing fast search and access through the Europeana Portal. Assets needs to process and persist these objects in a database, therefore the AssetsFullDoc representation of the objects was created. All Assets domain objects need to implement a common interface: AssetsDomainObject. The other objects of the Assets domain model are presented together with the components which are responsible for their management.



Interface Name	AssetsAbstractFactory
Key Concepts	Domain object, component factory
Operations	<ul style="list-style-type: none"> <li>• <b>public AssetsDomainObject createDomainObject(String componentName, String domainObjectName)</b> - This method is used for the instantiation of the given domain objects from the given component</li> </ul>

Interface Name	ComponentFactory
Key Concepts	Instantiation of current component domain objects. Each component will provide a ComponentFactory Implementation.
Operations	<ul style="list-style-type: none"> <li>• <b>public AssetsDomainObject createDomainObject(String domainObjectName)</b> - This method creates an instance of the domain object identified by the given domain object name.</li> </ul>

Interface Name	AssetsDomainObject
Key Concepts	Field enumeration
Operations	<ul style="list-style-type: none"> <li>• <b>public String getId()</b> - Retrieve the identifier of the object stored in database</li> <li>• <b>public String getDomainObjectName()</b> - This method returns the logical name of the current domain object. By default, the simple classname will be used as object name.</li> <li>• <b>public String getComponentName()</b> - This method returns the name of the component to which the current domain object belongs.</li> <li>• <b>public FieldDefEnum getFieldsEnum()</b> - This method returns the list with the names of the attributes which hold information which defines the current domain object.</li> </ul>

### 4.2.3 ASSETS Common API and Common Server API Components

The API components implement functionality that is common and should be reused by all Assets Services. There is functionality which is independent from the location at which it is used. For example, the reading of the configuration files, conversion between different textual representations of date information can be used without any restrictions in server-side and client-side components. This functionality is implemented in the “common-api” component.

Further more, there is functionality which needs to access restricted or protected resources, like the persistence system. This functionality is made available only for being accessed on



the server; therefore, it resides in the “common-api-server” component. In the current version of the system, this component implements a generic implementation of the MongoDB based DataStore and the logging for the media indexing functionality. Further common functionality will be identified during the implementation of the Assets services.

Service Name	Common Server API
Responsibility	<ol style="list-style-type: none"><li>1. Generic data Store</li><li>2. Logging support for media indexing</li></ol>
Provided Interfaces	<ol style="list-style-type: none"><li>1. DataStoreDao</li><li>2. MediaIndexingLogService</li></ol>
Dependencies	Common data model

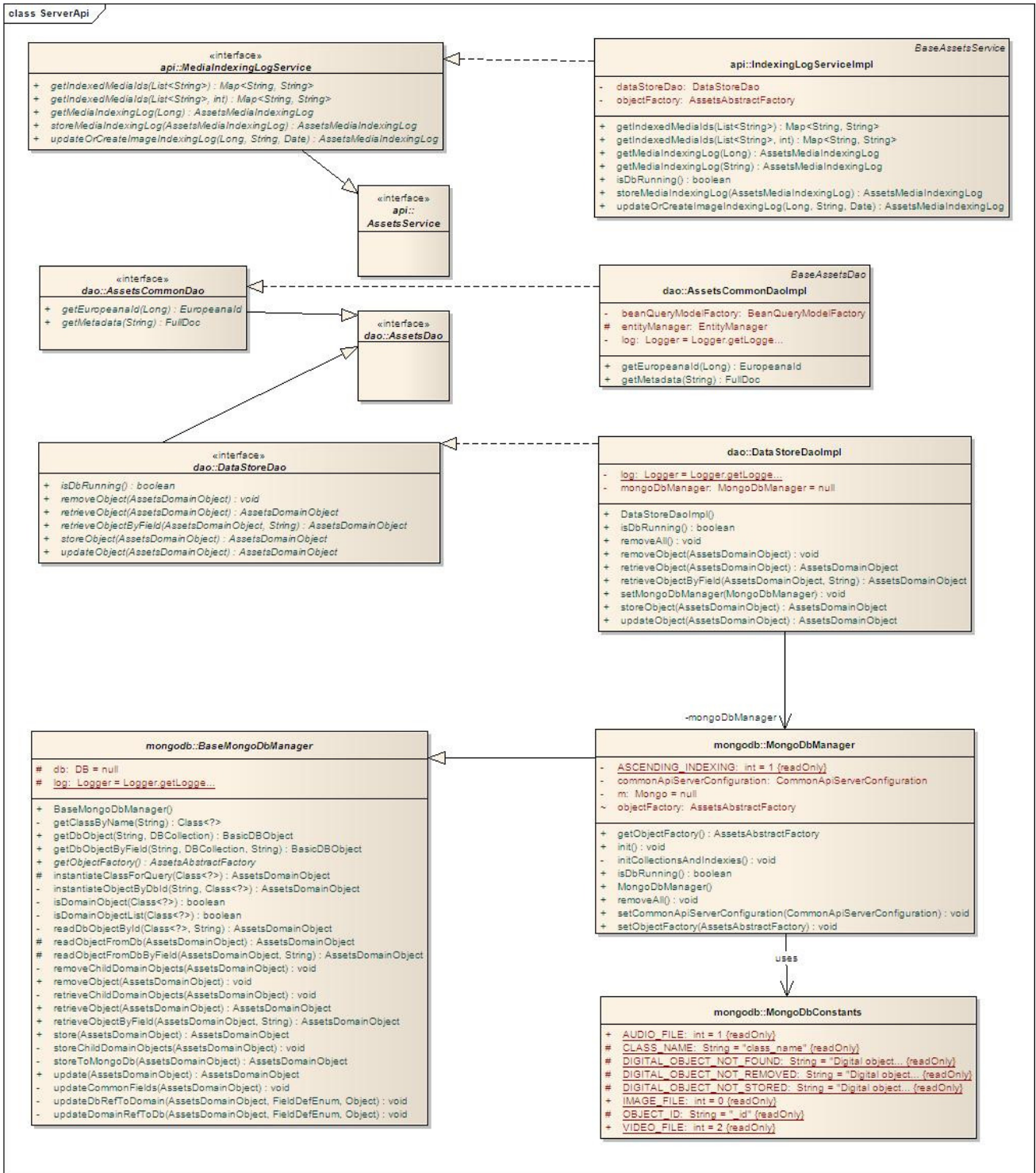


Figure 19 – Common Server API

Interface Name	DataStoreDao
Key Concepts	AssetsDomainObject
Operations	<ul style="list-style-type: none"> <li>• <b>public AssetsDomainObject storeObject(AssetsDomainObject object)</b> - Stores the given domain object into the database</li> <li>• <b>public AssetsDomainObject retrieveObject(AssetsDomainObject object)</b> - Reads the object identified by the given object id from the database</li> <li>• <b>public AssetsDomainObject retrieveObjectByField(AssetsDomainObject object, String fieldName)</b> - Reads the object identified by the passed field from the database</li> <li>• <b>public AssetsDomainObject updateObject(AssetsDomainObject object)</b> - Updates the object identified by the given object id from the database</li> <li>• <b>public void removeObject(AssetsDomainObject object)</b> - This method removes the given object from the database</li> <li>• <b>public boolean isDbRunning()</b> - This utility method checks if the database connection can be established</li> </ul>

Interface Name	MediaIndexingLogService
Key Concepts	AssetsDomainObject
Operations	<ul style="list-style-type: none"> <li>• <b>public Map&lt;String, String&gt; getIndexedMediaIds(List&lt;String&gt; europeanaUris)</b> - This method evaluates the indexing log for media objects and returns a map of EuropeanId.ids which are already available in the media index.</li> <li>• <b>public Map&lt;String, String&gt; getIndexedMediaIds(List&lt;String&gt; europeanaUris, int type)</b> - This method evaluates the indexing log for media objects and returns a map of EuropeanId.ids which are already available in the media index.</li> <li>• <b>public AssetsMediaIndexingLog getMediaIndexingLog(Long europeanId)</b> - This method returns the AssetsMediaIndexingLog for the given EuropeanId.id</li> <li>• <b>public AssetsMediaIndexingLog storeMediaIndexingLog(AssetsMediaIndexingLog mediaIndexingLog)</b> - This method stores the AssetsMediaIndexingLog data representation in database. If the object already exists in the database it will be overridden with the current database.</li> <li>• <b>public AssetsMediaIndexingLog updateOrCreateImageIndexingLog(Long europeanId, String europeanaUri, Date imageIndexingDate)</b> - This method updates the image indexing date for the object identified by the given</li> </ul>

	europeanald.
--	--------------

#### 4.2.4 Common Server and Common Client

The common layer of Assets architecture follows the same structure as the regular components. This layer is also provided through a REST and a client API and makes the core information of the assets system available to the other components. The business functionality provided through the Server, Rest and Client interfaces are identically; therefore, they will only be described once in the Server interface.

Service Name	Metadata Management Service
Responsibility	<ol style="list-style-type: none"> <li>1. Define a unified representation for the assets domain model</li> <li>2. Instantiate domain objects</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. MetadataManagementService,</li> <li>2. CommonRest,</li> <li>3. DataManagement</li> </ol>
Dependencies	Europeana Core Data – Model

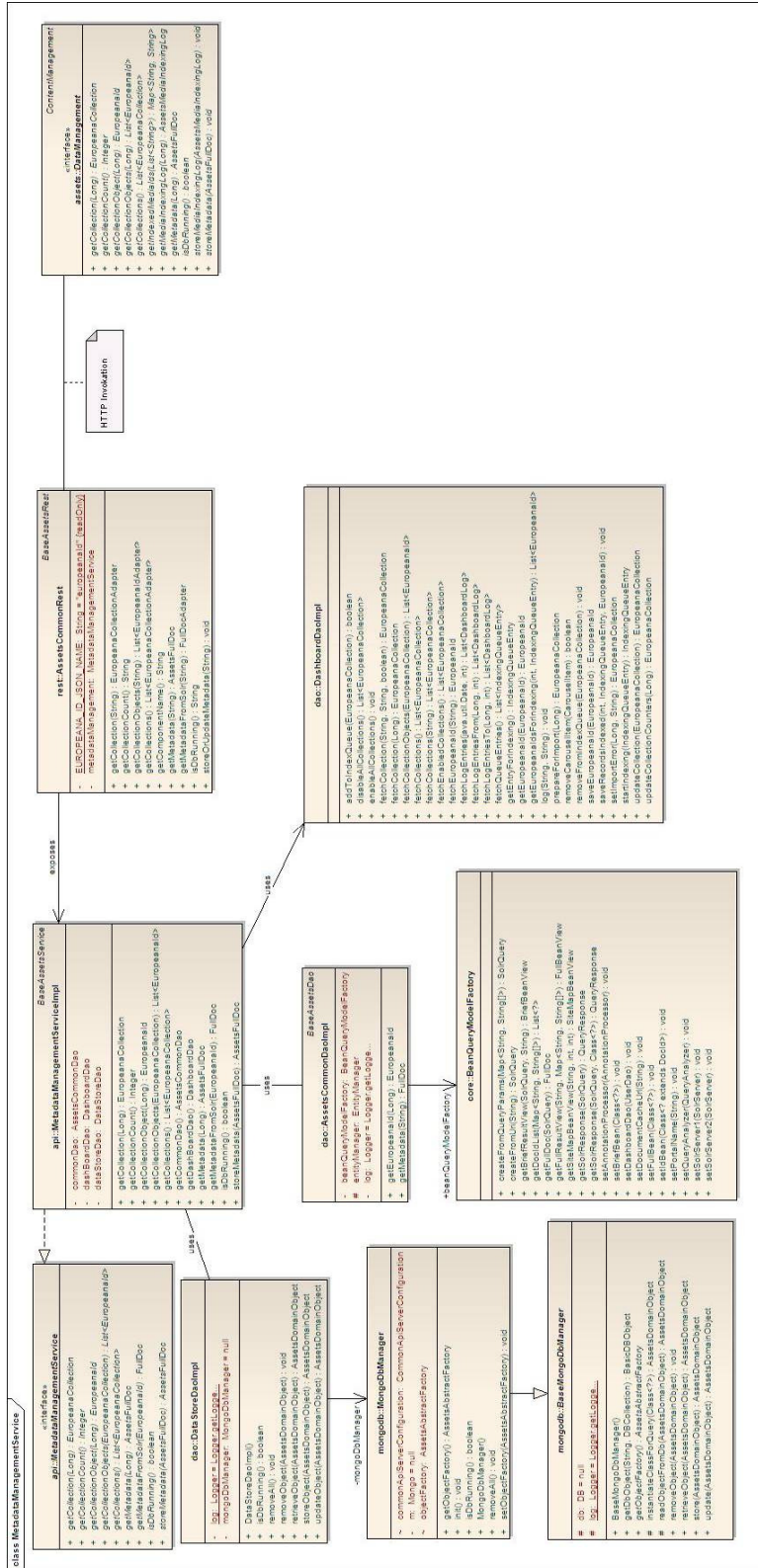


Figure 20 - Metadata Management service



Interface Name	MetadataManagementService
Key Concepts	Collection, CollectionObject, Metadata
Operations	<ul style="list-style-type: none"> <li>• <b>public Integer getCollectionCount()</b> - This method returns the number of collections available into the database</li> <li>• <b>public EuropeanCollection getCollection(Long id)</b> - This method returns the collection identified by the given id</li> <li>• <b>public List&lt;EuropeanCollection&gt; getCollections()</b> - Fetch all collections.</li> <li>• <b>public List&lt;EuropeanId&gt; getCollectionObjects(EuropeanCollection collection)</b> - The list of European ids available in the collection</li> <li>• <b>public EuropeanId getCollectionObject(Long id)</b> - This method returns the EuropeanId object identified by the given database id</li> <li>• <b>public FullDoc getMetadataFromSolr(EuropeanId euid)</b> - This method retrieves the metadata of the collection object from the SolrIndex</li> <li>• <b>public AssetsFullDoc getMetadata(Long euid)</b> - This method retrieves the metadata of the collection object from the database</li> <li>• <b>public AssetsFullDoc storeMetadata(AssetsFullDoc afd)</b> - This method stores the FullDoc metadata representation in database. If the object already exists in the database it will be overridden with the current database.</li> <li>• <b>public Long getCollectionObjectId(String europeanUri)</b> - This method retrieves the ID of the EuropeanId object identified by the given URI</li> </ul>

#### 4.2.5 Notification and Taxonomy Models and Interfaces

The need for a common ‘notification’ component was explicitly identified throughout different parts of the ASSETS project especially in the Work Packages: WP2.3 “Preparing the ground for digital preservation” and WP3.2 “Community Services”. For that reason, this paragraph analyses and describes the rationale and concepts behind the decisions taken for the ASSETS Services: Preservation Notification (i.e. outcome of the Task 2.3.3), Taxonomy-based Notification (i.e. outcome of the Task 3.2.3) and Content Creation (i.e. outcome of the Task 3.2.2).

The notification of occurred events (i.e. events impacting on preservation of archived document, publishing and/or update events) represents an important feature for digital libraries. In fact it enables the process of easily spreading structured information among the members of the digital library community (e.g. new document published by authors. This feature can be also be used for tracking changes occurred on documents, allowing to provide useful documentation and evidence on the integrity and authenticity of the archived documents in the digital library).

For that reason, we propose a **publish/subscribe system** for digital libraries which



continuously evaluates **queries** over a large repository containing **document descriptions**.

When users submit a query, they actually submit a query expression. This contains a set of terms of well-defined vocabularies. The same occurs for the document descriptions, in fact publishers adopt set of terms which belong to vocabularies too. For instance, a publisher at UNESCO could use the UNESCO Thesaurus (<http://www2.ulcc.ac.uk/unesco/thesaurus.htm>) for notifying the publication of a new content about the Coliseum described by terms “3.50 Visual arts/ Architecture / Monuments / Historic monuments”. Any person interested in “historic monuments” could receive that notification.

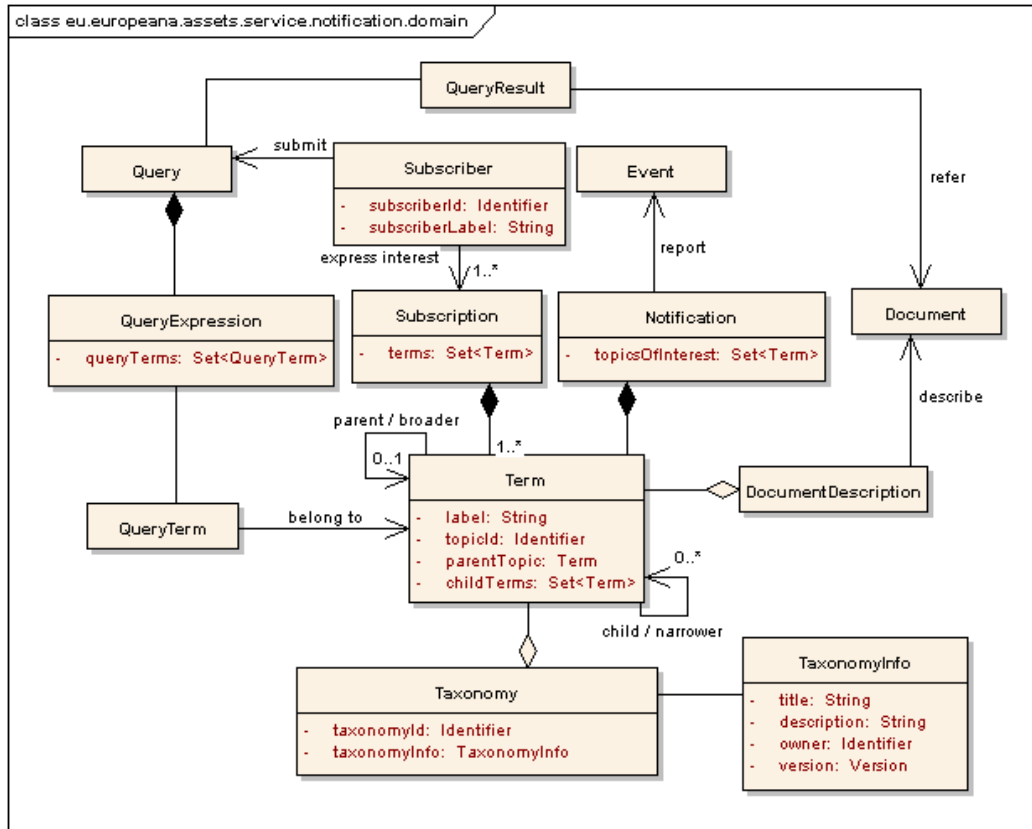
The idea behind the taxonomy-based notification is to suggest subscription for specific topics of interest whenever users submit queries. The subscription allows them to identify topics of interest among the terms contained in the query expression.

The **subscriptions**, the **query expressions** and the **document descriptions**, all rely on a **taxonomy** that is a hierarchically organised set of keywords, or **terms**. The digital library supports insertion, update and removal of a document. Each of these operations is seen as an **event** notifying users whose subscriptions match the document’s description [5].

We focus on digital libraries (DL) which maintain (in a repository) descriptions of documents and pointers to their binary content: that scenario exactly matches with the Europeana DL. In this context publishers are authors that provide to the DL descriptions of their documents and ways to access their contents (e.g. their URIs), whereas a subscriber is a user willing to be informed of any event affecting a document that relates to his topics of interest. We consider a DL model with the following characteristics:

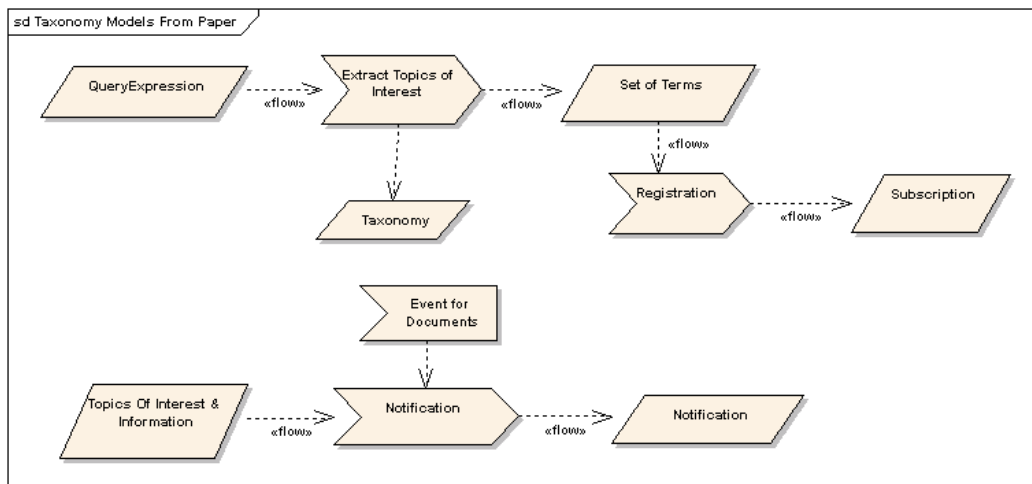
- There is a taxonomy to which authors of documents and subscribers of the library adhere; this taxonomy is just a set of keywords, or terms structured as a tree. An example of a taxonomy is the well known ACM Computing Classification System (The ACM computing classification system, 1999. [www.acm.org/class](http://www.acm.org/class)) ;
- A document is represented in the DL repository by a description (metadata) together with an identifier (say, the document’s URI) allowing to access the document’s content. The description is based on free text and categorization information which is part of the taxonomy;
- A query against the library is a conjunction of terms from the taxonomy (i.e. a conjunctive query) ;
- A user is represented by an identifier together with a subscription; where a subscription is a query defining (intentionally) the documents of interest to the user;

The picture below shows an overview of the common data model for the notification and taxonomy.



**Figure 21 – Notification and Taxonomy Common Model**

It is important to remark that the query expressed by the user contains terms which refer to a taxonomy and that are consequently used for handling the subscription. The picture below shows the processes and its information flow.



**Figure 22 - Notification and Taxonomy Common flow**

Service Name	Notification
Responsibility	<ol style="list-style-type: none"> <li>3. Allows the submissions of messages for specific topics;</li> <li>4. Delivers messages to subscribers;</li> <li>5. Allows the subscription to topics of interest</li> <li>6. Manages the taxonomies</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>4. NotificationManager</li> <li>5. RegistrationManager</li> <li>6. TaxonomyManager</li> </ol>
Dependencies	ASSETS Common

Interface Name	NotificationManager
Key Concepts	Message
Operations	<ul style="list-style-type: none"> <li>• <b>Notification createMessage(Identifier serviceId, Publisher publisher, Set&lt;Term&gt; terms)</b> – allows a publisher, through a service, to create a notification message for events and/or objects, which are represented by a set of terms (of an existing taxonomy);</li> <li>• <b>void publishMessage(Notification notification)</b> – allows a publisher to submit a notification message;</li> <li>• <b>List&lt;Alert&gt; deliverMessages(Identifier serviceId, Identifier subscriptionId, FilteringRule filterRule, int indexFrom, int maxBunch, MessagePolicyAge policyAge)</b> – allows a subscriber to receive a bunch of alert messages for a specific subscription of a service, according to the expressed message policy (e.g. if the message has been posted before the subscription) and the filtering rule (e.g. AND, OR). FilteringRule and MessagePolicyAge will be implemented later;</li> <li>• <b>List&lt;Alert&gt; deliverMessages4Term(Identifier serviceId, Term term, int indexFrom, int maxBunch, MessagePolicyAge policyAge)</b> - allows a subscriber to receive a bunch of alert messages for a specific term of interest, according to the expressed message policy. The messages refer to the exact matching for the term, and not for its children. MessagePolicyAge will be implemented later;</li> <li>• <b>MessageStatus getMessageStatus(Identifier messageId)</b> – returns the status of the delivered alert message. The status may be read or unread. This method will be implemented later;</li> <li>• <b>boolean markAlertAsRead (Identifier subscriberId, Identifier messageId)</b> – allows to set the status of the delivered alert message as read. This method will be implemented later;</li> </ul>

Interface	RegistrationManager
-----------	---------------------



Name	
Key Concepts	Subscriber and Subscription
Provided Interfaces	<ul style="list-style-type: none"> <li>• <b>Subscription createSubscription(Identifier serviceId, Subscriber subscriber, Set&lt;Term&gt; terms)</b> – allows to register a subscription for a specific subscriber. That allows to specify the set of terms of interest for receiving alerts. The operation provides an identifier for the registered subscription;</li> <li>• <b>Subscription updateSubscription(Identifier serviceId, Identifier subscriptionId, Set&lt;Term&gt; terms)</b> – allows to update the information for a registered subscription;</li> <li>• <b>boolean deleteSubscription(Identifier serviceId, Identifier subscriptionId)</b> - allows to remove the registration of a specific subscription;</li> <li>• <b>List&lt;Identifier&gt; getAllSubscriptions(Identifier serviceId, Identifier subscriberId)</b> – allows to obtain all the identifiers of subscriptions created by a service for a specific subscriber. This method will be implemented later;</li> <li>• <b>List&lt;Identifier&gt; getAllSubscribers(Identifier serviceId)</b> –allows to obtain a list of all the identifiers of registered subscribers. This method will be implemented later;</li> </ul>

Interface Name	TaxonomyManager
Key Concepts	Term and Taxonomy
Operations	<ul style="list-style-type: none"> <li>• <b>Identifier createTaxonomy (Identifier serviceId, TaxonomyInfo taxonomyInfo, Taxonomy taxonomy)</b> – allows a service to create and register a taxonomy by providing the taxonomy and its information;</li> <li>• <b>TaxonomyInfo getTaxonomyInfo (Identifier taxonomyId)</b> – allows to obtain the information of a taxonomy through its identifier;</li> <li>• <b>List&lt; TaxonomyInfo&gt; listTaxonomies(Identifier serviceId)</b> - allows to obtain the information of all the registered taxonomies by a service. This method will be implemented later;</li> <li>• <b>Taxonomy getPartOfTaxonomy (termId: Identifier, taxonomyId: Identifier)</b> – allows to obtain the part of taxonomy which has a specific term as root. This method will be implemented later;</li> <li>• <b>boolean replaceTaxonomy (Identifier taxonomyId, Taxonomy newTaxonomy)</b> – allows to replace an existing taxonomy with a new one. This method will be implemented later;</li> <li>• <b>boolean deleteTaxonomy (Identifier taxonomyId)</b> – allows to remove an existing taxonomy. This method will be implemented later;</li> </ul>

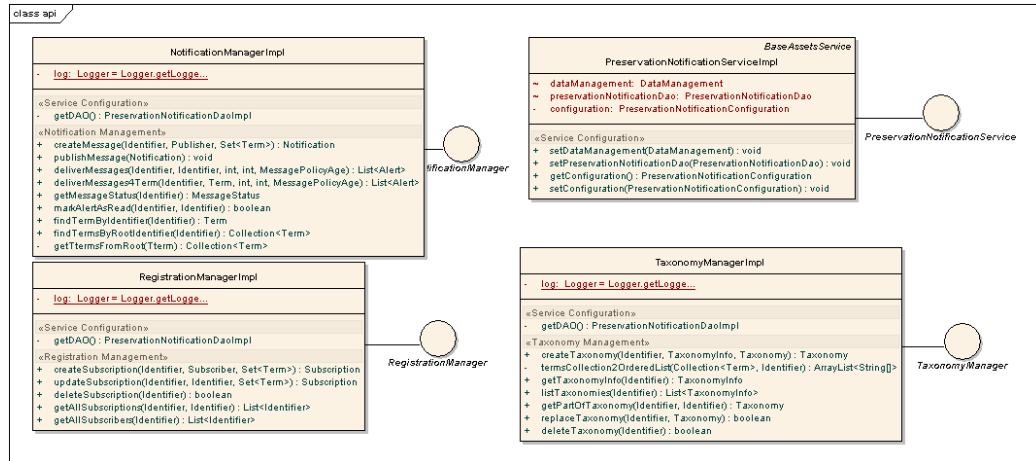


Figure 23 – Common Notification APIs

#### 4.2.6 Session Management and Identification

For many usage scenarios, there is a crucial need to properly handle the identification and certification of the actors throughout the broad spectrum reaching from the object ingestion process up to searching and interacting with the portal. The ASSETS platform has the role of being a content aggregator for Europeana. In this perspective, it is reasonable to have a clear identification of users and/or services through unique identifiers or sessions/tokens. For that reason, it is introduced the generic concept of Identifier defined as a unique identifier for objects, persons and services within the ASSETS federation and processes.

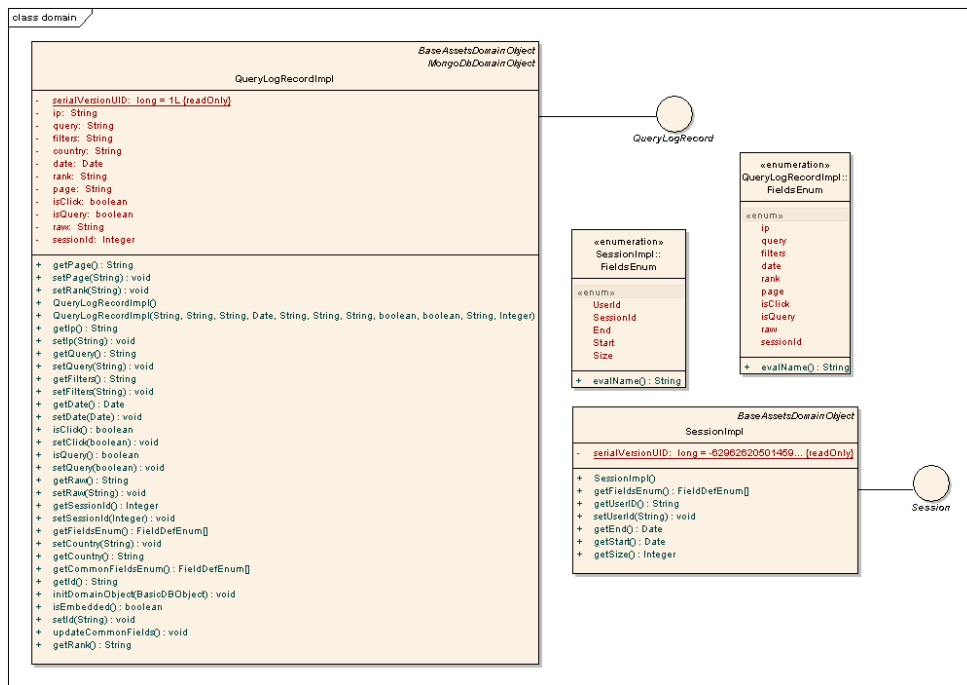


Figure 24 – Common Session and Query Log Record API

### 4.3 The Ingestion Models and Interfaces

The Ingestion Services are composed by the:

- Metadata Cleaning which reduces errors in the metadata and perform a basic enrichment of specific elements of the metadata records;
- Knowledge Extraction which enables to extract relevant structured metadata fields;
- Metadata Classification which supports automated classification of metadata records under a taxonomy of semantic categories.

Post Ingestion Processing allows to harvest and index the multimedia content during the ingestion. More, an Ingestion Workflow guarantees the proper flow of tasks and information.

#### 4.3.1 The Metadata Cleaning Service Models and Interfaces

Notice each interface described above will depend on the key concept "MetadataDataset". Such concept has not been explicitly included.

Service Name	Metadata Cleaning Service
Responsibility	<ol style="list-style-type: none"> <li>1. Basic error correction</li> <li>2. Value normalization</li> <li>3. Basic enrichment.</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. MetadataErrorCorrection</li> <li>2. MetadataValueNormalization</li> <li>3. MetadataFieldEnrichment</li> <li>4. MetadataCleaningManager</li> </ol>
Dependencies	ASSETS common, other modules and services used during the ingestion stage

Interface Name	MetadataErrorCorrection
Key Concepts	MetadataErrorCorrectionDescriptor
Operations	<ul style="list-style-type: none"> <li>• correctRecord</li> </ul>

Interface Name	MetadataValueNormalization
Key Concepts	MetadataValueNormalizerDescriptor
Operations	<ul style="list-style-type: none"> <li>• normalizeRecord</li> </ul>

Interface Name	MetadataFieldEnrichment
----------------	-------------------------



Key Concepts	MetadataFieldEnricherDescriptor
Operations	<ul style="list-style-type: none"> <li>enrichMetadataWithAuthorityFile</li> <li>enrichMetadataWithControlledVocabulary</li> </ul>

Interface Name	MetadataCleaningManager
Key Concepts	MetadataErrorCorrection, MetadataFieldEnrichment, MetadataValueNormalization,
Operations	<ul style="list-style-type: none"> <li>trainMetadataErrorCorrector</li> <li>getStatus</li> <li>listTrainingMethodsForCorrection</li> <li>listMetadataErrorCorrectors</li> <li>deleteMetadataErrorCorrector</li> <li>listMetadataValueNormalizers</li> <li>deleteMetadataValueNormalizer</li> <li>listAuthorityFiles</li> <li>listControlledVocabularies</li> </ul>



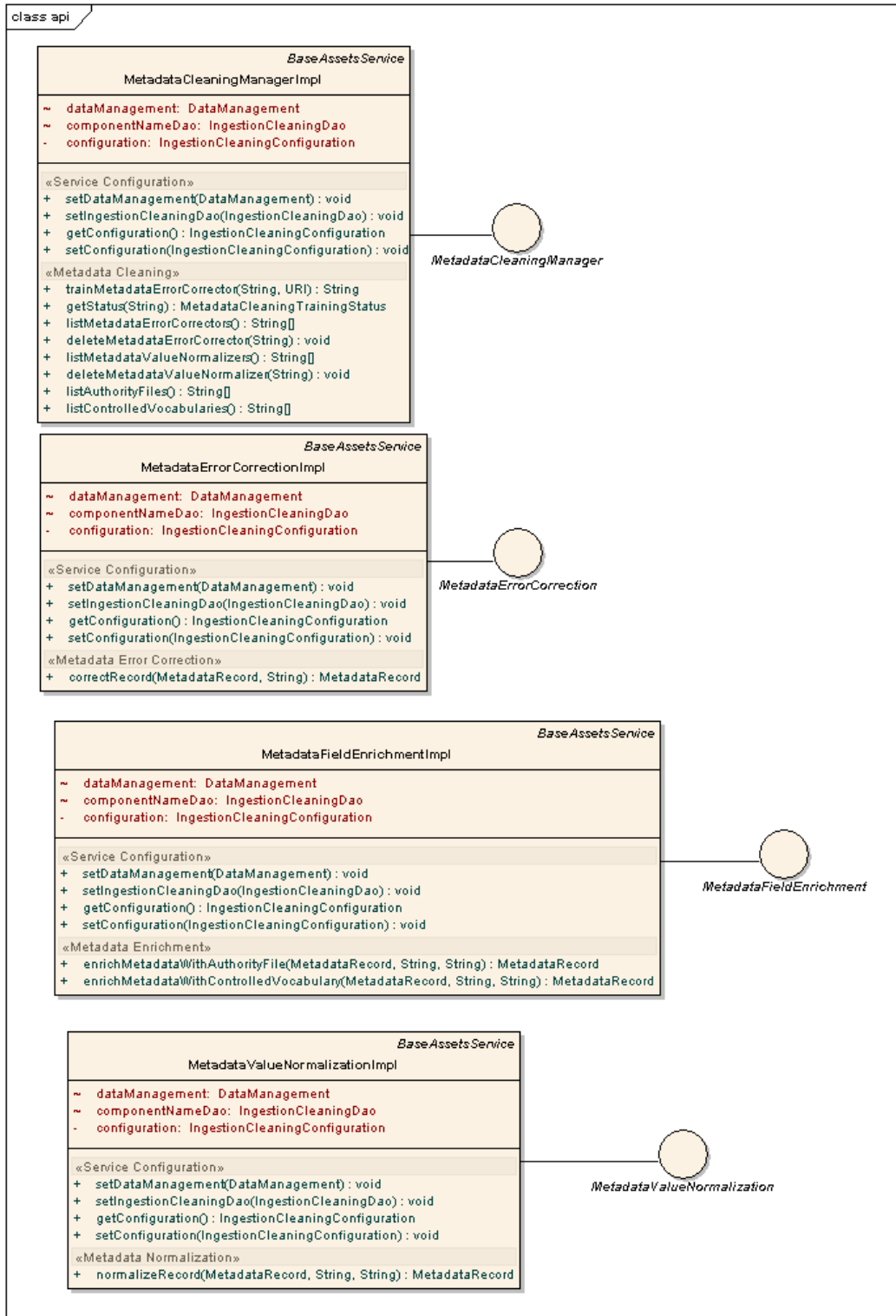


Figure 25 - Ingestion Cleaning API: Overview

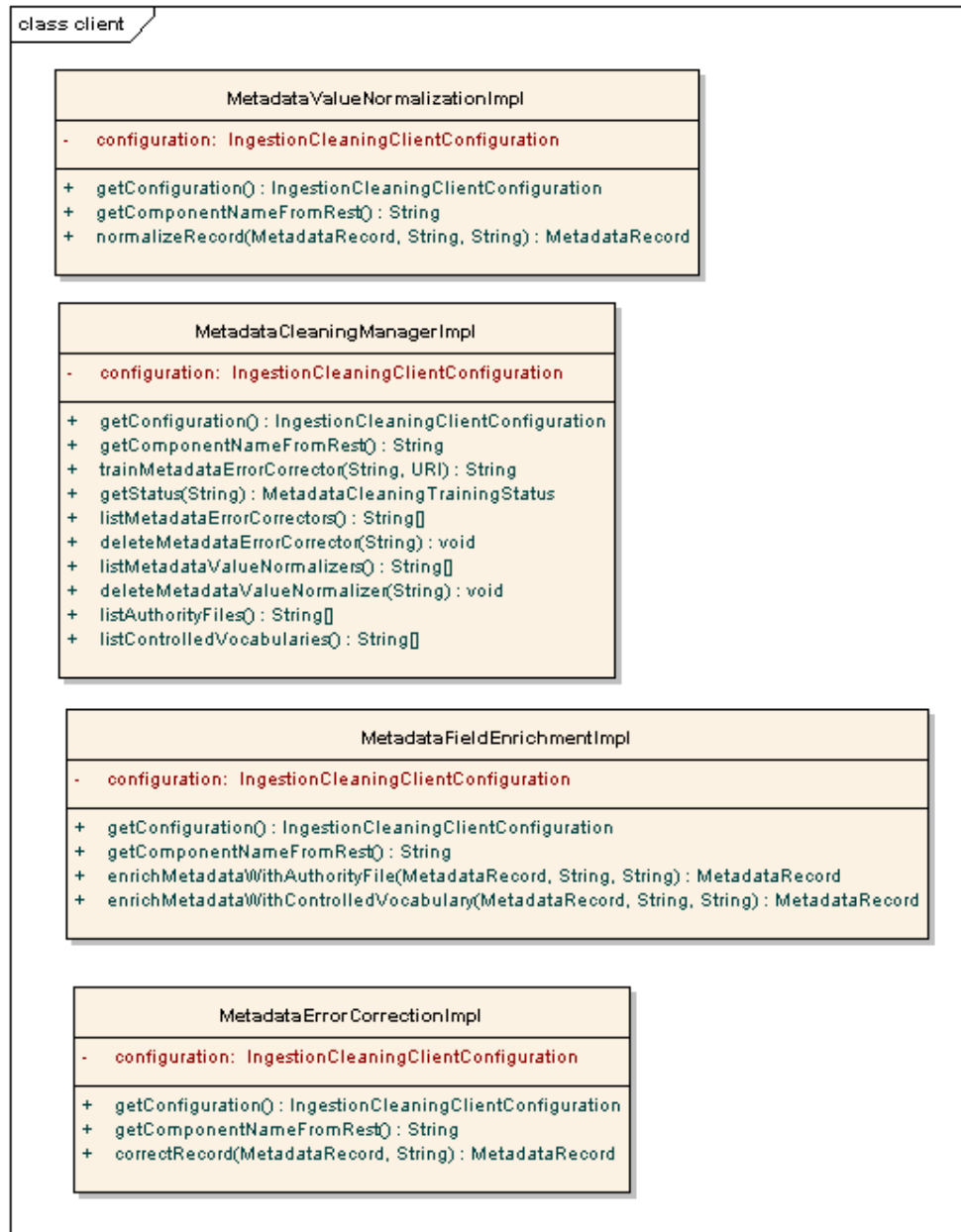


Figure 26 – Ingestion Cleaning : Client Side Models

#### 4.3.2 Knowledge Extraction Models and Interfaces

Service Name	Knowledge Extraction
Responsibility	1. Extraction of structured information from unstructured textual metadata fields of europeana metadata records
Provided Interfaces	1. KnowledgeExtractionTrainer, 2. KnowledgeExtractionManager,



	3. KnowledgeExtractor
Dependencies	ASSETS common, other modules and services used during the ingestion stage

Interface Name	KnowledgeExtractionTrainer
Key Concepts	MetadataKnowledgeExtractionModel, KnowledgeExtractorDescriptor
Operations	<ul style="list-style-type: none"> <li>listMetadataKnowledgeExtractor</li> <li>deleteMetadataKnowledgeExtractor</li> <li>getKnowledgeExtractorDescriptor</li> </ul>

Interface Name	KnowledgeExtractionTrainer
Key Concepts	MetadataKnowledgeExtractionTrainingSet, MetadataKnowledgeExtractionModel
Operations	<ul style="list-style-type: none"> <li>trainMetadataKnowledgeExtractor</li> <li>getTrainingStatus</li> </ul>

Interface Name	KnowledgeExtractor
Key Concepts	MetadataDataset, MetadataKnowledgeExtractionModel
Operations	<ul style="list-style-type: none"> <li>extractKnowledgeFromMetadata</li> </ul>

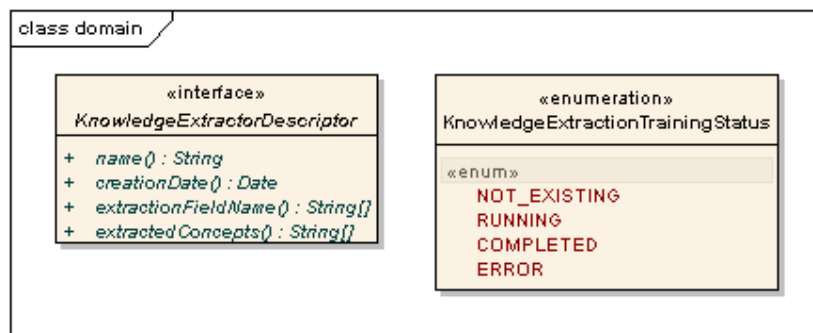


Figure 27 – Ingestion Knowledge Extraction Data Model

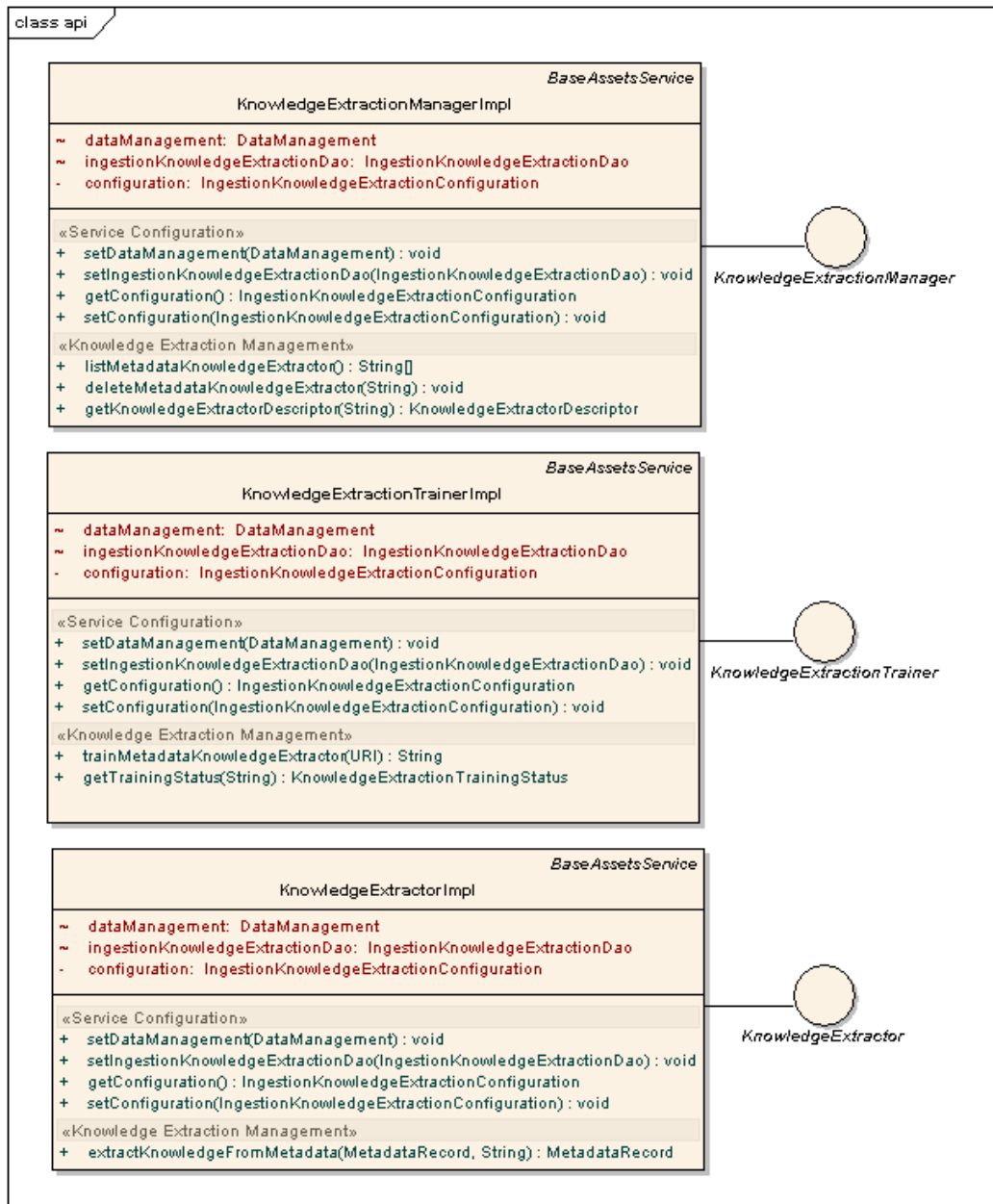


Figure 28 - Ingestion Knowledge Extraction API

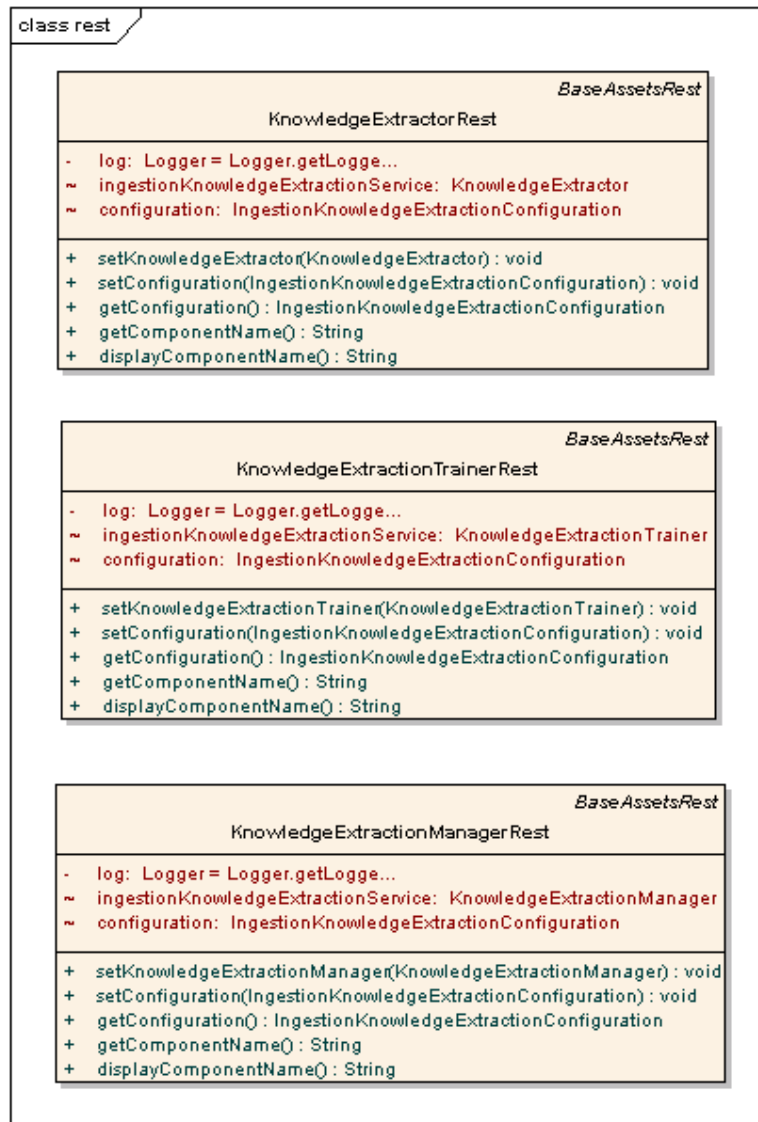


Figure 29 - Ingestion Knowledge Extraction REST API

### 4.3.3 Metadata Classification Models and Interfaces

Service Name	Metadata Classification
Responsibility	1. Classification of europeana metadata records on relevant taxonomies
Provided Interfaces	1. ClassificationTrainer, 2. ClassificationManager, 3. ClassificationService
Dependencies	ASSETS common, other modules and services used during the ingestion stage

Interface Name	ClassificationTrainer
Key Concepts	MetadataClassificationTrainingSet, MetadataClassificationModel
Operations	<ul style="list-style-type: none"> <li>trainMetadataClassifier</li> </ul>

Interface Name	ClassificationManager
Key Concepts	MetadataClassificationModel
Operations	<ul style="list-style-type: none"> <li>listMetadataClassifier</li> <li>deleteMetadataClassifier</li> </ul>

Interface Name	ClassificationService
Key Concepts	MetadataDataset, MetadataClassificationModel
Operations	<ul style="list-style-type: none"> <li>classifyMetadata</li> </ul>

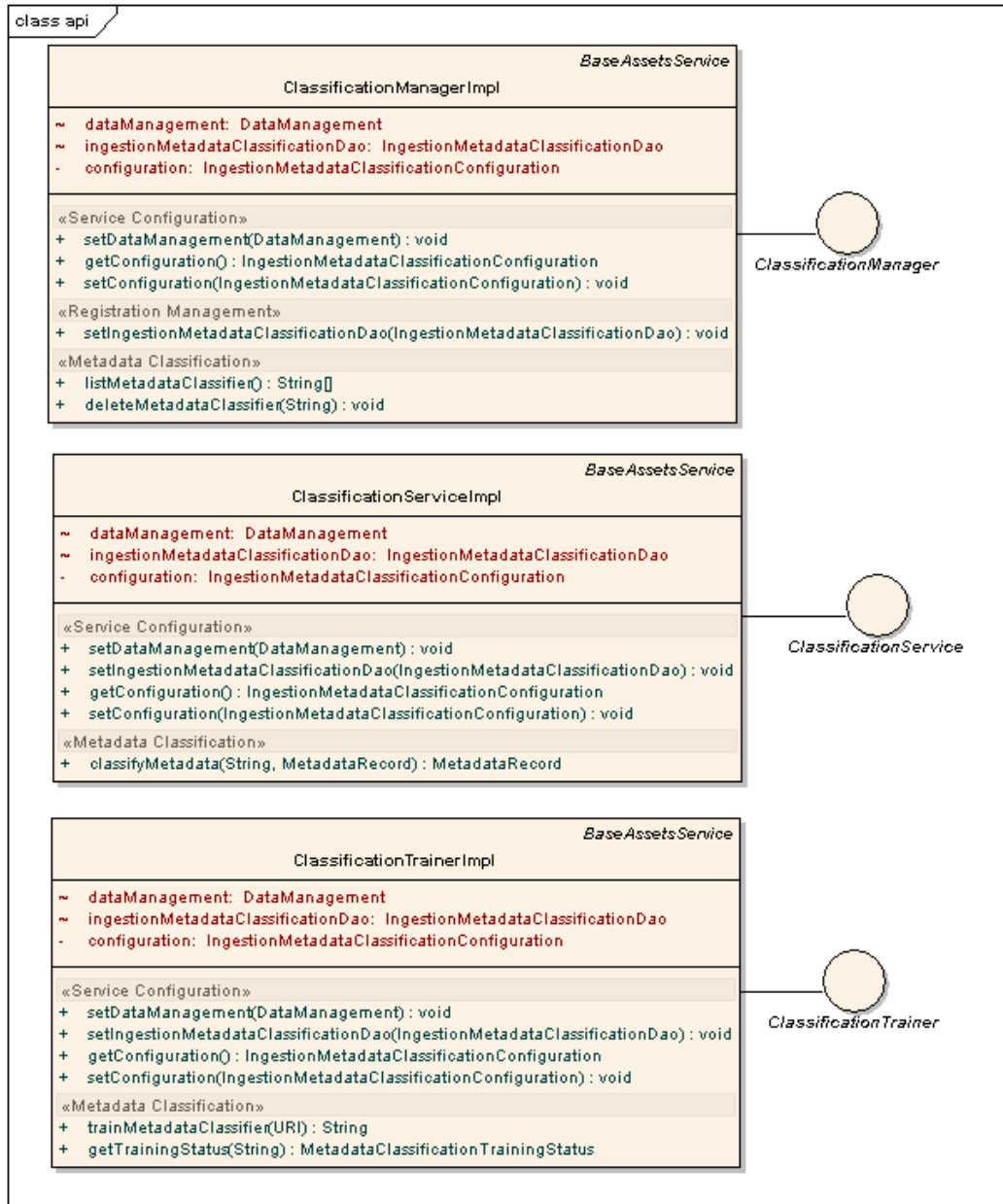


Figure 30 – Ingestion Metadata Classification Service API

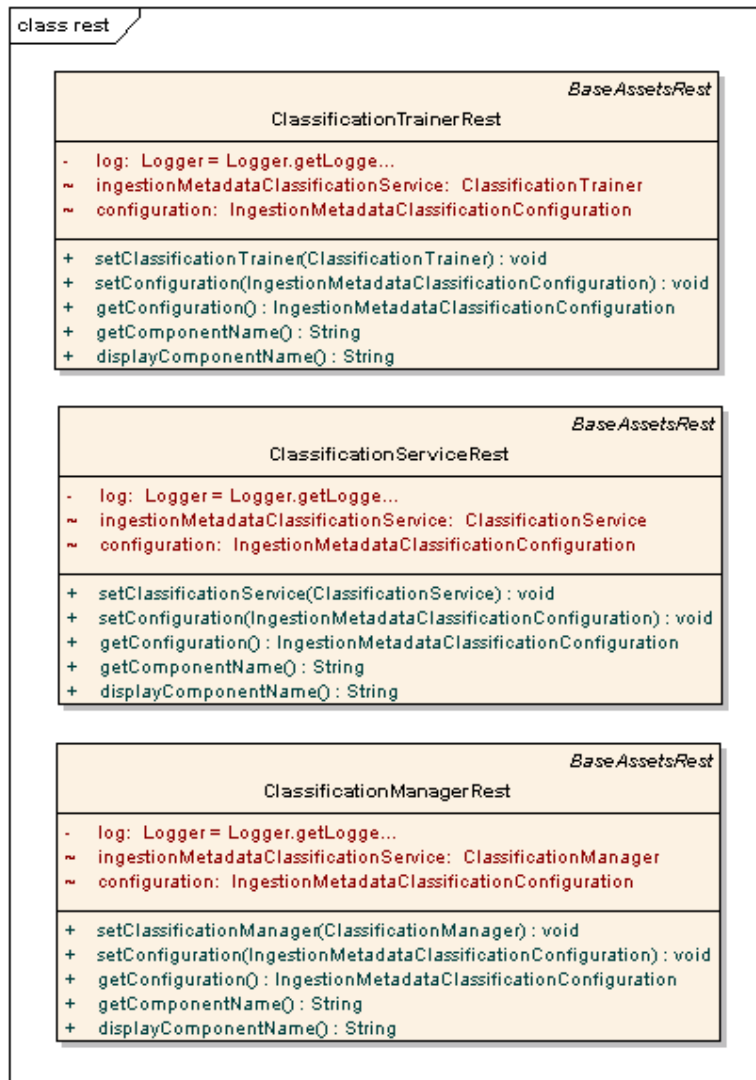


Figure 31 - Ingestion Metadata Classification REST API

#### 4.3.4 Ingestion Workflow Management Models and Interfaces

##### Concept Definitions

- **configuration time:** execution phase during which the plugins are configured. In OSGi this corresponds to the registration phase, during which validity checks are performed by the system.
- **processing time:** execution phase during which the MetaDataRecords are being processed

##### Workflows

A number of predefined workflows will be provided which cover the standard steps for processing and ingesting collections in Europeana platform. These workflows can be adapted and tweaked by technical staff for certain collections. The ingestion team needs to





assign a workflow to the collection before its going to be processed.

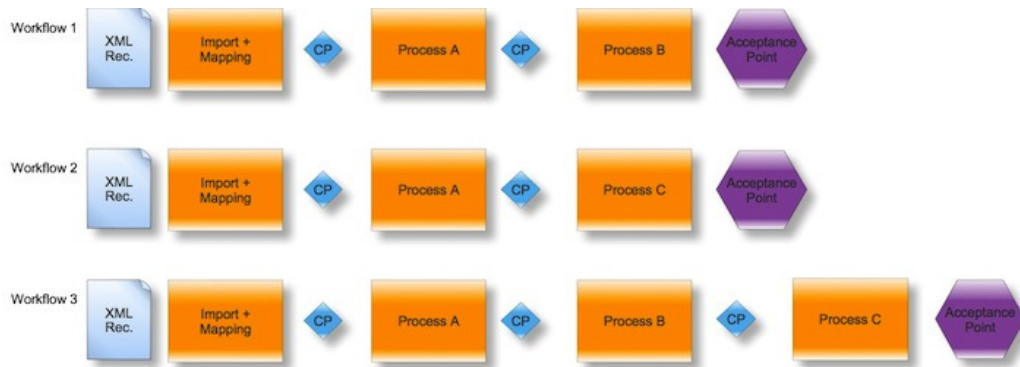


Figure 32 – Ingestion Workflows

### Processing Model

Due to the necessity of optimal resource usage, each process part is asynchronous executed within a thread pool. A plugin must make explicit if it is not thread safe - in which case the framework ensures that no more than one thread at a time uses the plugin. To uncouple each processing block from each other a FIFO queue will be provided. The input queue will thereby be filled by the framework controller to ensure, that the framework is in control of all load balancing issues.

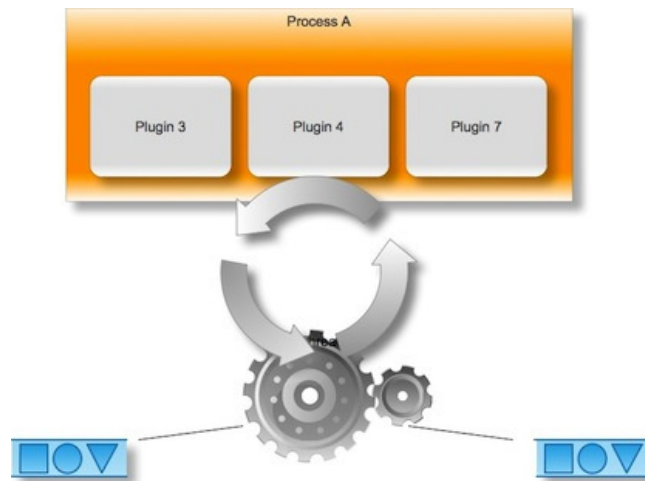


Figure 33 – Ingestion Processing

The ingestion workflow management is developed as a joint effort with Europeana and The European Library. It provides a framework for a scalable and robust execution of the ingestion of large quantities of meta-data records and allows specialized processing by using a plugin based mechanism.



Service Name	Unified Ingestion Manager
Responsibility	<ol style="list-style-type: none"> <li>1. definition of ingestion workflows;</li> <li>2. workflow execution orchestration;</li> <li>3. reporting</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. Workflow,</li> <li>2. MetaDataRecord,</li> <li>3. IngestionPlugin,</li> <li>4. SavePoint,</li> <li>5. Execution,</li> <li>6. Orchestrator</li> </ol>
Dependencies	Apache Karaf OSGi implementation

Interface Name	Workflow
Key Concepts	Representation of a workflow, composed of multiple WorkflowSteps.
Operations	<ul style="list-style-type: none"> <li>• String getName() - @return name of the workflow, should be reasonable meaningful</li> <li>• String getDescription() - @return description of this specific workflow (what does it perform, what should be the outcome, etc.)</li> <li>• WorkflowStart getStart() - @return defined start point of work flow</li> <li>• List&lt;IngestionPlugin&gt; getSteps() - @return plugins as steps in this workflow</li> <li>• boolean isSavepoint(String pluginName) - @return Is this a save point plugin?</li> <li>• boolean isMandatory(String pluginName) - @return Is this a mandatory plugin, so unsuccessful processing is a failure?</li> </ul>

Interface Name	IngestionPlugin
Key Concepts	Definition of a plugin that processes meta-data records. Services that provide ingestion-time capabilities need to implement this interface. An ingestion plugin is a single processing step within a workflow
Operations	<ul style="list-style-type: none"> <li>• String getName() - Get the class name of the plugin which is used to register the plugin with the registry.@return the name for this plugin (should be Plugin.class.getSimpleName()).</li> <li>• String getDescription() -Get the description of the plugin which is</li> </ul>

provided to the operators when starting analyzing workflows.

- TKey<?, ?> getInputFields() - Get the list of fields this plugin wants to operate on. This is used for information purposes, so that it can be validated if the records hold these data. @return a list of fields this plugin requires
- TKey<?, ?> getOptionalFields() - Get the list of fields this plugin would like to operate on or can get additional information for the working process. This is used for information purposes, so that it can be validated if the records hold these data. @return a list of fields this plugin requires
- TKey<?, ?> getOutputFields() - Get the list of output fields. @return a list of fields this plugin creates
- void initialize() - Initialize the plugin when it is loaded in the OSGI container and attached to the uim registry.
- void shutdown() - Shutdown the plugin when it is removed from the uim registry (due to OSGI shutdown or reinstallation etc).
- List<String> getParameters() - List of configuration parameters this plugin can take from the execution context to be configured for a specific execution. @return list of configuration parameters.
- int getPreferredThreadCount() - A plugin is always executed within a thread pool, this parameter defines the preferred size of the pool. Plugins should know best, what's a good level of parallelism. @return number of threads this plugin should usually be processed.
- int getMaximumThreadCount() - Number of maximum threads. The plugin might specify here one (1) if it is not thread safe. @return the number of maximal threads
- void initialize(ExecutionContext context) throws IngestionPluginFailedException - Initialization method for an execution context. The context holds the properties specific for this execution. @param context holds execution depending information for this processing call. This context can change for each call, so references to it have to be handled carefully. @throws IngestionPluginFailedException is thrown if the initialization fails and so the plugin is not ready to take records for this {@link ExecutionContext}
- void completed(ExecutionContext context) throws IngestionPluginFailedException - Finalization method (tear down) for an execution. At the end of each execution this method is called to allow the plugin to clean up memory or external resources. @param context holds execution depending information the {@link ExecutionContext} for this processing call. This context can change for each call, so references to it have to be handled carefully. @throws IngestionPluginFailedException is thrown if the tear down encountered a severe failure during deleting external resources, so that executing it in a new {@link ExecutionContext} will most likely fail

	<ul style="list-style-type: none"> <li>boolean processRecord(MetaDataRecord mdr, ExecutionContext context) throws IngestionPluginFailedException, CorruptedMetadataRecordException - Process a single meta data record within a given execution context. It returns true, if processing went well and false, if something failed. NOTE, false in this context means only that the plugin could not do its work, but neither is the {@link MetaDataRecord} corrupted nor is the plugin itself damaged, so that the record can further processed and this plugin can take other records as well. Furthermore, additional information can be logged ({@link LoggingEngine}). @param mdr the {@link MetaDataRecord} to process @param context holds execution depending, information the {@link ExecutionContext} for this processing call. This context can change for each call, so references to it have to be handled carefully. @return true, if the plugin could do its work and false if something failed during processing @throws IngestionPluginFailedException is thrown if the plugin encounters a severe problem and it is therefore impossible to process any more records for this {@link ExecutionContext} @throws CorruptedMetadataRecordException the plugin encountered a severe problem for a specific {@link MetaDataRecord} so that further processing of this specific {@link MetaDataRecord} does not make sense any longer.</li> </ul>
--	---

Interface Name	ActiveExecution
Key Concepts	Type-safe representation of a meta-data record
Operations	<ul style="list-style-type: none"> <li>getId</li> <li>addField</li> <li>addQField</li> <li>setField</li> <li>setQField</li> </ul>

Interface Name	Execution
Key Concepts	An Execution in a running state. It keeps track of the overall progress.
Operations	<ul style="list-style-type: none"> <li>StorageEngine getStorageEngine()</li> <li>public void setPaused(boolean paused);</li> <li>boolean isPaused();</li> <li>boolean isFinished() - test the execution if all tasks are done eather completely finished or failed. so if true: scheduled == finished + failed</li> <li>void setThrowable(Throwable throwable);</li> </ul>



	<ul style="list-style-type: none"> <li>• <code>Throwable getThrowable();</code></li> <li>• <code>Queue&lt;T&gt; getSuccess(String name);</code></li> <li>• <code>Queue&lt;T&gt; getFailure(String name);</code></li> <li>• <code>Set&lt;Task&gt; getAssigned(String name);</code></li> <li>• <code>void incrementCompleted(int count); int getProgressSize();</code> - gives an estimate of tasks/records which are currently in the pipeline. Note that failed tasks are not counted. The system can not guarantee the number of records, due to the problem that some of the tasks might change their status during the time of counting.</li> <li>• <code>int getCompletedSize();</code> - gives the number of tasks/records which are completely finished successful by all steps.</li> <li>• <code>int getFailureSize()</code> - gives the number of tasks/records which have failed on the way through the workflow no matter where.</li> <li>• <code>int getScheduledSize()</code> - gives the number of tasks/records which have been scheduled to be processed in the first place. So <code>scheduled = progress + finished + failure</code>.</li> <li>• <code>int getTotalSize()</code> - gives the number of records which this execution will need to deal with. If not possible to estimate <code>Integer.MAX_VALUE</code> is given.</li> <li>• <code>List&lt;WorkflowStepStatus&gt; getStepStatus();</code></li> <li>• <code>WorkflowStepStatus getStepStatus(IngestionPlugin step);</code></li> <li>• <code>public Properties getProperties();</code></li> <li>• <code>void waitUntilFinished();</code></li> <li>• <code>void incrementScheduled(int work);</code></li> </ul>
--	--

Interface Name	Orchestrator
Key Concepts	Workflow execution orchestration
Operations	<ul style="list-style-type: none"> <li>• <code>public String getIdentifier();</code></li> <li>• <code>ActiveExecution&lt;?&gt; executeWorkflow(Workflow w, DataSet dataset);</code></li> <li>• <code>ActiveExecution&lt;?&gt; executeWorkflow(Workflow w, DataSet dataset, Properties properties);</code></li> <li>• <code>&lt;T&gt; ActiveExecution&lt;T&gt; getActiveExecution(long id);</code></li> <li>• <code>&lt;T&gt; java.util.Collection&lt;ActiveExecution&lt;T&gt;&gt; getActiveExecutions();</code></li> <li>• <code>void shutdown();</code></li> </ul>

#### 4.3.5 Post-Ingestion Processing

This service supports the harvesting of the metadata based on the OAI protocol for Metadata Harvesting.

Service Name	Post-Ingestion Processing
Responsibility	<ol style="list-style-type: none"> <li>1. multimedia content harvesting,</li> <li>2. multimedia content indexing</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. MultimediaContentHarvesting,</li> <li>2. MultimediaContentIndexing</li> </ol>
Dependencies	Assets Common, Europeana Core, Text Indexing, Image Indexing, Audio Indexing, 3D Indexing

Interface Name	MultimediaContentHarvesting
Key Concepts	<p>MultimediaContent</p> <p>MultimediaContentUrl</p> <p>ObjectMetadata (FullDoc/ESE/EDM)</p>
Operations	<ul style="list-style-type: none"> <li>• downloadMultimediaContent</li> </ul>

Interface Name	MultimediaIndexing
Key Concepts	Multimedia Index, MultimediaContent
Operations	<ul style="list-style-type: none"> <li>• startIndexing,</li> <li>• saveIndexedRecord,</li> <li>• getIndexQueueSize</li> </ul>
Extends	<ul style="list-style-type: none"> <li>• Core-Indexing</li> </ul>

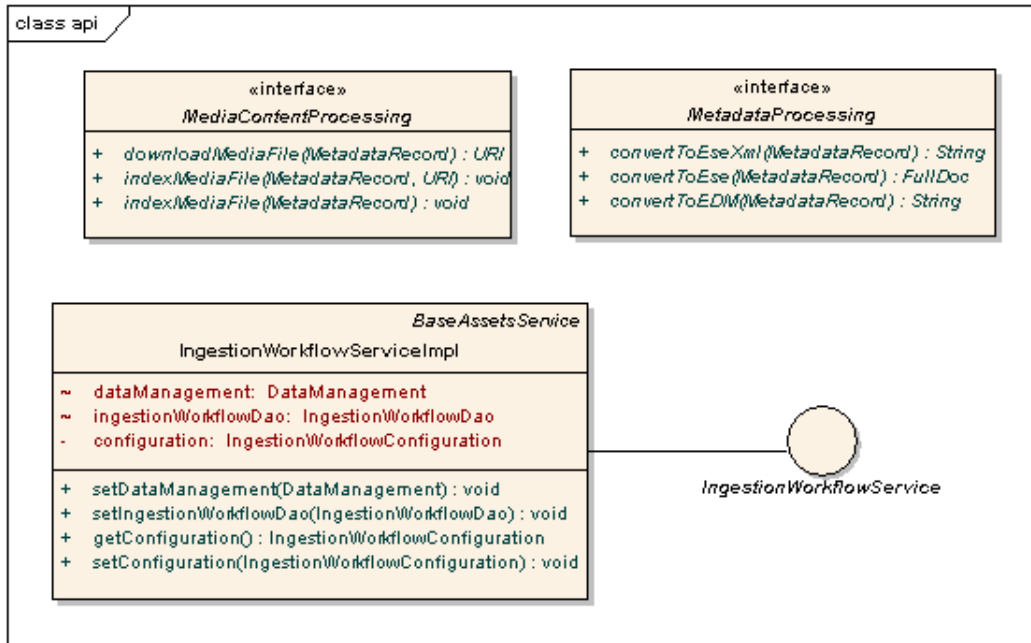


Figure 35 – Ingestion Workflow API model

## 4.4 The Indexing, Ranking and Retrieval Models and Interfaces

### 4.4.1 Post-query processing Models and Interfaces

**Query suggestion** helps the user to either better specify the information he/she is looking for, or helps him in browsing semantically related concepts. For instance, given the query "Pablo Picasso", possible interesting suggestions are "Pablo Picasso blu's period" (specification) or "cubism" (related concept).

Service Name	Query Suggestion service
Responsibility	1. Query Suggestion
Provided Interfaces	1. Suggest
Dependencies	ASSETS Common, ASSETS Core, Query Logs, BM25F

Interface Name	Suggest
Key Concepts	Queries, Shortcuts, Ranking
Operations	<ul style="list-style-type: none"> <li>getSuggestions</li> </ul>

Class diagram of the domain objects used by the service. *Suggestions* contains a set of queries suggested to the user.



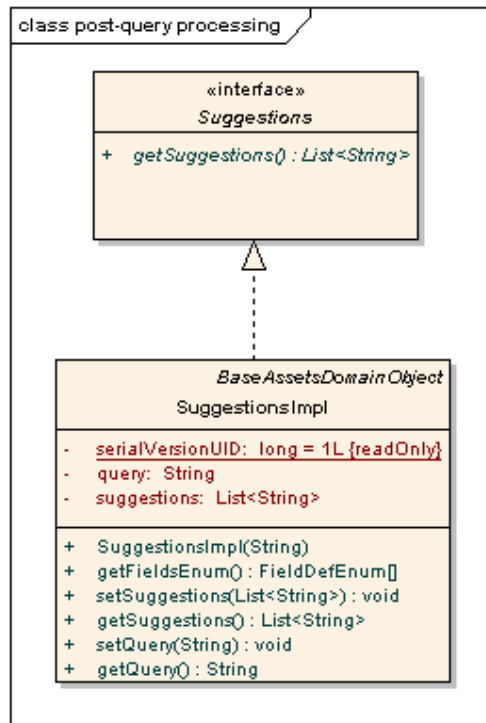


Figure 36 – Post-Query Processing : Query Suggestion model

The service gets a user query and returns a set of suggested queries (method getSuggestion())

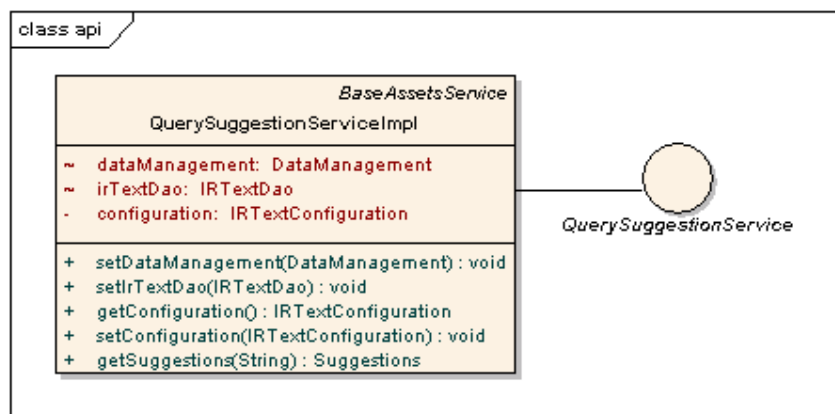


Figure 37 - BM25F Scoring function Service model

#### 4.4.2 Metadata Based Ranking Models and Interfaces

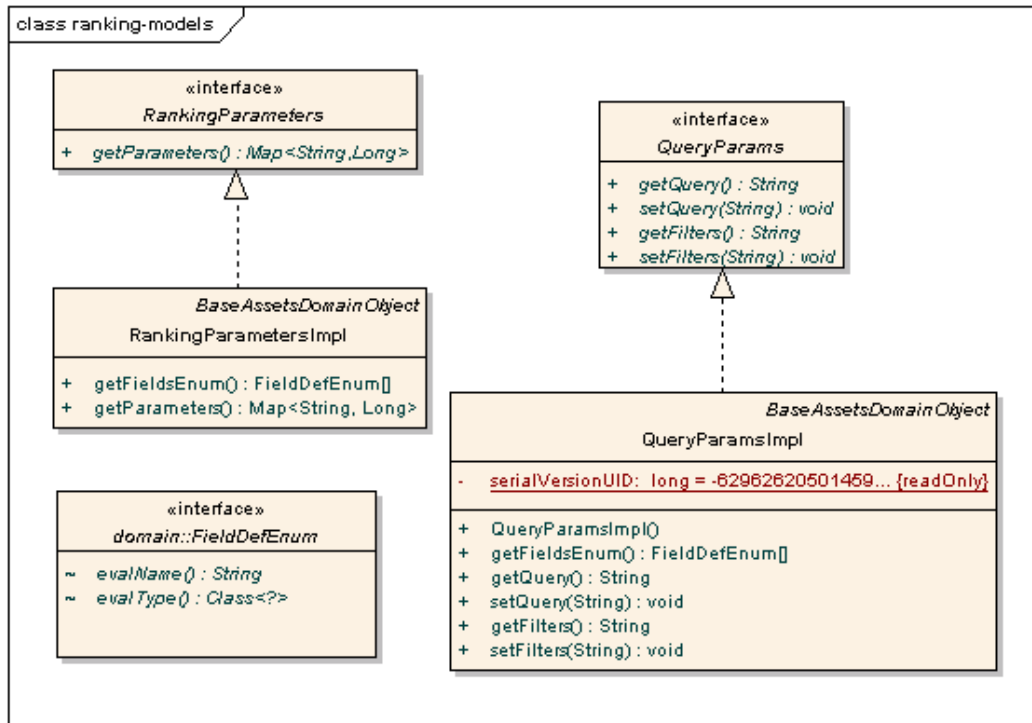
Full text search returns results (mostly) based on frequency of query terms in the collection and in the returned documents. This approach doesn't perform the best in the context of Europeana, where the indexed information is not a large document but a concise, well structured metadata record. The objective of this service is to exploit the structure present in Europeana metadata objects in order rank search results according to the user interests.

Service Name	BM25F Scoring function
Responsibility	<ol style="list-style-type: none"> <li>1. Search &amp; Retrieval</li> <li>2. Learn from query logs BM25F's parameters</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. BM25F</li> </ol>
Dependencies	ASSETS Common, ASSETS Core, Query Logs

Interface Name	BM25F
Key Concepts	Queries, Learning to rank, Ranking, QueryLogs
Operations	<ul style="list-style-type: none"> <li>• search</li> <li>• learning to rank</li> </ul>

**Class diagram of the domain objects used by the service.**

QueryParams Contains the query submitted by the user, together with the filters on the query (using solr syntax) and the result page requested.



**Figure 38 – Ranking models: BM25F Scoring function Service**

The service allows to perform a search using the bm25f scoring function (search method) and returns a list of AssetsFullDoc. Furthermore, the service exposes a method to retrieve a good tuning for the bm25f parameters.

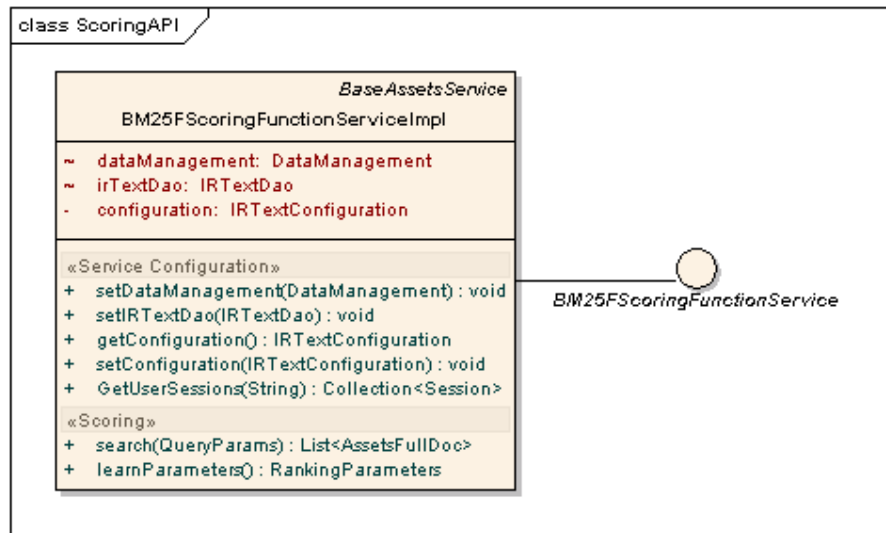


Figure 39 –BMF25 Scoring function Client

#### 4.4.3 Text Indexing and Retrieval Service

Europeana and Assets have many software components that exploit textual information. The goal of this service is to provide efficient access to structured information to end users and to other assets services, with particular reference to T2.2.1 and T2.2.2.

Service Name	Query Log Indexing
Responsibility	1. Cleaning and indexing of query log information for learning
Provided Interfaces	1. BuildIndex 2. GetUserSession 3. GetQueryPopularity
Dependencies	ASSETS Common, ASSETS Core, Query Logs

Interface Name	QueryLogIndexing
Key Concepts	Session Detection, Data cleaning
Operations	Analysis and indexing of query log index

The class diagram of the domain objects used by the service is presented in Figure 40.

- **QueryLogRecord interface** describes the object that models a record in the query log. It represents a user interaction with the portal (submitting a query, clicking on a results, navigating..).
- **Session** groups a the connected search activities of the current user. A session is a sequence of queries by a single user made within a limited range of time. It tries to



capture a single user's information need. Queries by the same user are divided in different sessions limiting the time gaps in a session (e.g., if a user does not search anything within an interval of five minutes, a new session starts).

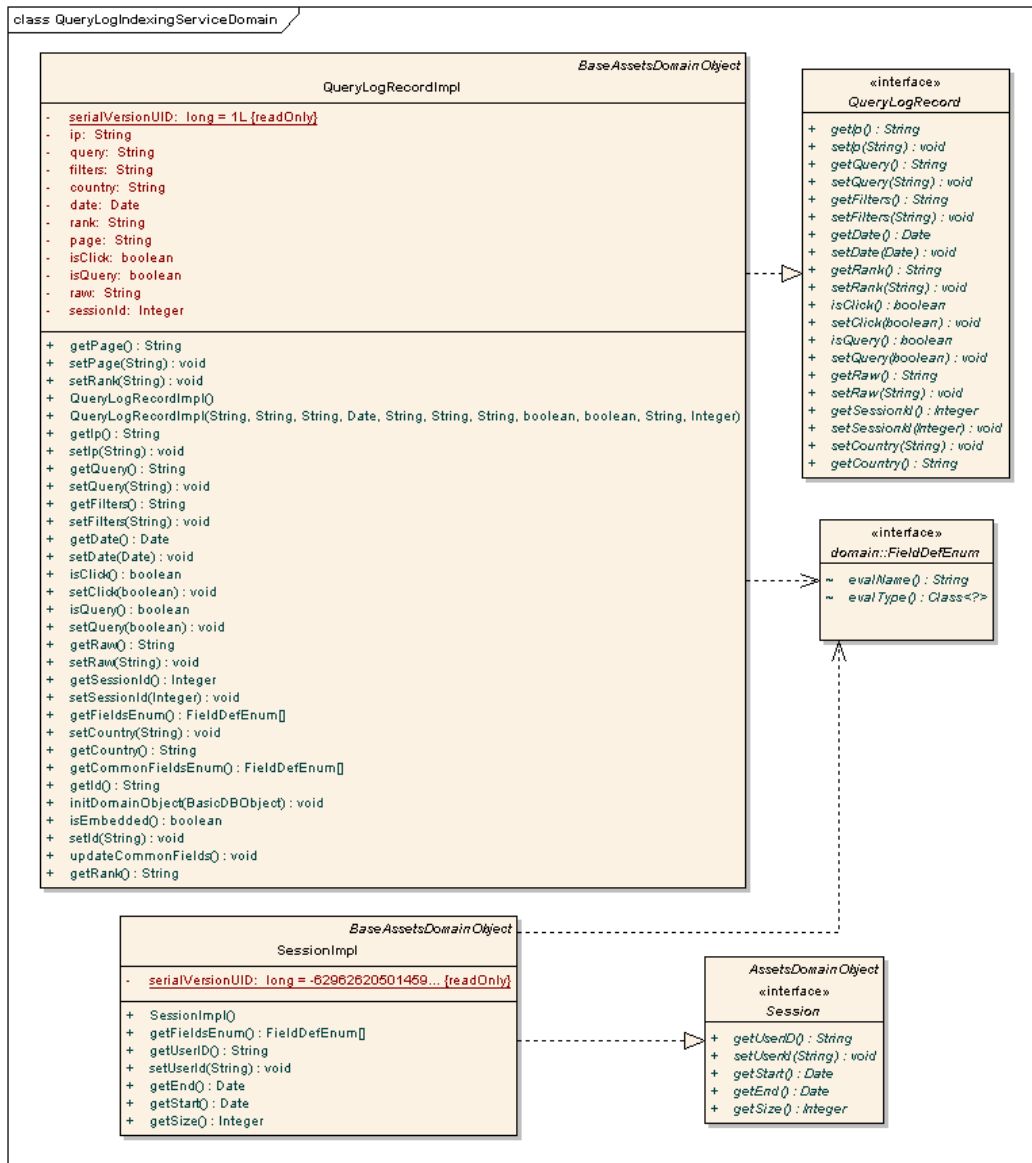


Figure 40 – Query Log Analysis Domain model

The class diagram for the Query Log Indexing Service is presented in Figure 41. It takes the query logs and splits them into user sessions. Furthermore it computes relevant statistics and other parameters used by the query suggestion service. The functionality of these classes is listed in the followings:

- `initIndex` and `insertQueryLog` allow to create a newIndex, and to add new query log.
- `getUserSessions`, `getNumberOfDistinctUsers`, `getNumberOfSessions`, `getNumberOfQueries`, `getNumberOfDistinctQueries`, `getAverageQueryLength`, `getTopQueriesWithFrequencies`, `getNumberOfSessionsForDay`,



getNumberOfSessionsForHour, getSession , getUserSessionIds. Allow to retrieve statistics on the indexed query logs (also filtering on the date)

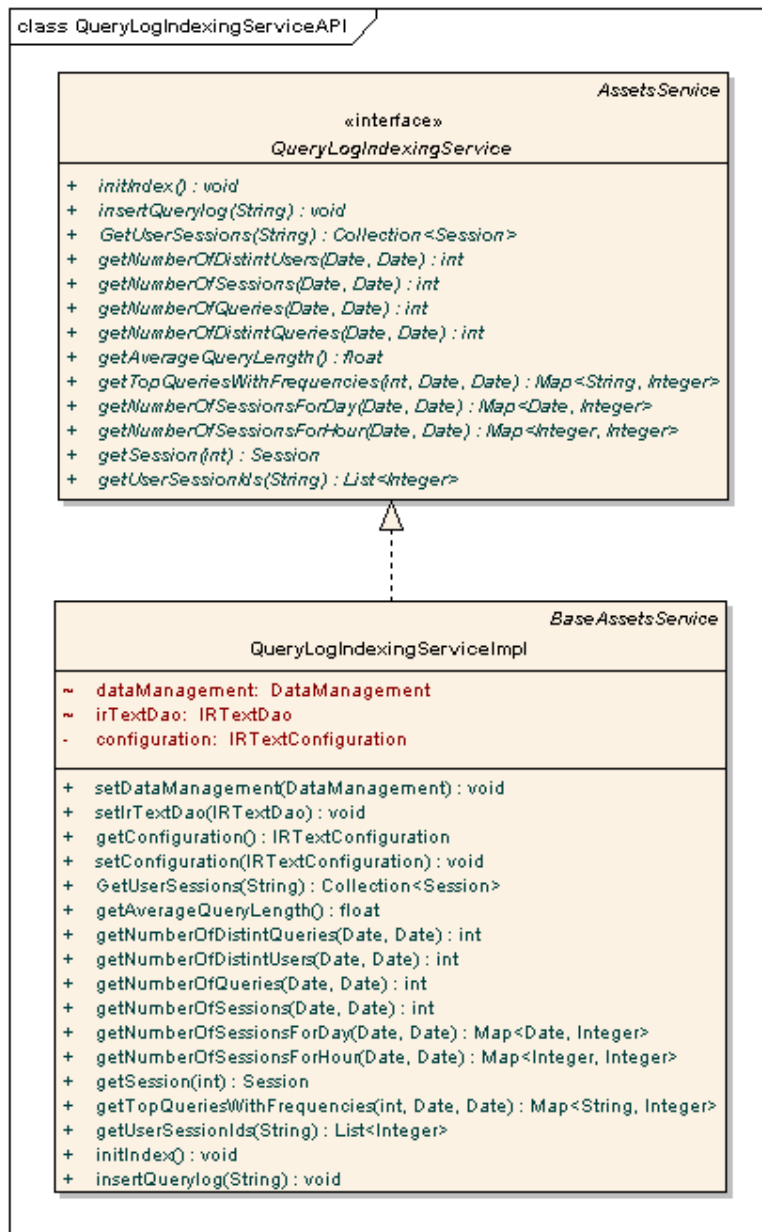


Figure 41 – Query Log Indexing API model

The class diagram for the Query Log Indexing Client is presented in Figure 42. It presents the client API used for remote invocation of the service’s REST interface.

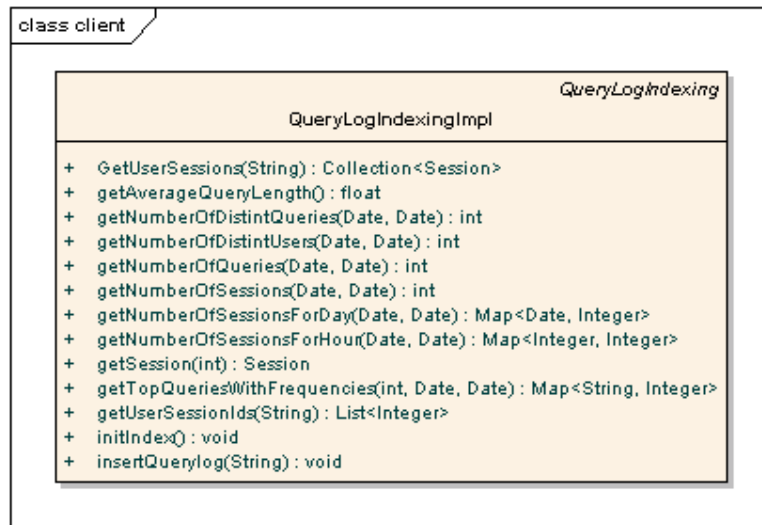


Figure 42 – Query Log Analysis Client model

#### 4.4.4 Images Indexing and Retrieval Service

Images Indexing and Retrieval Service allows users to find similar images using an image as query (image content-based searching).

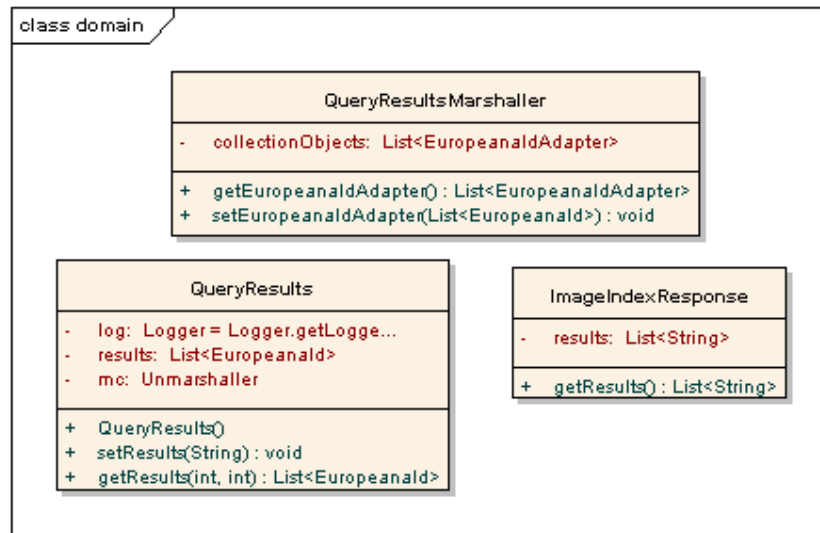
Service Name	Images Indexing and Retrieval Service
Responsibility	1. Search & Retrieval
Provided Interfaces	1. ImageIndexing 2. ImageSearching
Dependencies	ASSETS Common

Interface Name	ImageIndexing
Key Concepts	ImageID
Operations	Image features extraction and image indexing

Interface Name	ImageSearching
Key Concepts	ImageID, imageRanking
Operations	Performs an image content based similarity search from a given image ID, returning a list of image IDs ranked by similarity

The class diagram of the domain objects used by the service is presented in Figure 43. The functionality of these classes is listed in the followings:

- QueryResults contains the query results with the most similar images to the query, as a list of Europeanald.
- QueryResultsMarshaller serializes the query results in a list of EuropeanaldAdapter objects (Rest Service purpose).
- ImageIndexResponse contains the query results as a list of EuropeanUri.



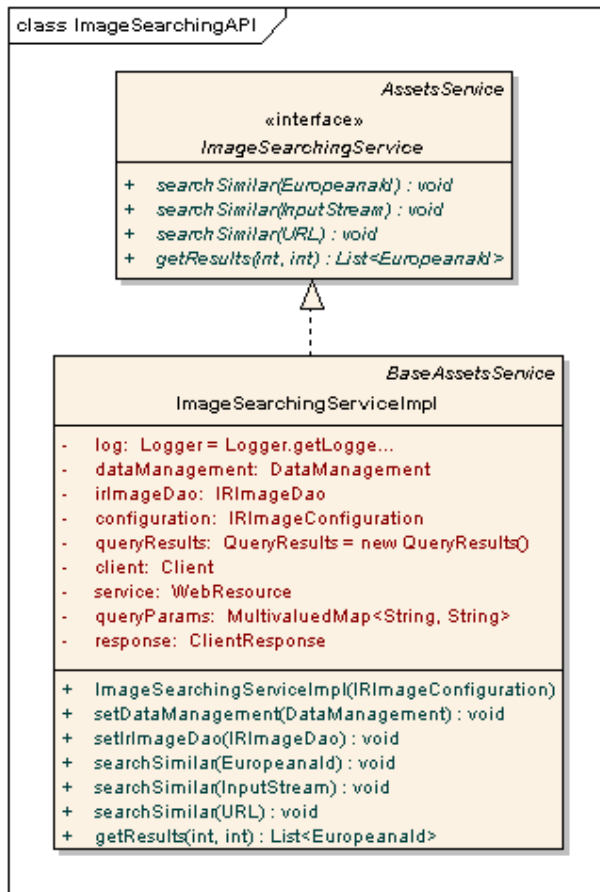
**Figure 43 – Images Indexing and Retrieval: Domain model**

The class diagram of the Image Searching Service is presented in Figure 44.

In this diagram are shown the available methods to perform an image similarity search. It is possible to perform a search by

- an image id (Europeanald);
- the stream of an image;
- an image URL.

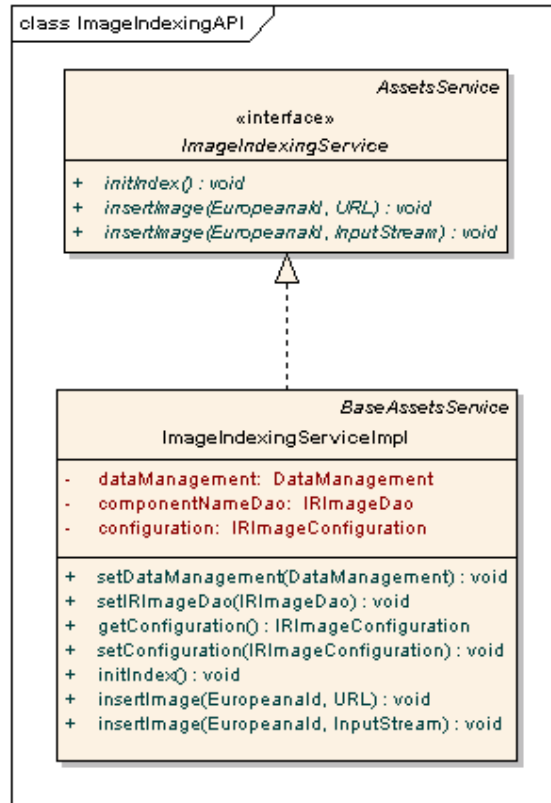
Then, calling `getResult` the system returns the results of the query ranked by similarity.



**Figure 44 –Images Indexing and Retrieval: Serching Service model**

In this diagram are shown the available methods to create and insert images into the image similarity search index. The `initIndex()` method creates a new image index; it destroys the previous index (if any) to build a new one. Then, by calling the `insertImage()` method the index can be populated, by inserting images by their URLs or by their streams.



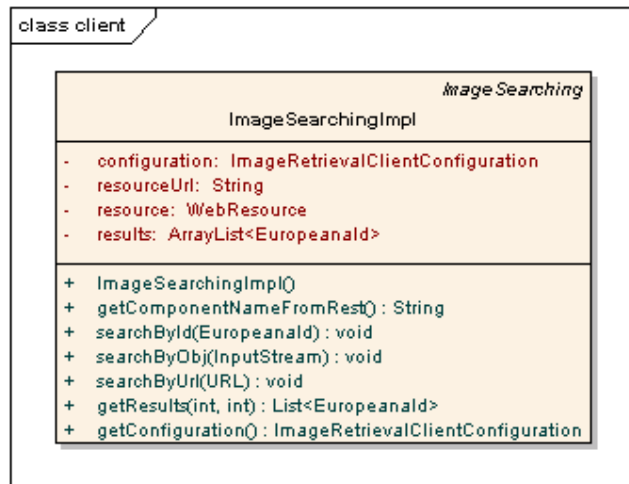


**Figure 45 – Images Indexing and Retrieval: Indexing Service model**

In this diagram are shown the available client methods used to perform an remote image similarity search. This allows to perform a search by

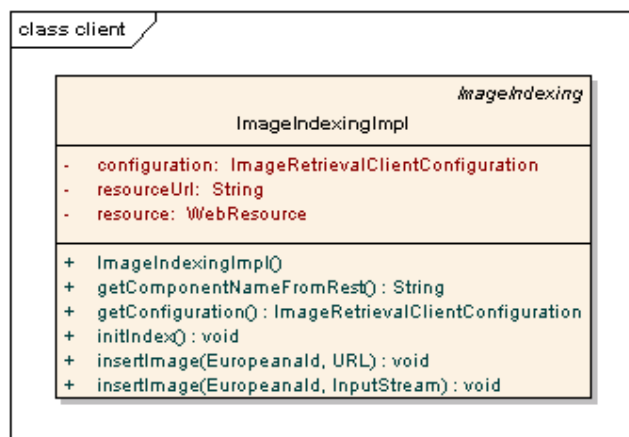
- an image id (EuropeanId);
- the stream of an image;
- an image URL.

Then, calling `getResults` the system returns the results of the query ranked by similarity.



**Figure 46 – Images Indexing and Retrieval: Retrieval Client Model**

In this diagram are shown the client side methods used to create and insert images into the image similarity search index. `initIndex` is the method to call to create a new image index; it destroys the previous index (if any) to build a new one. Then, calling `insertImage` the index can be populated, by inserting images by their URLs or by their streams.



**Figure 47 - Images Indexing and Retrieval: Indexing Client Model**

#### 4.4.5 3D-Model Indexing and Retrieval Services

The 3D Model Indexing and Retrieval Services allow users to search for 3D models geometrically similar to a query 3D model (uploaded or returned from the search) or a hand-drawn sketch.

Service Name	3D Low-level feature extraction service
Responsibility	1. Feature extraction
Provided Interfaces	1. Extraction3D
Dependencies	ASSETS Common

Interface Name	Extraction3D
Key Concepts	LowLevelFeatureVector
Operations	<ul style="list-style-type: none"> <li>Extraction of Low Level Features</li> </ul>

Service Name	3D Indexing service
Responsibility	1. Indexing
Provided Interfaces	<ol style="list-style-type: none"> <li>InitIndex</li> <li>InsertIntoIndex</li> <li>ClearIndex</li> </ol>
Dependencies	ASSETS Common, 3D Low-level feature extraction service

Interface Name	InitIndex
Key Concepts	ObjectID
Operations	<ul style="list-style-type: none"> <li>3D index initialization</li> </ul>

Interface Name	InsertIntoIndex
Key Concepts	ObjectID
Operations	<ul style="list-style-type: none"> <li>Insertion of a new object into the index</li> </ul>

Interface Name	ClearIndex
Key Concepts	
Operations	<ul style="list-style-type: none"> <li>3D index deletion</li> </ul>

Service Name	3D Search & retrieval service
Responsibility	1. Search & Retrieval
Provided Interfaces	<ol style="list-style-type: none"> <li>SearchingUploaded</li> <li>SearchingSelected</li> <li>SearchingSketched</li> </ol>
Dependencies	ASSETS Common, 3D Low-level feature extraction service

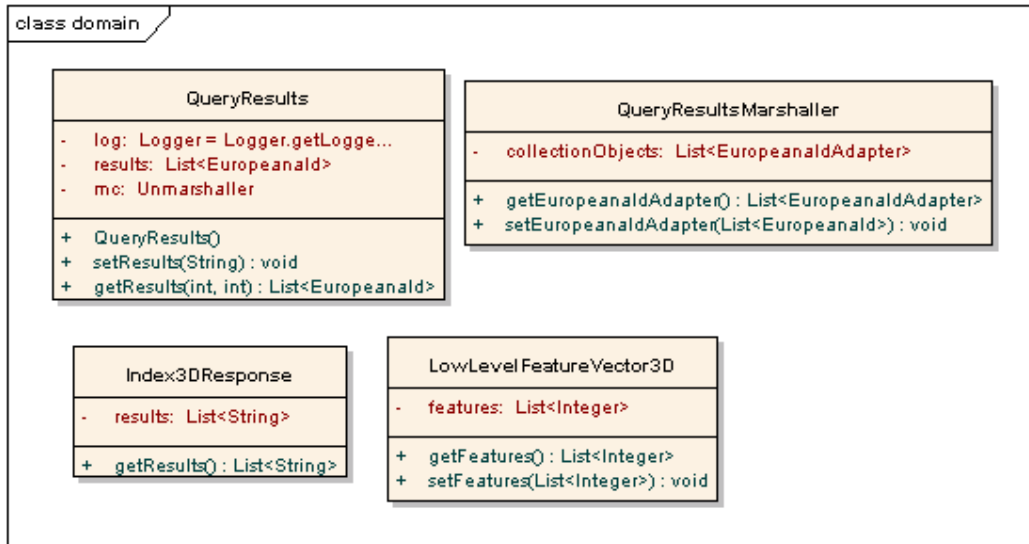
Interface Name	SearchingUploaded
Key Concepts	LowLevelFeatureVector, Ranking3D
Operations	<ul style="list-style-type: none"> <li>Search for objects similar to uploaded object</li> </ul>

Interface Name	SearchingSelected
Key Concepts	ObjectID, Ranking3D
Operations	<ul style="list-style-type: none"> <li>Search for objects similar to selected object</li> </ul>

Interface Name	SearchingSketched
Key Concepts	LowLevelFeatureVector, Ranking3D
Operations	<ul style="list-style-type: none"> <li>Search for objects similar to sketched object</li> </ul>

**The class diagram of the domain objects used by the 3D services is presented in Figure 48.**

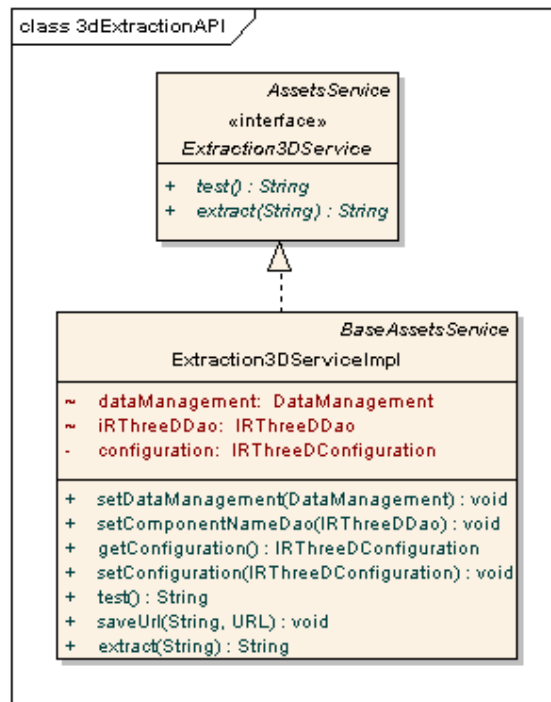
- LowLevelFeatureVector3D contains the results of the 3D of the extraction service, as a list of Integer.
- QueryResults contains the query results with the most similar 3D models to the query, as a list of EuropeanId.
- QueryResultsMarshaller serializes the 3D search results in a list of EuropeanIDAdapter objects (Rest Service purpose).
- ImageIndexResponse contains the 3D search results as a list of EuropeanUri.



**Figure 48 - 3D Domain Model**

The class diagram of the 3D Extraction Service is presented in Figure 49.

The service performing the extraction of 3D low-level features is shown. The extract method passes the url of a 3D model to the extractor; the results are returned in a XML file.

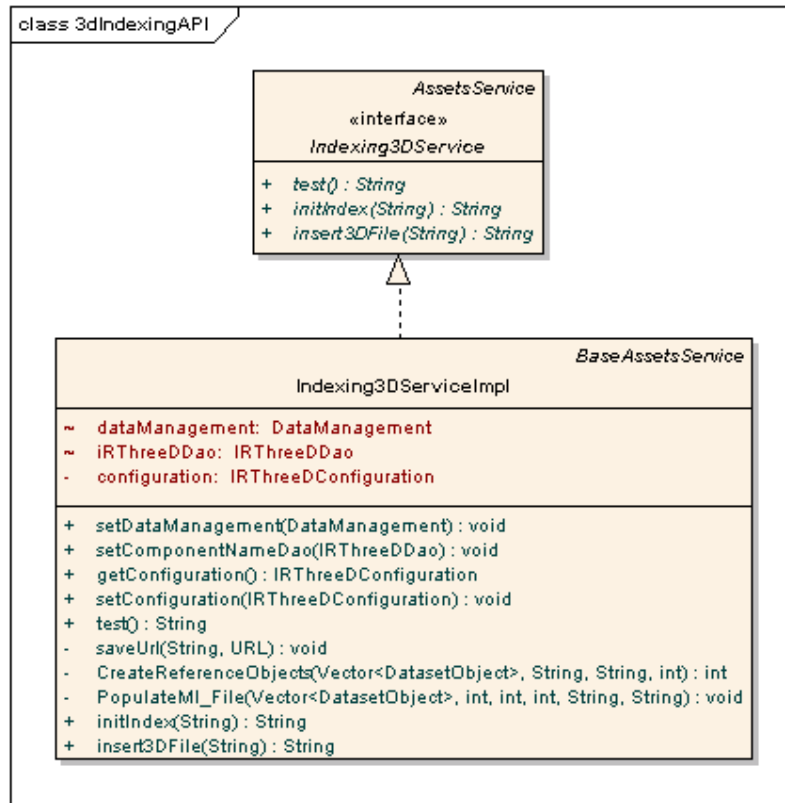


**Figure 49 – 3D Extraction Service model**

The class diagram of the 3D Indexing Service is presented in Figure 50.

The methods performing the creation and update of the 3D index are shown. The `initIndex` method creates a new 3D index. The invocation of `insert3DFile` method will populate the

index by passing the URL of a new 3D model.

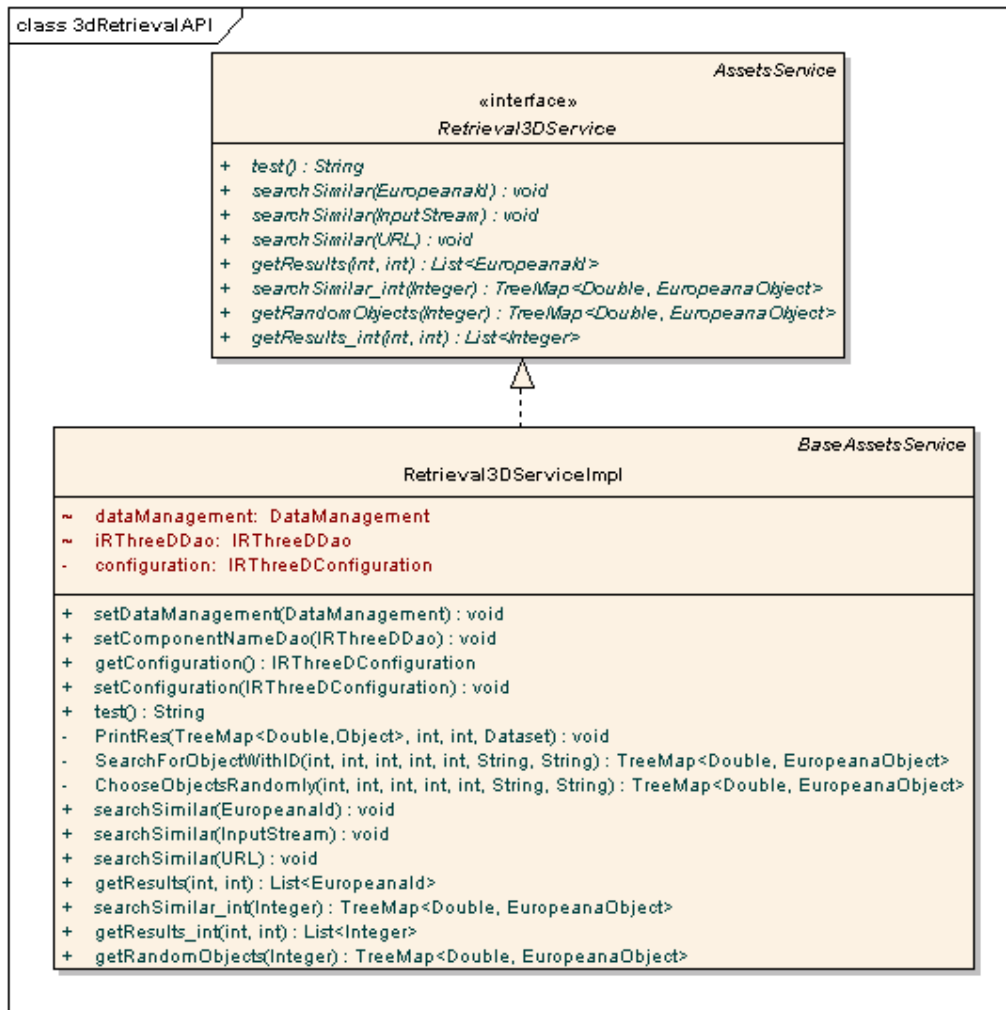


**Figure 50 - 3D Indexing Service model**

The class diagram of the 3D Retrieval Service is presented in the Figure 51. The methods performing a 3D similarity search are shown. The following query forms are supported:

- a 3D model id (EuropeanId);
- the stream of a 3D model;
- a 3D model URL.

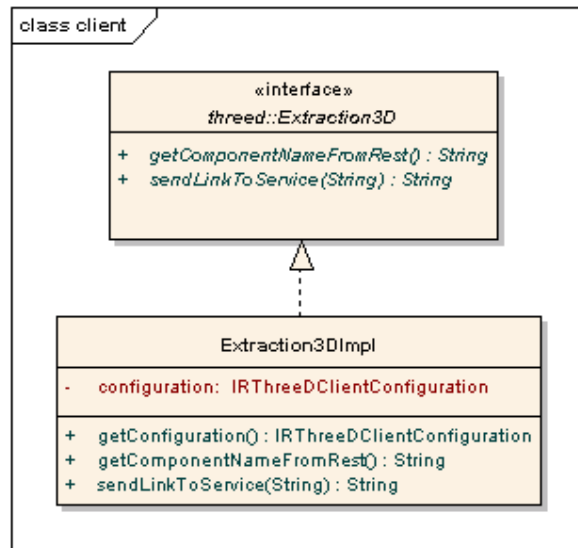
The results, ranked by similarity, are returned by calling the `getResults` method.



**Figure 51 - 3D Retrieval Service Model**

The class diagram of the 3D Extraction Client is presented in Figure 52.

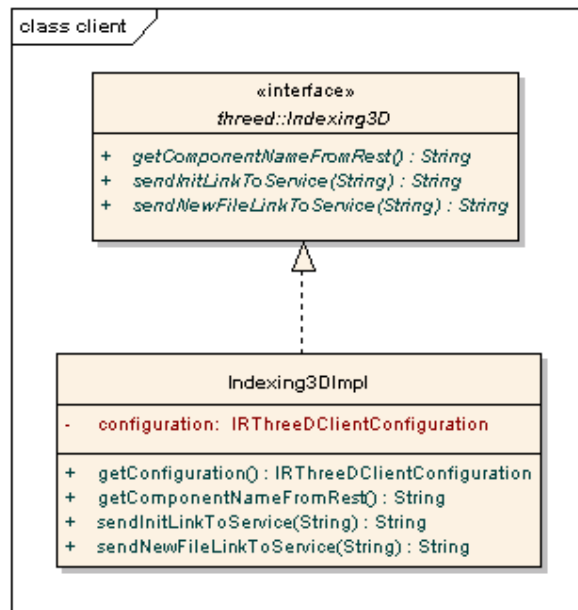
The methods to test the 3D feature extraction are shown. The `sendLinkToService` method sends the 3D model file URL to the extractor and receives the URL of an XML file, containing the results of the feature extraction process.



**Figure 52 - 3D Extraction client model**

The class diagram of the 3D Indexing Client is presented in Figure 53.

The methods to test the 3D indexing are shown. The `sendInitLinkToService` method sends the URL of a file containing 3D model to the index creator. The `sendNewFileLinkToService` method sends the URL of a 3D model to the index updater.



**Figure 53 - 3D Indexing client model**

The class diagram of the 3D Retrieval Client is presented in Figure 54. The methods for remote invocation of the 3D retrieval service are shown. The following query forms are supported:

- a 3D model id (EuropeanId);



- the stream of a 3D model;
- a 3D model URL.

When calling `getResults` method, the system returns the results of the 3D search ranked by similarity.

#### 4.4.6 Audio Indexing and Retrieval Service

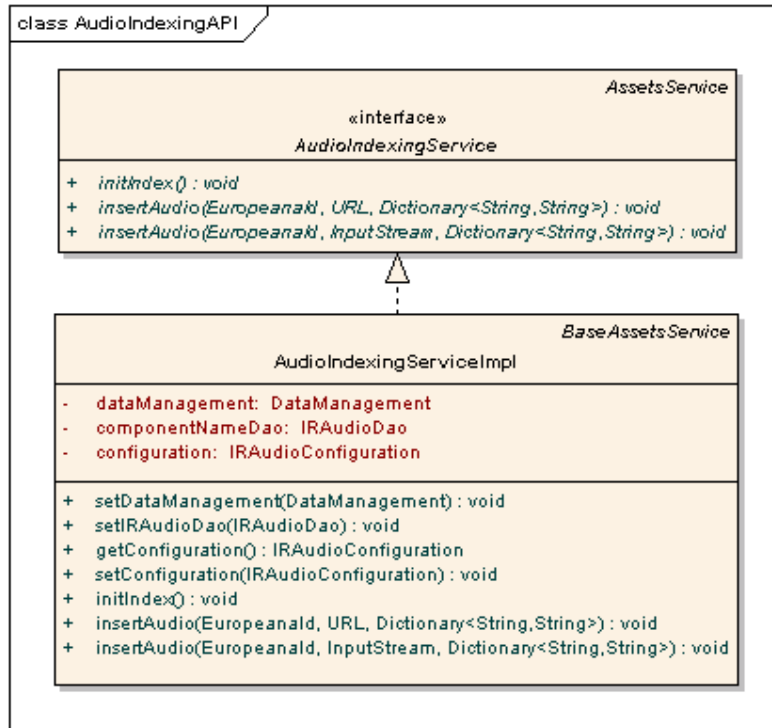
The audio indexing and retrieval service provides advanced music search and recommendation functionalities. It enhances Europeana by enabling end users to discover audio media based not only in textual similarity but based on audio content and context similarities.

This service provides two interfaces: one for inserting audio and the other for searching amongst the indexed one.

Service Name	Audio Indexing And Retrieval
Responsibility	1. Indexing and search of music based on content-based as well as contextual similarity
Provided Interfaces	1. <code>AudioIndexingService</code> 2. <code>AudioSearchingService</code>
Dependencies	ASSETS Common

The `AudioIndexingService` interface allows kick starting the content based analysis, the context enrichment and the actual indexation within the audio dataset. It requires an audio identifier, audio metadata (in order to match it against the context information for enrichment) and the audio url for audio feature extraction.

Interface Name	<code>AudioIndexingService</code>
Key Concepts	Music features extraction and audio indexing
Operations	<ul style="list-style-type: none"> <li>• <code>insertAudio</code> - Inserts audio based on Audio ID, Audio Metadata and Audio URL</li> </ul>



**Figure 54 – Audio Indexing and Retrieval: InsertAudio model**

AudioSearchingService interface enables audio similarity searches, mood based searches (or high level musical feature) or hybrid search. The hybrid search uses similarity search filters for audio descriptors.

Interface Name	AudioSearchingService
Key Concepts	Audio Similarity Search
Operations	<ul style="list-style-type: none"> <li>searchSimilar - Audio Similarity search based on Audio ID. Optionally, search can be filtered by mood or other high level musical feature.</li> </ul>

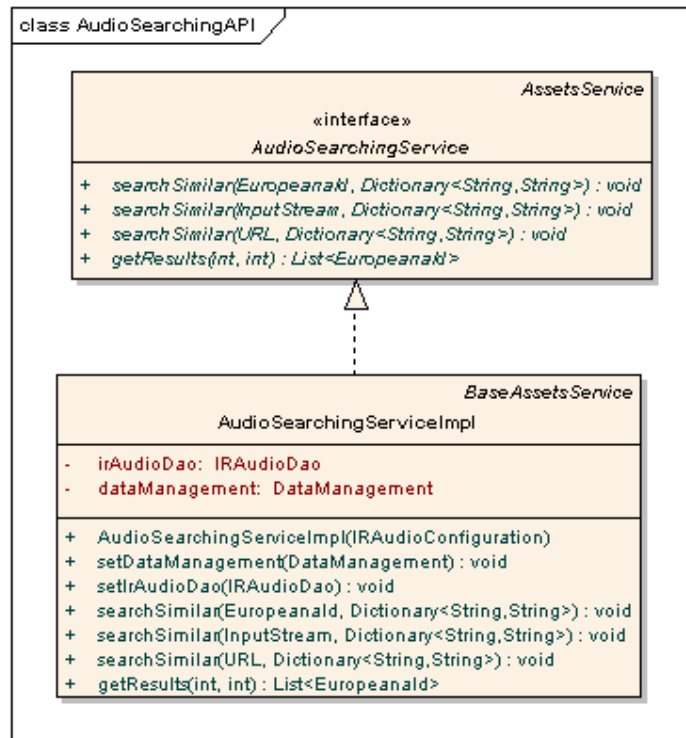


Figure 55 - Audio Indexing and Retrieval: SearchAudio model

#### 4.4.7 Video summarisation, indexing and retrieval

The goal of this service is to enhance the functionalities of Europeana for searching, browsing, previsualizing and accessing video content, which is a time consuming activity due to the high amount of information available in media (video) files.

Service Name	Video Summarization
Responsibility	1. Generation of reduced length versions of original videos and extraction of representative keyframes
Provided Interfaces	1. VideoSummarizationService 2. SummarizedVideo
Dependencies	ASSETS Common

Interface Name	VideoSummarizationservice
Key Concepts	Europeanald
Operations	<ul style="list-style-type: none"> <li><b>getSummarizedVideo()</b> - returns a SummarizedVideo object from which the video summary and keyframes can be obtained. The caller can optionally indicate the desired percentage for the video summary.</li> </ul>

Interface Name	SummarizedVideo
Key Concepts	URL, KeyFrame
Operations	<ul style="list-style-type: none"> <li>• <b>getOriginalVideo()</b> - returns the original video's EuropeanId</li> <li>• <b>getSummarizedVideo()</b> - returns the URL of the summarized video. This method receives an optional parameter with the percentage of the original video length to generate in the video summary.</li> <li>• <b>getKeyFrames()</b> - returns a collection of KeyFrame objects according to the percentage that was indicated for the video summary. Each KeyFrame instance is a representative picture of the video along with its timestamp in the original video.</li> </ul>

Service Name	Video Indexing and Retrieval
Responsibility	1. Indexing and search of videos based on visual similarity
Provided Interfaces	1. VideoIRService
Dependencies	ASSETS Common

Interface Name	VideoIRService
Key Concepts	EuropeanId
Operations	<ul style="list-style-type: none"> <li>• <b>indexVideo()</b> - calculate the visual similarity indexes of the video so that it can be compared to the indexes of other videos (for search purposes). The method does not return any information but stores the EuropeanId parameter and the indexes in our internal database.</li> <li>• <b>getVideoSimilarTo()</b> - returns a collection of EuropeanId objects corresponding to videos that are similar to the EuropeanId object received as parameter. This EuropeanId object can be either an image or a video. The visually similar videos that this operation returns are obtained from those ones that our module has previously analyzed and indexed (using indexVideo()).</li> </ul>

The next figures show the UML classes diagram of the domain object for the summarization service and the indexing and retrieval service.

The method `getSummarizedVideo()` of the `VideoSummarizationService` class is intended to create different `SummarizedVideo` objects. The same original video can be summarized several times with different values in the percentage parameter. Each instance of the `SummarizedVideo` class represents a summarization request. This data is associated to the `EuropeanId` object by means of the `getOriginalVideo()` method. The `getSummarizedVideo()` and `getKeyFrames()` methods do not create new `EuropeanId` instances but a URL or collection of `KeyFrame` objects, respectively. The `KeyFrame` class contains information for

the timestamp of the keyframe in the original video. The timestamp units are milliseconds and 0 means the beginning of the video. The KeyFrame class also stores the JPEG visual picture of the keyframe.

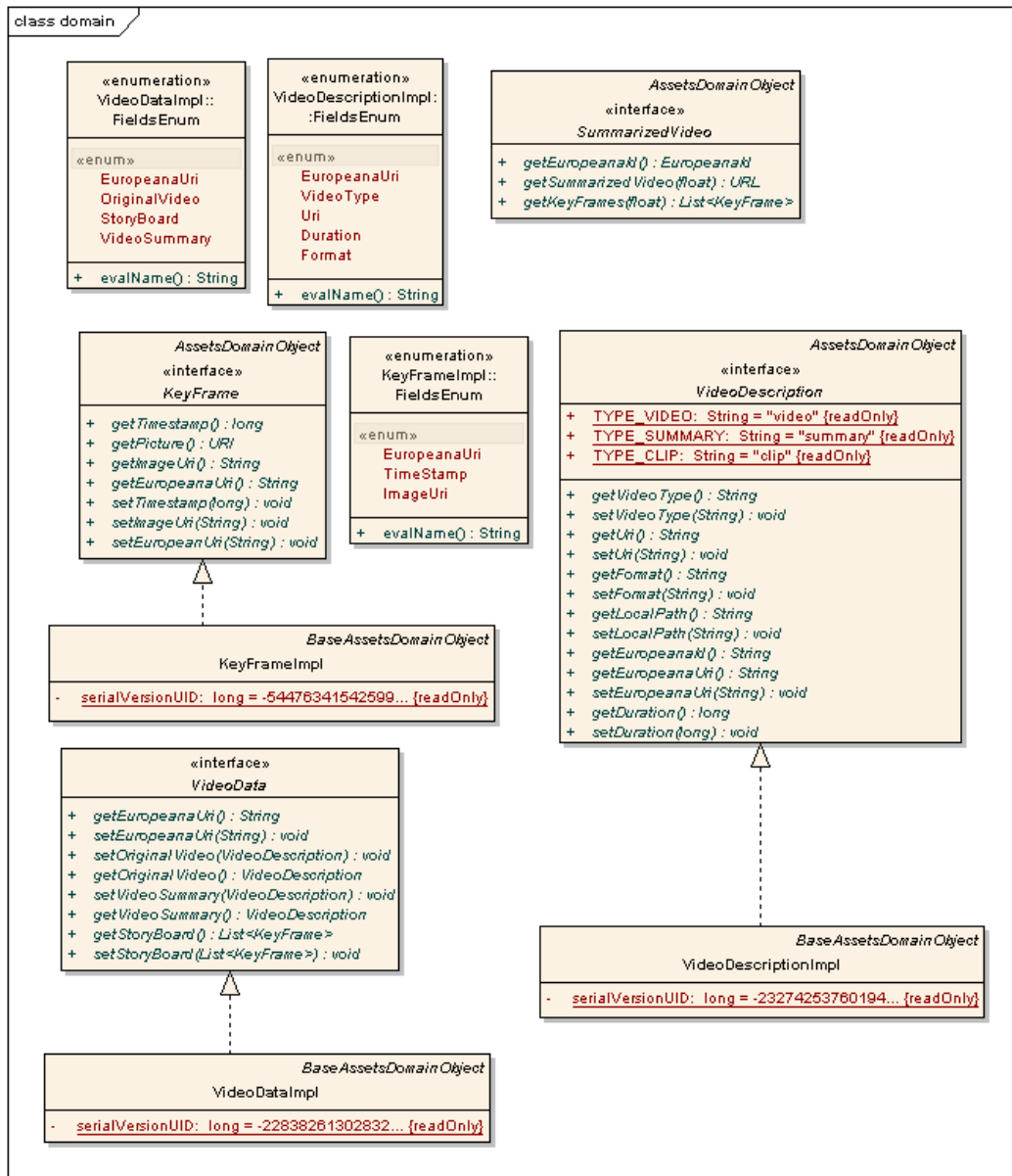
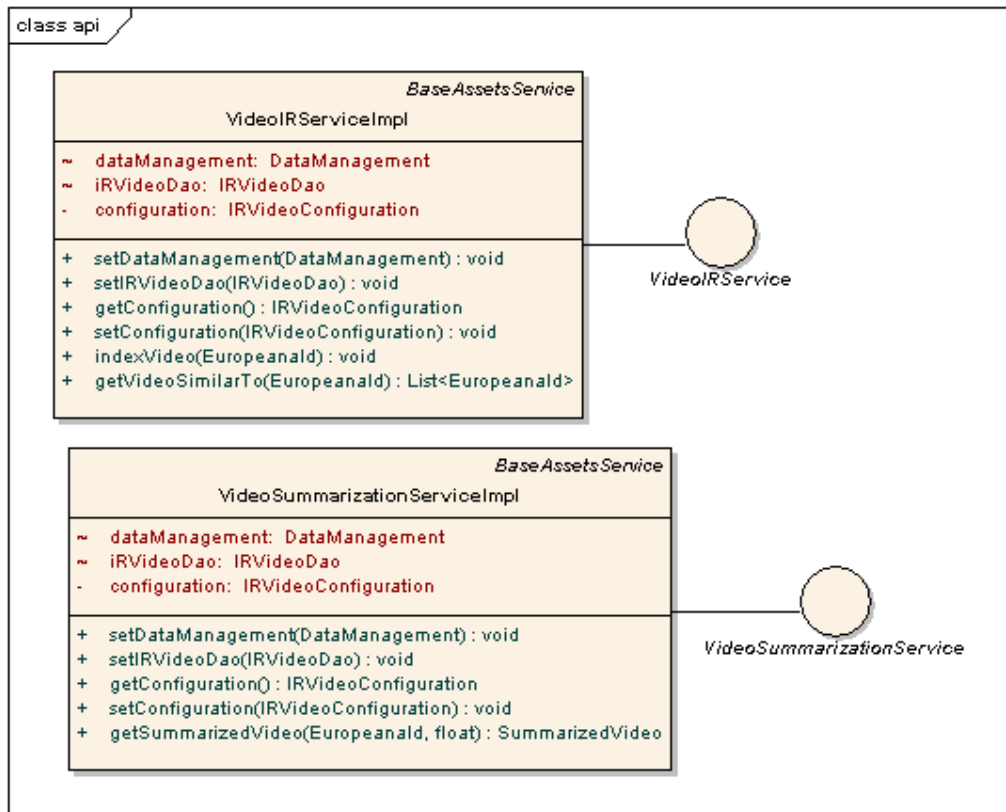


Figure 56 – Video Summarization Model

The method `getVideosSimilarTo()` of the `VideoIRService` class is intended to obtain a collection of visually similar videos (represented as instances of the `EuropeanId` class). The operation receives as parameter a `EuropeanId` object that can be either an image or a video. The visually similar videos that operation returns are obtained from those ones that the module has previously analyzed and indexed (i.e., the videos that were received through the `indexVideo()` method).



**Figure 57 – Video Summarization Service model**

The SummarizedVideo class has a protected constructor that prevents from other developers attending to instantiate it directly. This object always has to be instantiated throughout its factory object (i.e., VideoSummarizationService).

## 4.5 The Digital Preservation Models and Interfaces

### 4.5.1 The Risk Management Models and Interfaces

Service Name	Risk Management
Responsibility	<ol style="list-style-type: none"> <li>1. Content classification /collection profiling</li> <li>2. Risk analysis and rule based preservation plan recommendation</li> <li>3. Preservation watch</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. ContentClassification</li> <li>2. PreservationWatch</li> <li>3. RiskAnalysis</li> </ol>
Dependencies	ASSETS common, Normalisation Service, Notification Service

The diagram in Figure 58 represents the domain concepts used by the Risk Management Service:

- **CollectionAnalysisReport** - contains aggregated report information about the analyzed collection e.g. the total count of broken objects, their formats.
- **MetadataAnalysisResult** - comprises metadata analysis results like e.g. count of missing fields and broken URIs in europeana records
- **RiskAnalysis** - describes the underlying risk model which is specifically applied for a given record at hand to evaluate the individual risks per property as well as the overall weight between risk property sets.
- **RiskPropertySet** - is a container for risk properties and nested property sets;
- **RiskProperty** - is a leaf-node-concept which on the one hand contains static information on the property ID, version, query (e.g. how to query and evaluate the property's possible outcomes) but also contains the evaluated risk classification result.
- **RiskClassification** – contains a set of evaluated RiskFactors for a given object
- **RiskFactor** contains the actual domain knowledge of risk factors and rules for judging on risk properties in a given context. It defines rules with min and max limits, risk score and metrics for different data types.

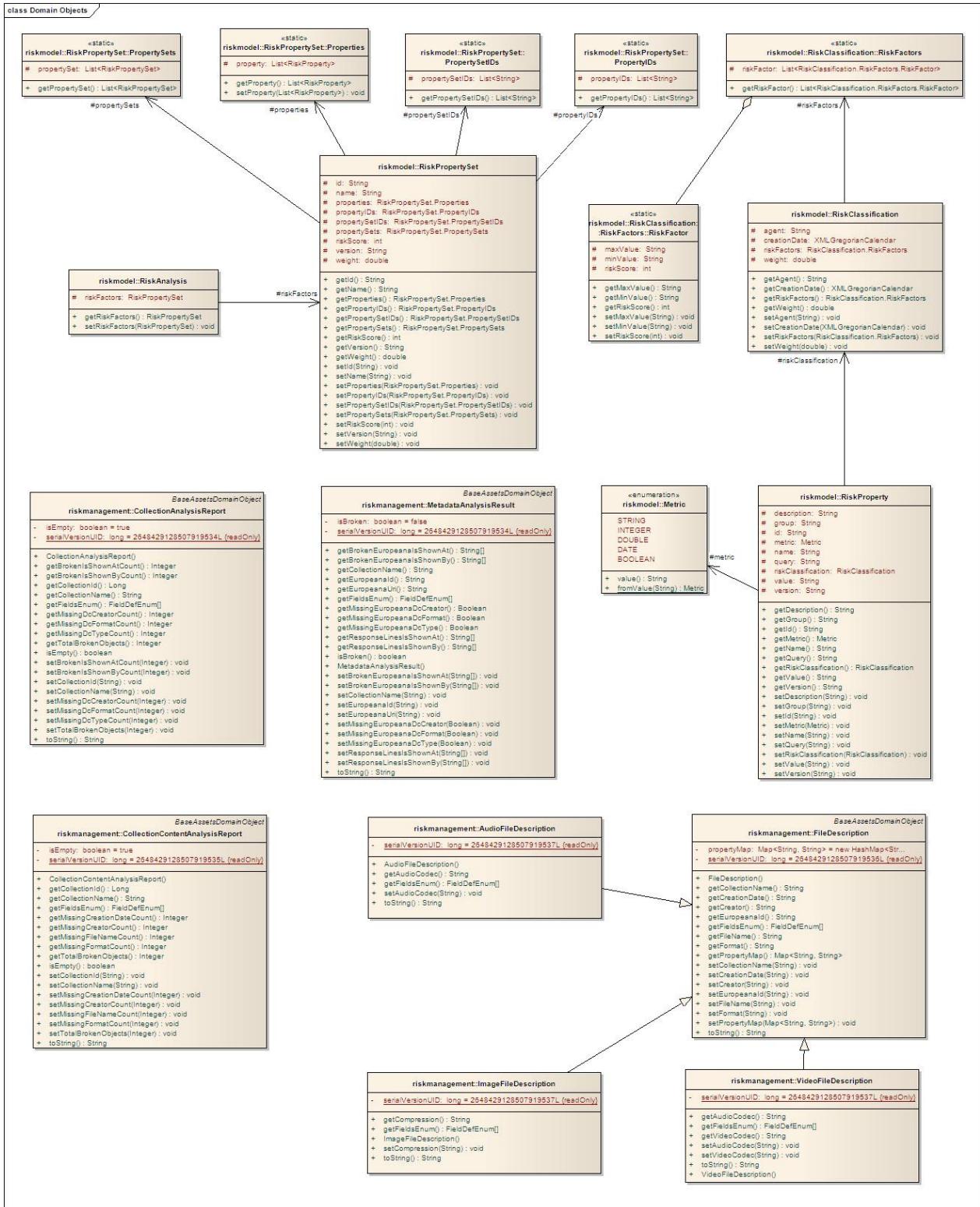


Figure 58 - Risk Management Service Domain Model



The Figure 59 briefly describes the most important interfaces and classes implementing server side business logic. The `PreservationRiskmanagementService` interface provides functionality needed to pass data received from client by REST services to the Risk Analysis Service. The `PreservationRiskmanagementServiceImpl` class contains actual implementation for corresponding analysis calculations.

The `PreservationRiskmanagementDao` interface provides the database connection including store and retrieve methods for different analysis types

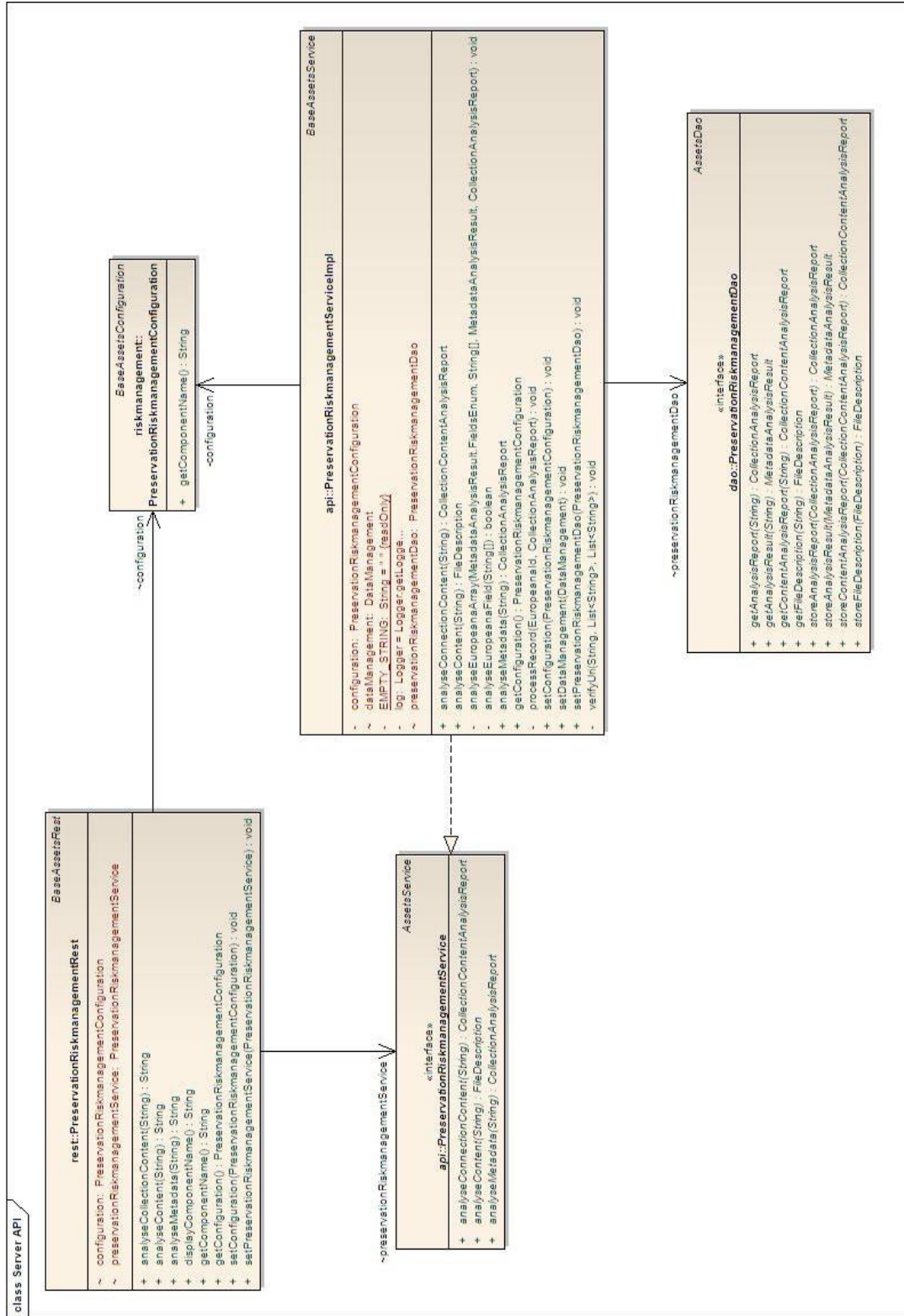


Figure 59 – Overview of Risk Management Interfaces

Interface Name	PreservationRiskmanagementService
Key Concepts	RiskAnalysisReport
Operations	<ul style="list-style-type: none"> <li>• <b>CollectionAnalysisReport analyseMetadata(String id)</b> - this method analyzes the metadata of Europeana objects for a particular collection identified by passed id;</li> <li>• <b>CollectionContentAnalysisReport analyseConnectionContent(String id)</b> - this method generates response to client for particular Europeana collection identified by the given ID in form of “colection content analysis report” objects;</li> <li>• <b>FileDescription analyseContent(String id)</b> - this method generates response to client for particular Europeana collection object identified by passed ID.</li> </ul>

Interface Name	PreservationRiskmanagementDao
Key Concepts	RiskAnalysisReport, StoreReport, RetrieveReport
Operations	<ul style="list-style-type: none"> <li>• <b>CollectionAnalysisReport getAnalysisReport(String id)</b> - this method retrieves collection analysis report from database by passed object id;</li> <li>• <b>CollectionAnalysisReport storeAnalysisReport(CollectionAnalysisReport collectionAnalysisReport)</b> - this method stores collection analysis report in database;</li> <li>• <b>MetadataAnalysisResult getAnalysisResult(String id)</b> - this method retrieves metadata analysis result from database by passed object id;</li> <li>• <b>MetadataAnalysisResult storeAnalysisResult(MetadataAnalysisResult metadataAnalysisResult)</b> - this method stores metadata analysis result in database;</li> <li>• <b>FileDescription getFileDescription(String id)</b> - this method retrieves file description from database by passed object id;</li> <li>• <b>FileDescription storeFileDescription(FileDescription fileDescription)</b> - this method stores file description in database;</li> <li>• <b>CollectionContentAnalysisReport getContentAnalysisReport(String id)</b> - this method retrieves collection content analysis report from database by passed object id;</li> <li>• <b>CollectionContentAnalysisReport storeContentAnalysisReport(CollectionContentAnalysisReport collectionContentAnalysisReport)</b> - this method stores collection content analysis report in database.</li> </ul>

Interface Name	ContentClassification
Key Concepts	SimpleClassificationProfile, ExtendedClassificationProfile, ProfilingConfiguration, Report, ExecutablePolicy, FeatureVectors
Operations	<ul style="list-style-type: none"> <li>• classifyContent</li> <li>• setProfilingWorkflow</li> <li>• setInspectionDepth</li> <li>• getReport</li> <li>• storeFeatureVectorsPerObject</li> </ul>

Interface Name	PreservationWatch
Key Concepts	PreservationWatchSource, FormatMonitor, Rules, FormatRiskScores, ServicePerformanceTests, FormalRecommendations
Operations	<ul style="list-style-type: none"> <li>• register,</li> <li>• deposit,</li> <li>• getFormatRiskScore,</li> <li>• getServicePerformanceTest,</li> <li>• getFormalRecommendation,</li> <li>• listAllFormatRiskScores,</li> <li>• listAllServicePerformanceTests,</li> <li>• listAllFormalRecommendations</li> </ul>

Interface Name	RiskAnalysis
Key Concepts	RiskReport, ExecutableNormalisationPolicy, RiskScores, Metrics, RecommendationMode, Reasoner
Operations	<ul style="list-style-type: none"> <li>• analyseBySample</li> <li>• analyseByMetadata,</li> <li>• configure</li> </ul>

Interface Name	RiskModel
Key Concepts	PropertySet XML, ClassificationXML
Operations	<ul style="list-style-type: none"> <li>• <b>RiskAnalysis</b> analyze(File file, String propertySetXml, String</li> </ul>



	<p><b>classificationXml)</b> - this method analyzes passed file using calculation model retrieved from passed property set XML and classification XML files. If XML files have no valid path - default files will be used;</p> <ul style="list-style-type: none"> <li>• <b>RiskAnalysis getRiskAnalysis()</b> - this method retrieves risk analysis structure;</li> <li>• <b>setRiskAnalysis(RiskAnalysis riskAnalysis)</b> - this method initializes a risk analysis structure;</li> <li>• <b>setProperty(RiskPropertySet set, String propertyId, String propertyValue)</b> - this method initializes property value.</li> </ul>
--	---

Interface Name	RiskScore
Key Concepts	RiskScore
Operations	<ul style="list-style-type: none"> <li>• <b>RiskPropertySet getRiskScoreBreakdown()</b> - returns a set of all RiskProperties that have been taken into account for the given data item profile to calculate the risk score;</li> <li>• <b>RiskPropertySet getRiskProperties()</b> - returns a set of all RiskProperties;</li> <li>• <b>Integer getRiskScore()</b> - returns the risk score of a given data item profile; risk score values are integers between 0 (no risk) and 100 (highest risk).</li> </ul>

Interface Name	Connector
Key Concepts	Request, Repository, Connection, Query
Operations	<ul style="list-style-type: none"> <li>• <b>String getQuery()</b> - this method returns base query for repository request;</li> <li>• <b>String getRiskPropertyIdentifier()</b> - this method returns a risk property identifier;</li> <li>• <b>String getSearchValue()</b> - this method returns search value; it is a request value for repository;</li> <li>• <b>List&lt;String&gt; getColumnNames()</b> - this method returns repository column names for particular request;</li> <li>• <b>setValue(String value)</b> - this method fills risk property with value retrieved from repository.</li> </ul>

Interface	RepositoryDescription
-----------	-----------------------



Name	
Key Concepts	Repository
Operations	<ul style="list-style-type: none"> <li>• <b>String getID()</b> - this is a unique repository ID;</li> <li>• <b>String getName()</b> - this is a repository name;</li> <li>• <b>URL getLocation()</b> - this is a URL to repository;</li> <li>• <b>String getProtocol()</b> - this is used protocol;</li> <li>• <b>String getUpdatePolicy()</b> - this is a update policy definition;</li> <li>• <b>String getNameSpace()</b> - this is repository name space.</li> </ul>

Interface Name	PreservationWatchSource
Key Concepts	PreservationWatch, Repository
Operations	<ul style="list-style-type: none"> <li>• <b>RepositoryDescription describe()</b> - this method retrieves a repository description;</li> <li>• <b>void update()</b> - this method retrieves data from repository</li> </ul>

The following diagram depicts the client side interfaces and classes used for remote invocation of risk management service.

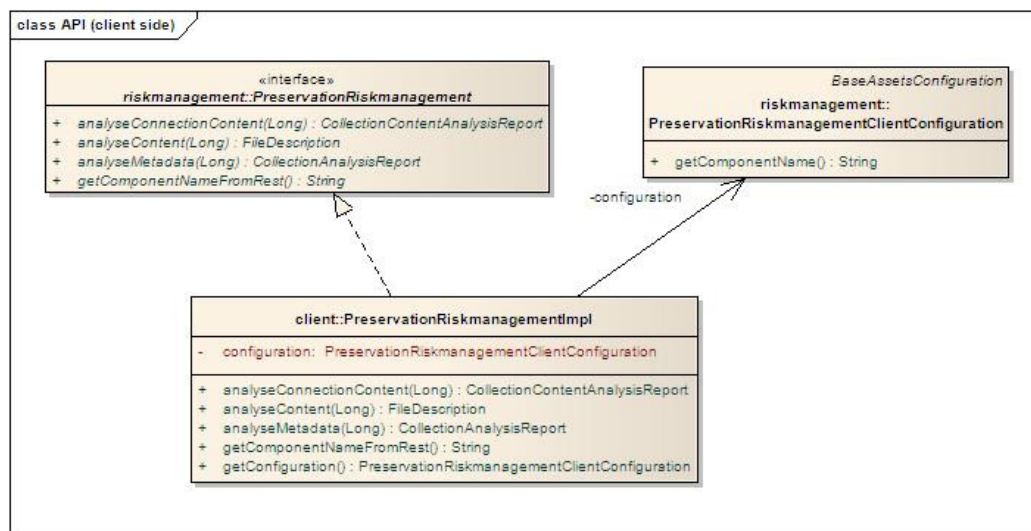


Figure 60 - Risk Management : Client Side Models

Interface Name	PreservationRiskManager
Key Concepts	Metadata Analysis
Operations	<ul style="list-style-type: none"> <li>• <b>CollectionAnalysisReport analyseMetadata(Long europeanId)</b> – allows to analyse broken URIs and missing preservation relevant fields in EuropeanID object.</li> <li>• <b>String getComponentNameFromRest()</b> - retrieves the component name from rest interface;</li> <li>• <b>CollectionContentAnalysisReport analyseConnectionContent(Long id)</b> - this method generates response to client for particular Europeana collection identified by the passed ID in form of collection content analysis report objects;</li> <li>• <b>FileDescription analyseContent(Long id)</b> - This method generates response to client for particular Europeana collection object identified by passed ID in form of content analysis report.</li> </ul>

#### 4.5.2 The Normalisation Models and Interfaces

Service Name	Normalisation
Responsibility	<ol style="list-style-type: none"> <li>1. Allows the workflow execution of preservation services on a specific object or a given collection;</li> <li>2. Design of Normalisation tasks;</li> <li>3. Administration and deployment of preservation services</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. WorkflowRegistry,</li> <li>2. ServiceRegistry,</li> <li>3. WorkflowExecutionManager</li> </ol>
Dependencies	ASSETS common, Notification

The diagram in Figure 61 represents the major normalisation data domain objects used by the service: Every service declares a set of parameters that it's able to cope with (e.g. the compression type, quality or algorithm to apply) to allow fine-tuning of the service's underlying tool. Data records that are produced and exchanged between services are wrapped up in a repository independent data abstraction called Planets Digital Object, which roughly speaking can be mapped to the PREMIS (PREservation Metadata: Implementation Strategy) notion of a representation object. It is composed of one bit-stream, associated to repository specific metadata and in addition is designed to sufficiently express the preservation process and its results as events, manifestations and properties.

- **DigitalObject** - A representation of a digital content object. Instances are created using a builder to allow optional named constructor parameters and ensure consistent state



during creation.

- **ImmutableDigitalObject** - Representation of an immutable, comparable concrete digital object, to be passed through web services and serializable with JAXB (Java Architecture for XML Binding).
- **ServiceDescription** - An entity to hold metadata about services. Using a builder ensures consistent object state during creation and models optional named constructor parameters while allowing immutable objects.
- **ServiceReport** - The result of calling a digital preservation service, where possible, information concerning the quality of the outputs should be placed in Events associated with DigitalObjects.
- **MigrationPath** - Simple class to build path metadata from. Contains the input and outputs of the path, and allows for parameters for that mapping.
- **Parameter** - This wraps the concept of a service parameter. When retrieved from a service, the default values should be set. This form does not allow optional v. required parameters, as ALL parameters should be explicitly specified. An 'optional' parameter implies an implicit default that would end up not being recorded in the audit trail.

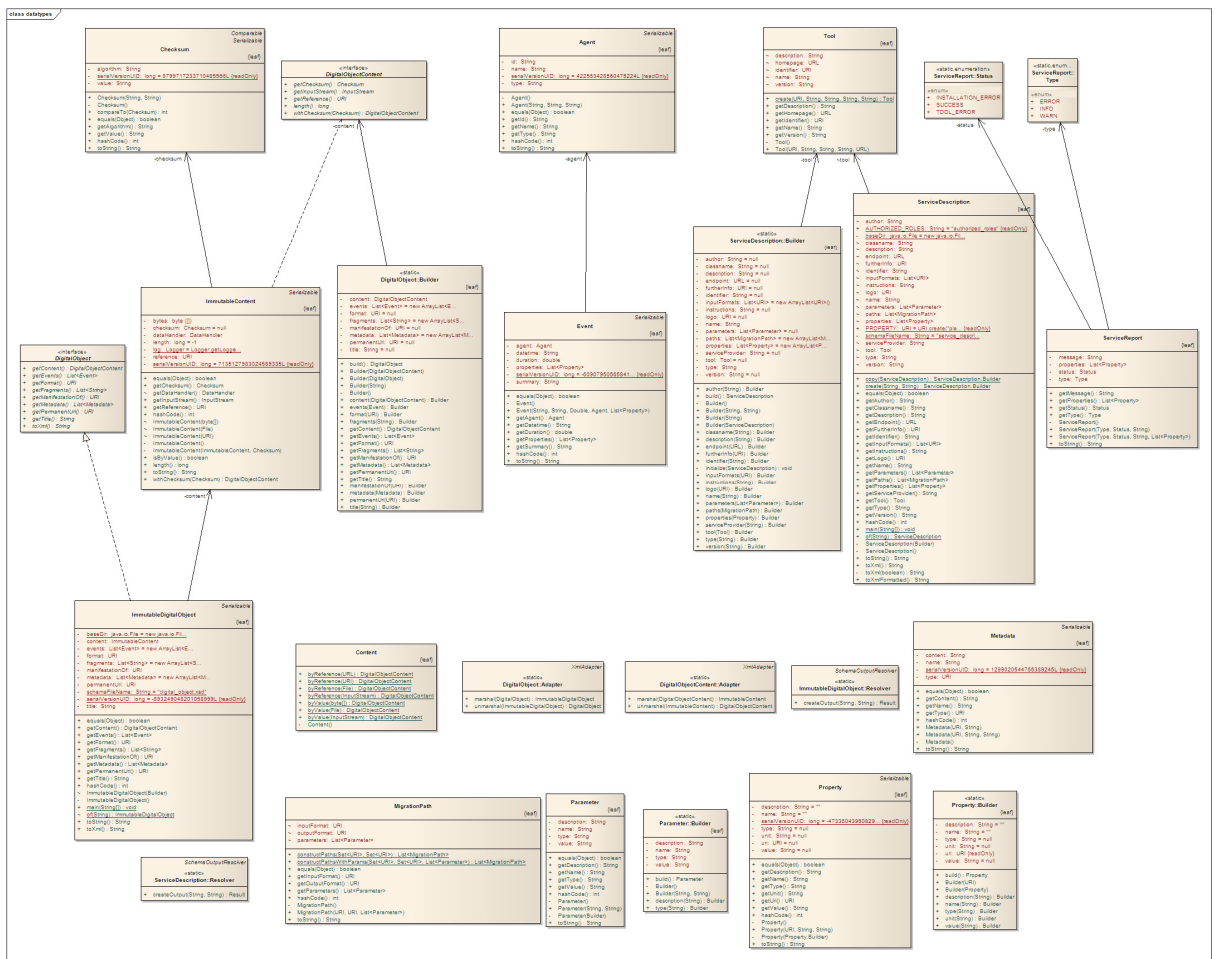


Figure 61 - Normalisation Service Data Model



The following diagram (Figure 62) represents the verbs or functional entities the service is able to provide. In essence, this is an attempt to concretely define the basic nouns and verbs of digital preservation, and so provide a suite of core preservation operations. These can be combined and swapped transparently when performing the same conceptual operation and therefore allow to perform a wide range of preservation processes.

Identification services extract a list of matching format URIs and record the methodology used to determine the format. The Characterisation operation makes it possible to extract a list of measurable properties and values from a given Digital Object. Validation services report on the well-formedness and validity of a record with respect to the indicated format, while Modification ones allow to enrich, corrupt, repair or crop a given digital object. The Comparison interface compares different objects based on metadata, extracted properties or normalized data to provide a degree of equivalence between those objects. The largest number of available tools implement the Migration interface i.e. preservation action to a specified target format. Planets has enhanced existing migration tools, as reported in [1].

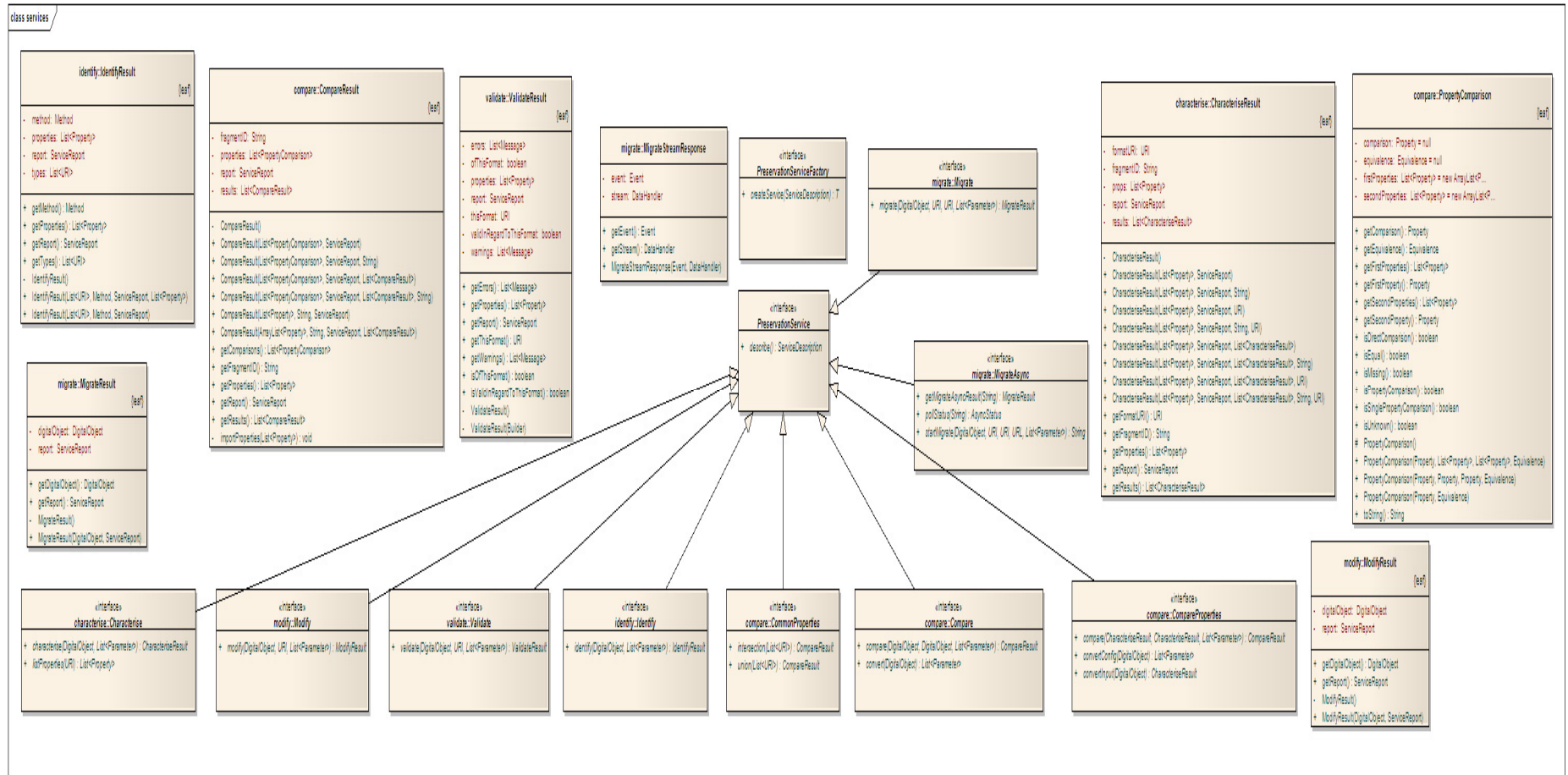


Figure 62 - Normalisation Services



The Assets Workflow and Service Execution Engine allow building complex high-level service assemblies and take care of their submission, execution and result delivery.

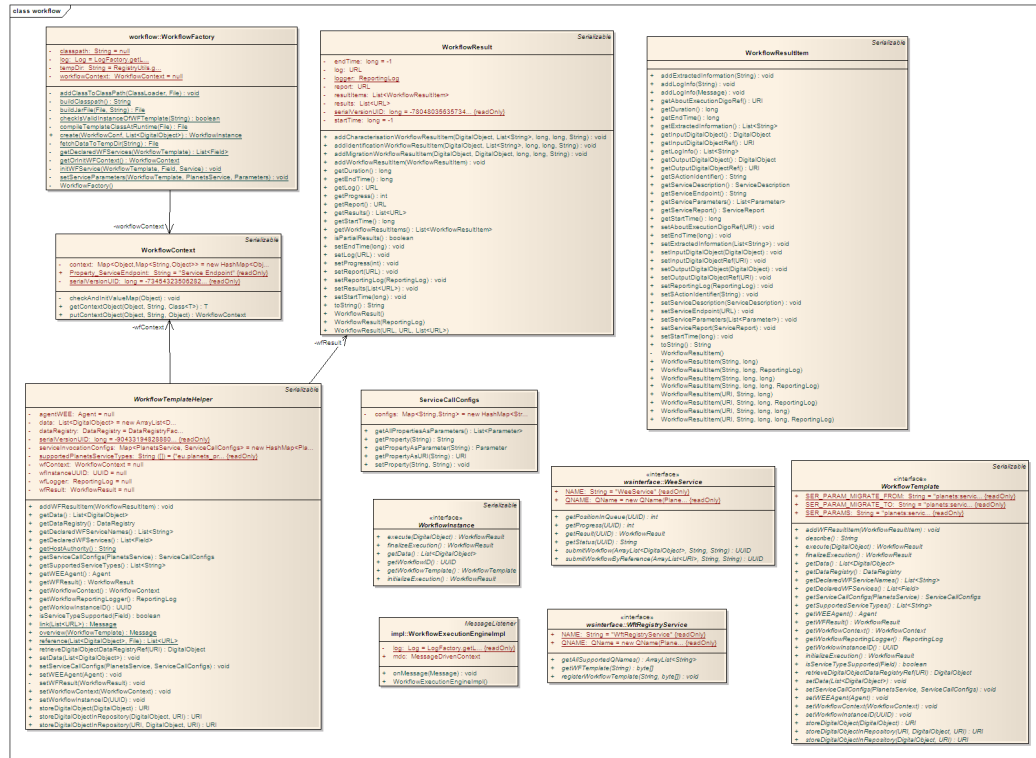


Figure 63 - Normalisation Workflow

Interface Name	ServiceRegistry
Key Concepts	ServiceDescription
Operations	<ul style="list-style-type: none"> <li>• <b>public Response register(ServiceDescription serviceDescription)</b> - @param serviceDescription The service description to register. @return A response message</li> <li>• <b>public List&lt;ServiceDescription&gt; query(ServiceDescription example)</b> - Query by example registry lookup. @param example The sample service description. @return The services for which all non-null values correspond to the values of the given sample object</li> <li>• <b>public List&lt;ServiceDescription&gt; queryWithMode(ServiceDescription example, MatchingMode mode)</b> - Query by example registry lookup with a specified lookup strategy. @param example The sample service description, @param mode The matching strategy to use when matching against the given sample, @return The services for which all non-null values correspond to the values of the given sample object, based on the supplied matching strategy</li> <li>• <b>public Response clear()</b> - Clears the registry of all entries. @return A response message</li> </ul>



	<ul style="list-style-type: none"> <li>• <b>public Response delete(ServiceDescription example)</b> - @param example The sample of the service descriptions to delete @return A response message.</li> </ul>
--	---

Interface Name	WorkflowExecutionManager
Key Concepts	WorkflowInstance, WorkflowConfiguration, WorkflowData, WorkflowReport, WorkflowStatus, WorkflowQueue
Operations	<ul style="list-style-type: none"> <li>• <b>public UUID submitWorkflow(WorkflowInstance workflow)</b> - Submits a workflow for execution. This will generate a UUID ticket and send a message containing all information (workflow, ticket) to the queue</li> <li>• <b>public WorkflowExecutionStatus getStatus(UUID ticket)</b> - returns the Workflow Execution Status for a given ticket.</li> <li>• <b>public int getPositionInQueue(UUID ticket)</b> - Returns the current position in the queue</li> <li>• <b>public void notify(UUID ticket, WorkflowResult wfResult, WorkflowExecutionStatus executionStatus)</b> - The callback method which is used by the execution engine to report back on a workflow execution and its status</li> <li>• <b>public void notify(UUID ticket, WorkflowExecutionStatus status)</b> - The callback method which is used by the execution engine to report back on a workflow execution's status</li> <li>• <b>public void notify(UUID ticket, WorkflowResult wfResult, WorkflowExecutionStatus executionStatus, int percentCompleted)</b> - The callback method which is used by the execution engine to report back on a workflow execution, its status, the current workflowResult and the percent of completed items</li> <li>• <b>public int getProgress(UUID ticket)</b> - submitted, completed or failed; 0-100 when execution is running</li> <li>• <b>public WorkflowResult getExecutionResult(UUID ticket)</b> - Method for polling the workflow execution results for a given ticket. public boolean isExecutionRunning(UUID ticket)</li> <li>• <b>public boolean isExecutionCompleted(UUID ticket)</b></li> <li>• <b>public boolean isExecutionFailed(UUID ticket)</b></li> </ul>

### 4.5.3 The Notification Models and Interfaces

This component is mainly an implementation of the *publish-subscribe* pattern, and for its responsibility it may be considered a component of the OAIS Preservation Planning.

In this perspective it's possible to identify its characterising concepts:

- Publisher - actor enabled to submit/publish messages related to specific terms of interest;



- Subscription – set of terms of interest which rely on a taxonomy;
- Subscriber - actor which expresses his/her own terms of interest for whom expects to receive alerts of changes
- Message - object used for delivering/notifying terms of interest. Notifications are the messages submitted by the publisher, and alerts are the messages received by the subscriber.

For the above reasons, this service shares the models and features defined in section 4.2.5.

In practice, the Preservation Notification Service supports a digital curator in managing the provenance information for a document within the Digital Library. In fact, the document is conceived as a “subscriber” which subscribes to “topics” according to its description. The alerts will be notified and will update the provenance whenever potentially impacting events occur. Provenance Information, is a key information of the OAIS (ISO:14721:2003) Preservation Description Information.

Service Name	Preservation Notification
Responsibility	1. Allow to manage the provenance information for objects
Provided Interfaces	1. ProvenanceManager
Dependencies	ASSETS Common, NotificationManager, RegistrationManager, TaxonomyManager

Interface Name	ProvenanceManager
Key Concepts	Provenance
Operations	<ul style="list-style-type: none"> <li>• <b>Provenance createProvenance(Identifier serviceId, Identifier documentId, Set&lt;Term&gt; documentDescription)</b> – allows a service to create and register provenance information for a potentially impacted document, described by a set of terms. This method will be implemented later;</li> <li>• <b>Provenance updateProvenance(Identifier serviceId, Identifier documentId, Alert alert)</b> – allows a service to update the provenance for the potentially impacted document through an alert. This method will be implemented later;</li> <li>• <b>Provenance getProvenance(Identifier serviceId, Identifier documentId)</b> - allows a service to obtain the provenance for a potentially impacted document. This method will be implemented later;</li> <li>• <b>boolean deleteProvenance(Identifier serviceId, Identifier documentId)</b> - allows a service to delete the provenance for a potentially impacted document. This method will be implemented later;</li> </ul>

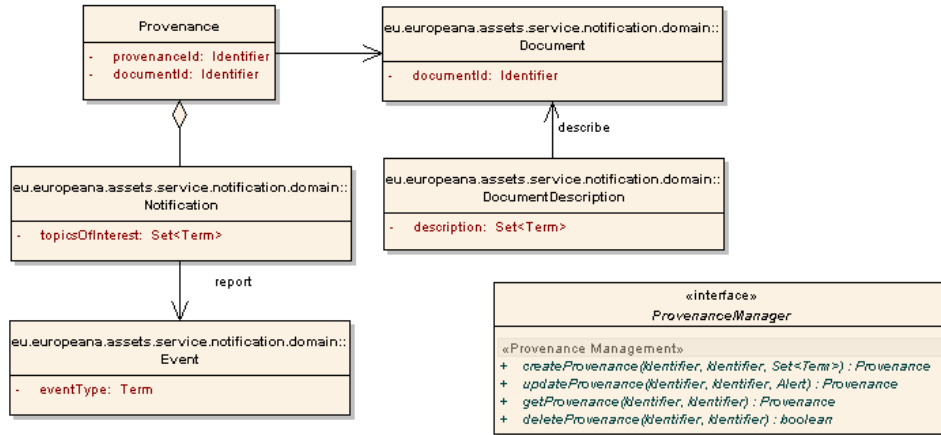


Figure 64 – Provenance Data Model and ProvenanceManager API

## 4.6 The Browsing and Content Characterisation Models and Interfaces

The interaction between the services engaged in the annotation propagation workflow is depicted in the Figure 65. The communication between the Content Management System (CMS), the Multimedia Index (MM Index), the Logging Service (Logs) and the various services is controlled by the ASSETS Common Interface.

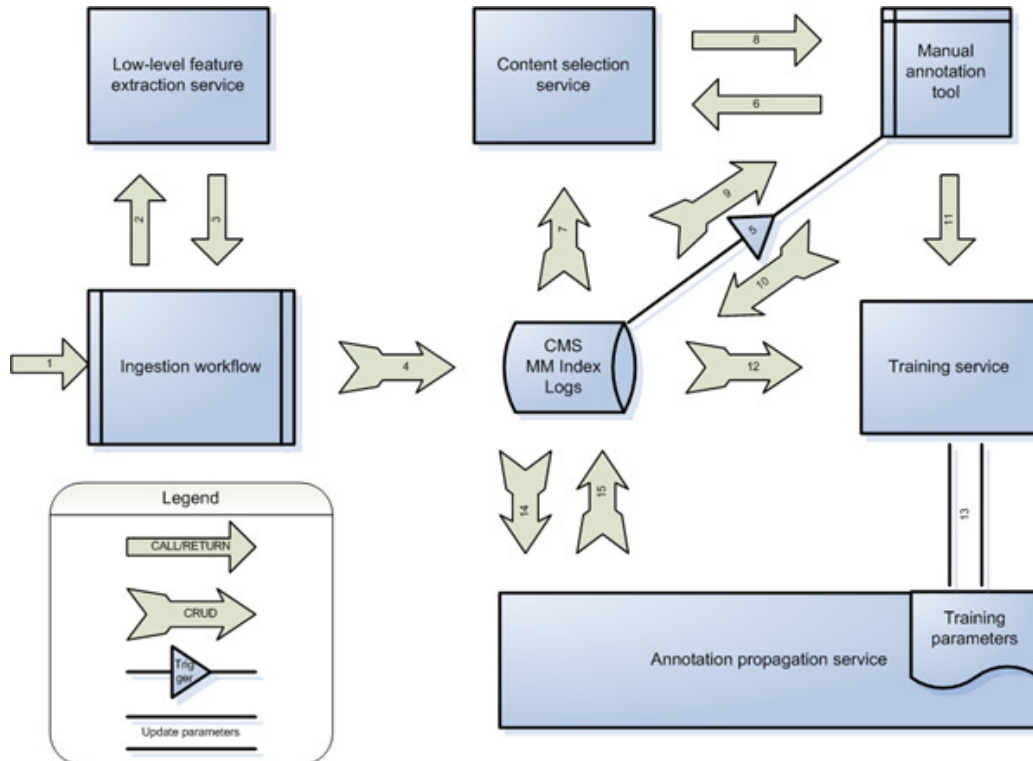


Figure 65 - Annotation Propagation Workflow

In general, the individual workflow steps are:

1. New objects (3D or images) arrive at the Ingestion Workflow.
2. The objects are sent to the Low-level feature extraction service.
3. The Low-level feature extraction service performs feature extraction and sends back the low-level descriptors of the objects.
4. The Ingestion Workflow stores the objects and their descriptors into the CMS and Multimedia Index respectively.
5. The manual annotation process is triggered (e.g. per email).
6. The Content selection service is called.
7. The Content selection service retrieves a list of all non-annotated objects and selects the “most appropriate” between them for manual annotation.
8. A list with the IDs of the selected objects is returned to the Manual annotation tool.
9. The selected objects/records (along with a screenshot or thumbnail) are retrieved from the CMS and an expert user, sitting in front of the Manual annotation tool, attaches labels to them, according to a predefined ASSETS taxonomy.
10. The Manual annotation tool updates the annotations of the selected objects in the Multimedia Index.
11. After the end of the manual annotation process, the Manual annotation tool passes the list of the annotated object IDs to the Training service.
12. The Training service retrieves the newly annotated objects.
13. The Training service uses the newly annotated objects to train the annotation propagation parameters.
14. The Annotation propagation service retrieves a list of all non-annotated objects and performs automatic annotation.
15. The Annotation propagation service updates the annotations produced by the automatic annotation.

#### 4.6.1 Annotation Propagation Service Models and Interfaces

This service will leverage information from already annotated objects, in order to automatically classify items that have not been annotated, e.g. newly entered items. The annotations derive from an ASSETS-specific taxonomy.

The classification can be used for media types, with low-level features that can be directly represented in a Euclidean metric space, e.g. images and 3D objects.

Service Name	Annotation Propagation Service
Responsibility	1. Automatic content classification
Provided Interfaces	1. ObjectClassification



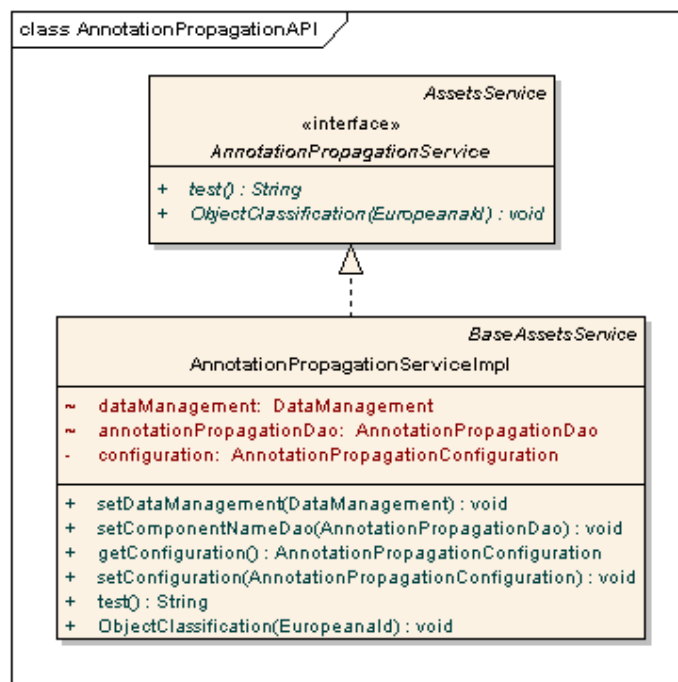
Dependencies	ASSETS Common, Training Service, Feature Extraction Services
--------------	--

Interface Name	ObjectClassification
Key Concepts	LowLevelFeatureVector, ASSETSTaxonomy, TrainingParameters
Operations	<ul style="list-style-type: none"> <li>• getTrainingParameters,</li> <li>• setTrainingParameters,</li> <li>• classifyObject</li> </ul>

The class diagram in Figure 66 presents the domain objects used by the annotation propagation service. It presents the TrainingParameters, which are computed during the training phase and are represented as a list of Float values.

The Class diagram of the Annotation Propagation Service.

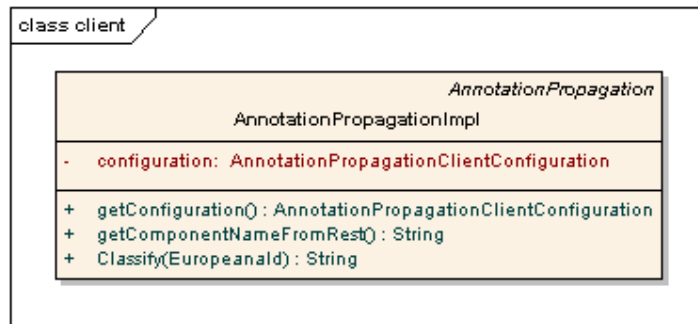
The method performing the classification of the europeana objects is shown. ObjectClassification receives a Europeana ID and classifies the given object.



**Figure 66 – Annotation Propagation Service model**

The class diagram in Figure 67 presents the client specification for the Annotation Propagation service. The methods used for testing the annotation propagation configuration are shown. Classify receives a Europeana ID and returns a String representation of the classification of the object.





**Figure 67 – Annotation Propagation client model**

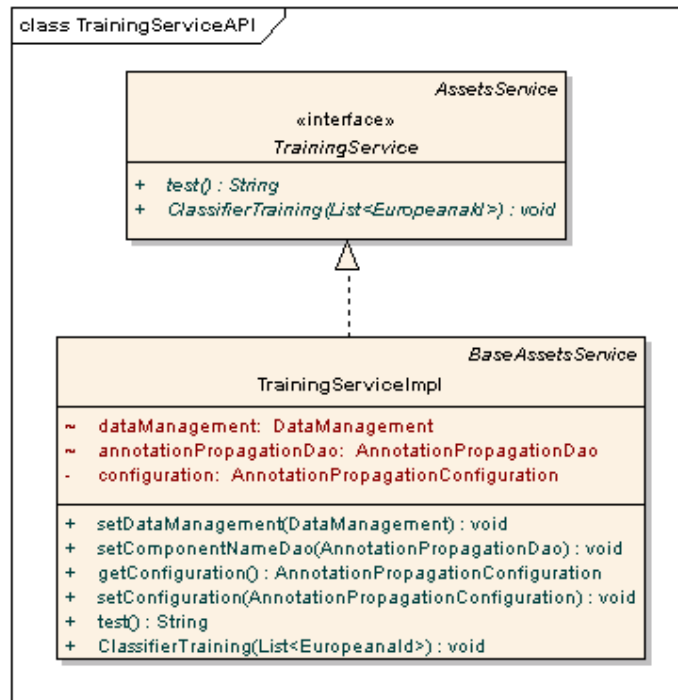
#### 4.6.2 Training Service Models and Interfaces

This service will train the Annotation Propagation Service, using pre-annotated training examples.

Service Name	Training Service
Responsibility	1. Content classification trainer
Provided Interfaces	1. ClassifierTraining
Dependencies	ASSETS Common, Manual Annotation and Annotation Correction Service, Feature Extraction Services

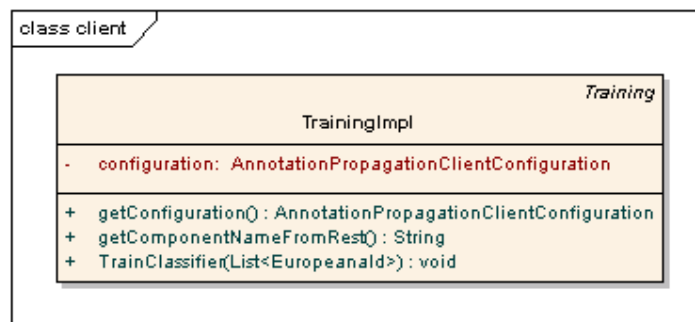
Interface Name	ClassifierTraining
Key Concepts	ObjectID, LowLevelFeatureVector, ASSETSTaxonomy, TrainingParameters
Operations	<ul style="list-style-type: none"> <li>• setAnnotatedObjectsList,</li> <li>• startTraining</li> </ul>

The class diagram of the Training Service is presented in Figure 68. The method performing the training of the classifier is shown. ClassifierTraining receives a list of Europeana IDs and uses the to compute the training parameters.



**Figure 68 - Classification Training Service model**

The class diagram of the Training Client is presented in Figure 69. The methods to test the training procedure are shown. TrainClassifier receives a list of Europeana IDs and uses the corresponding objects to train the classifier.



**Figure 69 - Classification Training client model**

#### 4.6.3 Manual Annotation and Annotation Correction Service Models and Interfaces

The service is a tool (standalone application) capable of manually correcting/completing automatic annotations that are provided by the ingestion process. This tool will also be used for providing the input to the training process of the Annotation Propagation Service.

For the first responsibility, the manual annotation tool calls the Log Service to retrieve a list of potentially wrong-annotated content. It interacts with ASSETS Common to retrieve

metadata of a selected content. Then the annotator corrects/completes these metadata according to a predefined ASSETS taxonomy. Finally, the tool updates Log and metadata through, respectively, the Log service and ASSETS Common.

For the second responsibility, the tool calls the Content Selection Service to retrieve a training list of content. It interacts with ASSETS Common to retrieve metadata of a selected content. Then the annotator assigns a classifying term(s) according to a predefined ASSETS taxonomy. The tool updates these metadata in the Multimedia Index through the ASSETS Common. Logs will be updated accordingly. After the end of the manual annotation process, the Manual annotation tool passes the list of the annotated content IDs to the Training service.

Service Name	Manual Annotation and Annotation Correction Service
Responsibility	<ol style="list-style-type: none"> <li>1. Manual correction and completion</li> <li>2. Training of the propagation service</li> </ol>
Provided Interfaces	No external interfaces are provided, except the graphical interface to the human annotator
Dependencies	<ol style="list-style-type: none"> <li>1. For the first responsibility : ASSETS Common, Log Service,</li> <li>2. For the second responsibility : ASSETS Common, Content Selection Service, Training Service, Log Service</li> </ol>

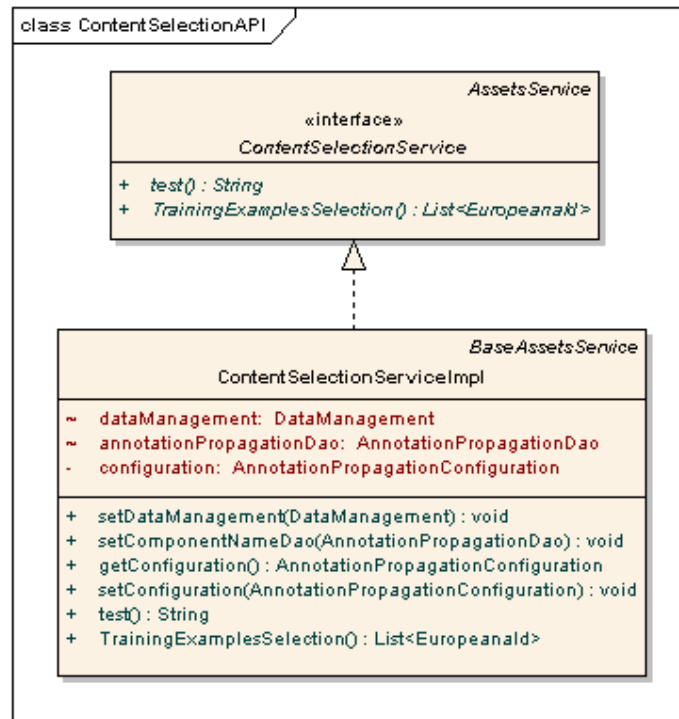
#### 4.6.4 Content Selection Service Models and Interfaces

This service will exploit Active Learning methods, in order to select the most "informative" training examples, which, after the manual annotation, will be used for training the Annotation Propagation Service.

Service Name	Content Selection Service
Responsibility	<ol style="list-style-type: none"> <li>1. Training examples selection</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. TrainingExamplesSelection</li> </ol>
Dependencies	ASSETS Common, Manual Annotation and Annotation Correction Service, Feature Extraction Services

Interface Name	TrainingExamplesSelection
Key Concepts	ObjectID, LowLevelFeatureVector
Operations	<ul style="list-style-type: none"> <li>• selectExamples</li> </ul>

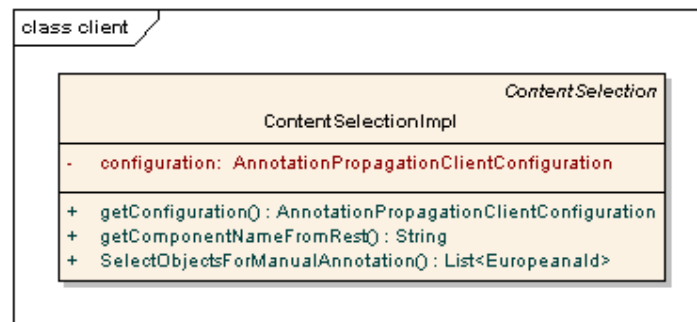
The class diagram of the Content Selection Service is presented in Figure 70. The method performing the content selection for the classifier training is shown. TrainingExamplesSelection selects a set of Europeana objects, which will be passed to the manual annotation and will be used as a training set for the training of the classifier.



**Figure 70 –Content Selection Service**

The class diagram of the Content Selection Client is shown in Figure 71.

The methods to test the content selection procedure are shown. SelectObjectsForManualAnnotation produces a list of Europeana IDs, which will be later used the classifier training.



**Figure 71 - Content Selection Client model**

#### 4.6.5 Relevance feedback service Models and Interfaces

User interaction will be translated to a “similarity score”, and will be inserted to the system in order to refine the already retrieved result set.

Service Name	Relevance feedback service
Responsibility	1. Retrieved results refinement
Provided Interfaces	1. ResultRefinement
Dependencies	ASSETS Common, Graphical User Interface

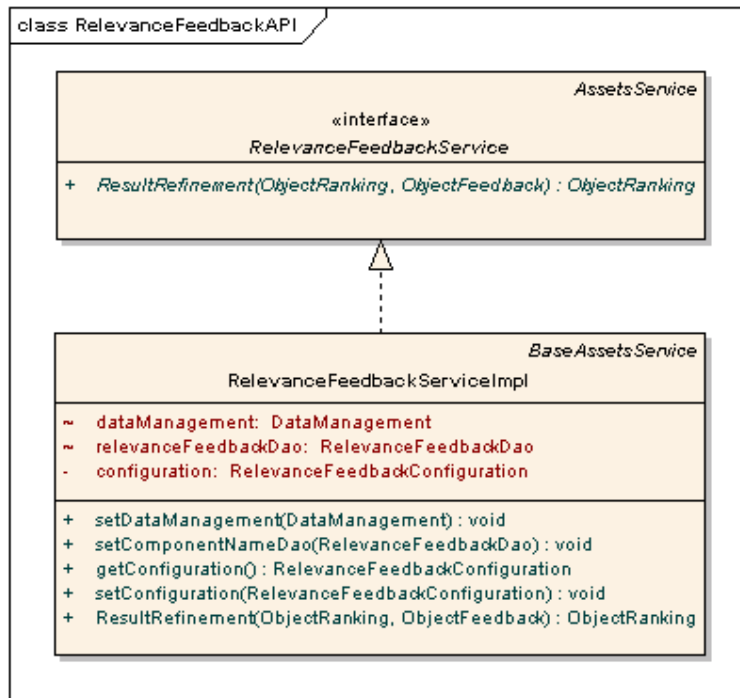
Interface Name	ResultRefinement
Key Concepts	RFElements, RFOldRanking
Operations	<ul style="list-style-type: none"> <li>refineResults</li> </ul>

The following domain objects used by the relevance feedback service:

- ObjectRanking contains object IDs, ranked according to a similarity measure, as they are returned after a search request, as a list of Europeanald.
- ObjectFeedback contains the user feedback, as a list of Object.

The class diagram of the Relevance Feedback Service is presented in Figure 72.

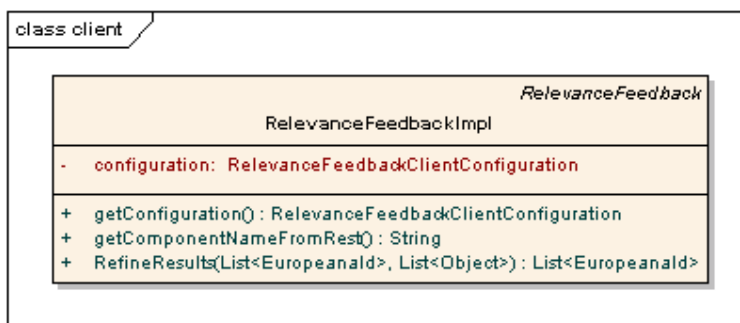
The method performing the search result refinement through user feedback is shown. ResultRefinement receives the old ranking and the user feedback and returns the new ranking.



**Figure 72- Relevance feedback Service model**

The class diagram of the Relevance Feedback Client is presented in Figure 73..

The methods to test the relevance feedback functionality are shown. RefineResults receives the old ranking, deriving from a previous search request, and the user feedback, and produces a new ranking of the search results.



**Figure 73 - Relevance Feedback Client**

#### 4.6.6 Log Analysis Service Models and Interfaces

This service provides a set of functionality for analyzing log data produced by services in the ASSETS ingestion system to monitor and review the content ingestion processes (in future, the service will be extended for query log analysis). This service allows client to formulate and evaluate analytical queries for obtaining statistical information of events happened in ingestion processes.

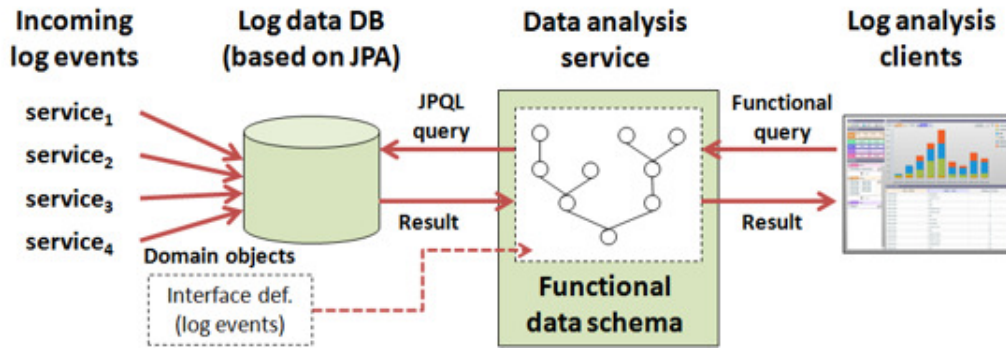


Figure 74- Log Analysis Outline

The main aim of this service is to provide a single point interface for analyzing log data coming from different services in heterogeneous forms. To achieve this aim, we introduce a higher abstract data model called functional data model (FDM) for data analysis proposed in [8]. The structure of each log data format is mapped to a graph-based data structure called functional data schema. The concept of the functional data schema gives us a uniform representation for different data structures. The FDM also provides a graph-based query language for formulating analytical queries over functional schema. Through a uniform representation of the data structure (functional schema) and FDM's query language, user can analyze log data produced by different services by means of the same manner.

The API of the log analysis service provides definitions of domain objects for representing functional schema and queries. The API also provides a synchronous version and asynchronous version of service interfaces for evaluating queries.

**Domain Model**

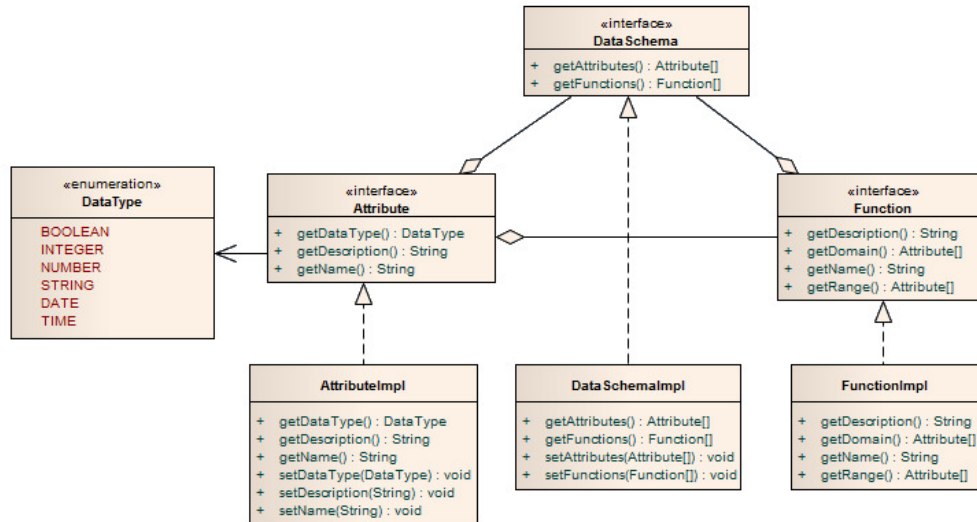


Figure 75 – Log Analysis Domain Model

In the FDM, a functional data schema is a directed acyclic graph (DAG) comprising a set of attributes as its nodes and functions as its edges.



## Queries

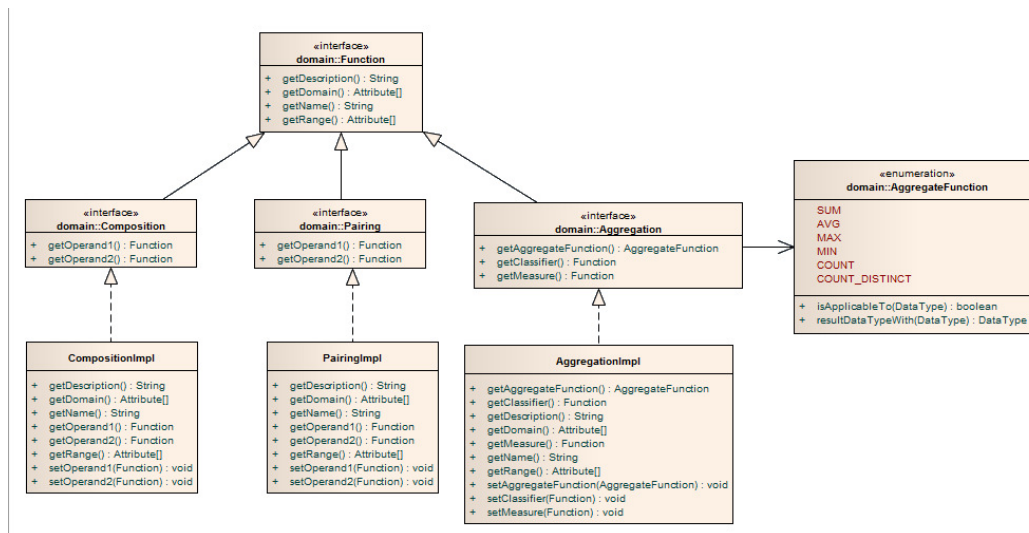


Figure 76 - Log Analysis Queries model

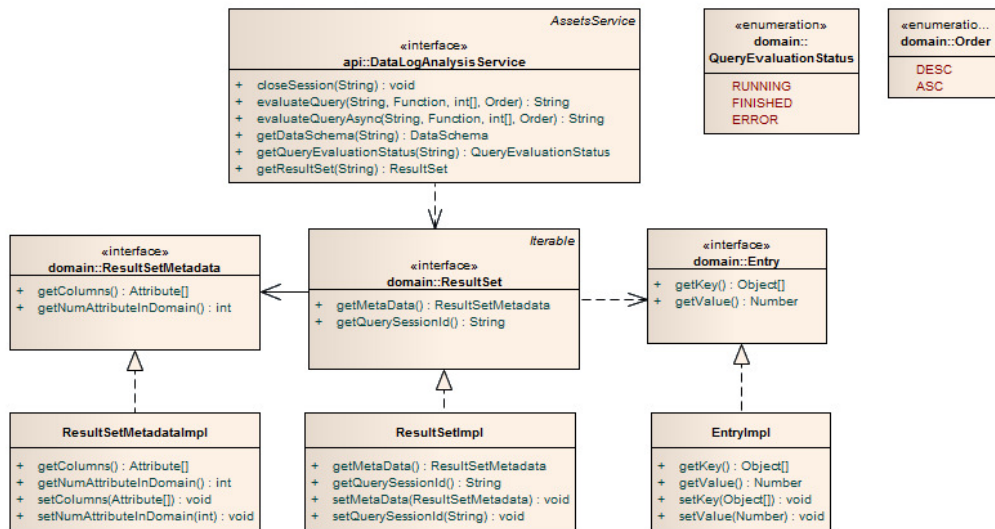
A query in the FDM is a data function in a data schema or a function derived from data function by applying operations. In FDM model, evaluating query is equivalent of obtaining an instance of extension of a given function. The FDM model provides two basic operation (pairing and composition) for deriving functions and aggregation operation. In particular, an analytical query for obtaining statistic of data is performed by an aggregation operation specified as a triple  $Q\langle c, m, op \rangle$ . Where “c” is a classifier function identifying values to make groups for aggregation; “m” is a measure function identifying values to aggregate “op” is an identifier of aggregation function (SUM, AVG, etc).

Service Name	Log Analysis Service
Responsibility	1. Evaluate analytical queries
Provided Interfaces	1. LogAnalysisService
Dependencies	ASSETS back-end database, ASSETS Commons, Ingestion Workflow Manager

Interface Name	LogAnalysisService
Key Concepts	Concepts in the functional data model (attribute, function, data schema, aggregate function, aggregation), result set, metadata of result set, query session
Operations	<ul style="list-style-type: none"> <li>getDataSchema</li> </ul>



- evaluateQuery
- evaluateQueryAsync
- getQueryEvaluationStatus
- getResultSet
- closeSession



**Figure 77 – Log Analysis Service API model**

The interface of the log analysis service provides the following functionality:

- to browse and obtain registered functional schema
- to evaluate queries synchronously
- to evaluate queries asynchronously

The evaluation result of a query is returned as an object implementing ResultSet interface. The ResultSet interface represents an evaluation result of a query (an extension of a function). It represents an extension of a function as a set of key-value mapping represented as Entry interface.

## 4.7 The Community Models and Interfaces

The ASSETS project focuses on three basic services for the final user, besides those addressed by WP2:

- A service for supporting the user in contributing content to Europeana; this is called User-Generated Content Service, abbreviated as UGC service.
- A service for supporting the user in accessing Europeana based on the publish/subscribe paradigm; this is called Taxonomy-based Notification service, abbreviated as TBN service.

- A service for supporting the user in expressing and using preferences when accessing Europeana; this is called the Personalization Service.

In this section, each service will be presented in detail, each in a separate sub-section.

#### **4.7.1 User Generated Content Models and Interfaces**

This service aims at supporting the composition of new digital objects from other simpler digital objects. To do so, a data model of objects as composite entities has been defined. The model describes how complex objects relate to their simpler constituents, including other objects as well as descriptions and versions. This data model is general enough to enable the definition of an abstraction layer to be put on top of existing models for the purposes of implementing advanced services. The service offers a basic API for implementing specific functionalities such as:

- tag or annotate existing object
- upload a new object
- translate metadata
- add metadata

and so on.

The UGC service will be deployed on the ASSETS Server. There will be a number of users of the service, each one authorized to contribute content to Europeana.

Each user of the service has its own Workspace (WS) on the ASSETS server, and is viewed by Europeana as a separate content provider. Overall, the ASSETS server is viewed by Europeana as an aggregator of content providers.

The user WS will contain units of works (UoW). Each UoW represents a separate contribution to Europeana that the user is constructing in his WS. A UoW contains objects, identified by URIs, and their accompanying descriptions. Objects in a UoW can be of two kinds:

- existing Europeana objects, that the user has included in a UoW in order to enrich them with new descriptions; existing Europeana objects can also be used as values of properties in descriptions.
- newly created objects, which we call UGC objects. These objects are original contributions to Europeana, and can be of two kinds:
  - they can be digital objects having an associated media file with the content; these objects will be called Media Objects
  - they can be digital objects for which no media file is available, or can be non-digital objects. These objects are simply UGC objects.

When the user has completed a UoW, it submits the UoW to Europeana, for inclusion in the Europeana database. To this end, the UoW will be packed in the form of a Submission Information Package (SIP) and placed on a special area of the user WS (Outbox), to be harvested by Europeana. After harvesting content, Europeana will analyze it and deliver a result message in a different special area of the user WS (Inbox). Submission is an asynchronous operation. There might occur hours before a submitted SIP is harvested by



Europeana, and days before the user receives the corresponding result message from Europeana.

Figure 78 below shows the general architecture of the UGC service.

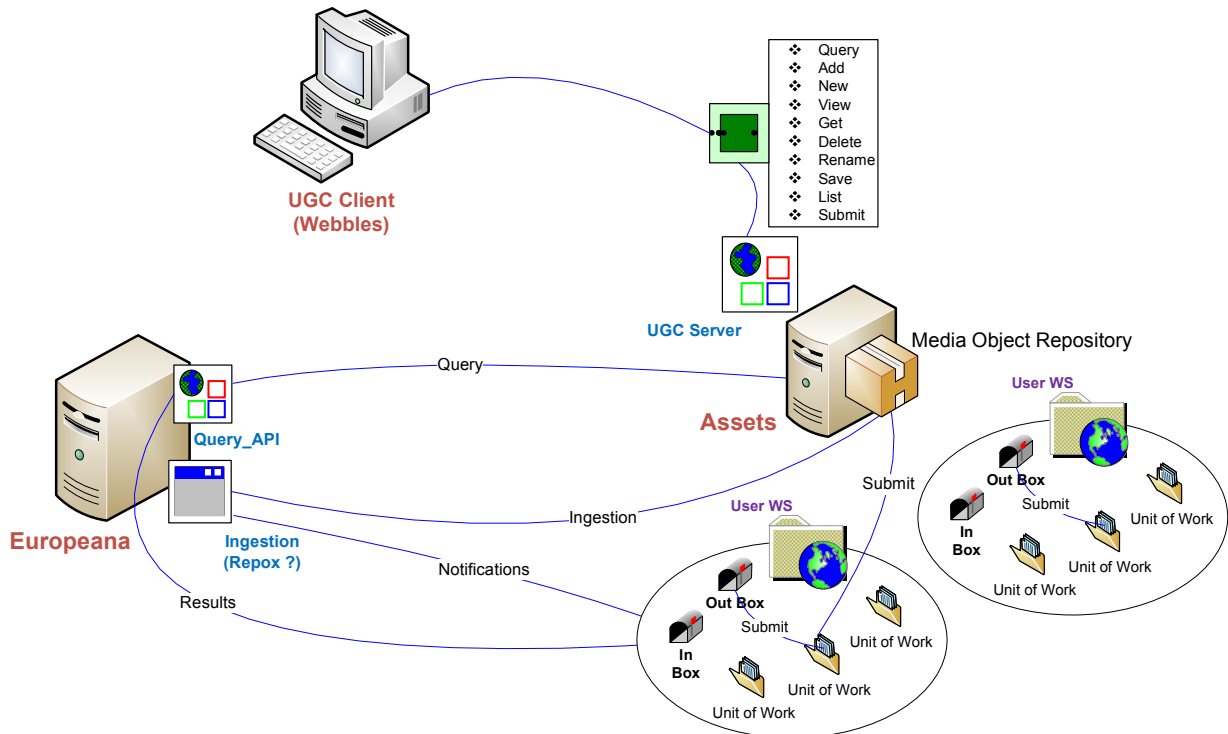


Figure 78 User Generated Content Service Architecture

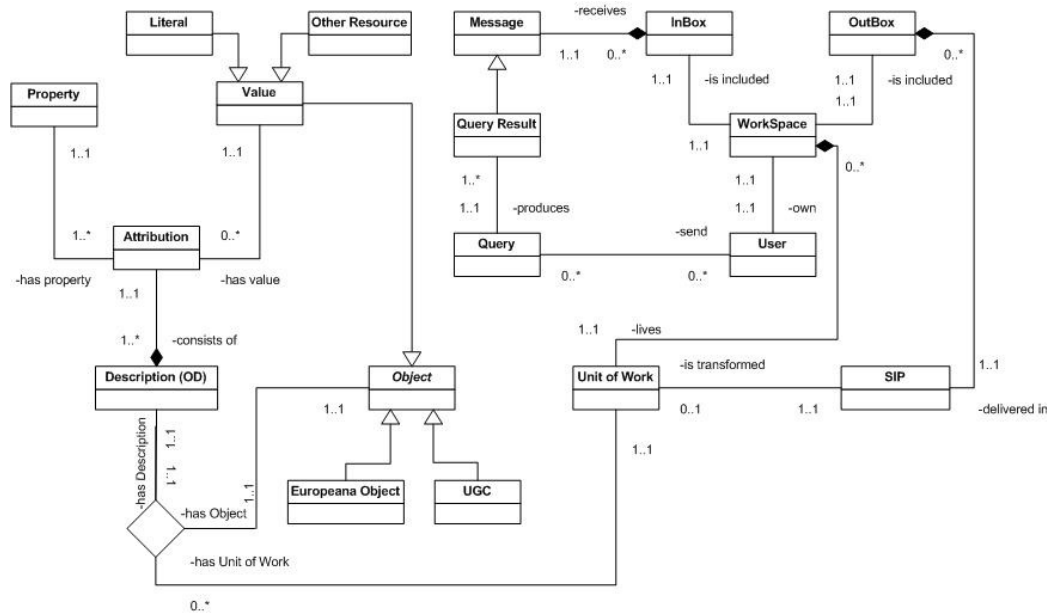
The three main components of this architecture are:

- **Europeana Server.** It provides an API to search for content in the digital library, using filters on metadata fields. Currently, the Europeana query API implements the "Open Search" directives (<http://www.opensearch.org/Home>). The result of a query contains a subset of metadata of the ESE schema and a URI to the full set of metadata and to the digital object. The Europeana server also provides an application for harvesting of data using the OAI-PMH protocol (e.g. Repos, is a GUI based tool for managing these activities).
- **Assets Server.** In the context of UGC, the ASSETS server provides functionality to communicate with Europeana and with the UGC client. The communication with the Europeana server is by invoking the Query\_API module (namely OpenSearch API). The server ASSETS allocate a workspace (USER WS) for each user, and provides an API (through the UGC Server) to manage the objects in it. The UGC server provides API as REST Web services to manage the user WS and to feed the UGC client. The API implementation is independent of any specific UGC client. The User WS contains an Inbox, an Outbox and a set of objects resulting from Europeana queries and/or loaded from the user workstation. The Inbox is used to contain Europeana query results and notification messages. Objects in the User WS are organized as **Units of Work (UoW)**, each of which forms a single contribution for Europeana, from the user point of view. The Inbox is used to receive Europeana query results and notification messages. The Outbox is used to hold messages about the submission of user-generated content, in the form of Submission Information Packages (SIP). Each SIP is generated from one UoW.

SIPs are stored in the ASSETS Media Object Repository (AMOR). Messages in the OutBox are useful to track user submissions and to restart a failed submission process. The AMOR implements OAI-PMH functionality to allow Europeana tools to harvest UGCs.

- **UGC Client.** This is a browser based GUI (e.g. Webbles as a special case) able to create new objects by assembling existing objects of the USER WS into more complex objects. The UGC Client interacts with the ASSETS Server via REST web services provided by the UGC Server module

The Figure below shows a UML class diagram including the classes and properties for modelling the UGC services. In the diagram, boxes represent classes and arcs represent association. Every association is bi-directional, but only the name in one direction is given for readability purposes. The name given refers to the association going from the class closest to the name to the other class. The reader can derive the name of the association in the opposite direction by following any notational convention.



**Figure 79 – The User Generated Contents Data Model**

Here we define the classes of the model in Figure 79 as key concepts that will lead to the definition of the UGC data types through a refinement process.

- **Object** is the class of objects that can be created or enriched by the user through the UGC service. An Object can be:
  - A newly created object, in which case it is an instances of class UGC (see below).
  - An existing Europeana object, in which case it is an instance of class Europeana object.

Each object and its associated description belongs to one or more Unit of Work (UoW):

- **User Generated Content (UGC)** is a new object created by the user by means of the UGC service. A UGC object can have an associated media file containing an image/text/sound/video. In this case, the UGC description specifies the name of the file



containing the image/text/sound/video, through some property. That file will be stored on the Assets server.

- **Object Description (OD)** is a metadata record, newly created by the User to describe a UGC or an existing Europeana object. In both cases, only one OD can be associated to an object. We model a description as a set of Attributions. The user is free of mixing in a single description properties from different schemas with new properties created by the user.
- **Attribution** models a single field of a metadata record. It is therefore linked to the property of the field and the value that the property assumes in the specific field.
- **Property** can be an EDM property or a user defined property. For each non-EDM property, the user must give the schema where the property belongs, the range of the property and the mapping to an EDM property. A property may be a multivalued property (e.g. italian DC.Title an english DC.Title).
- **Value** can be a Literal, an Object, or any Other resource. This means that within a description a user can refer to a UGC, thus creating complex objects.
- **Other resource** is the class of values used in Attributions that are neither UGCs nor Europeana objects (e.g. a VIAF<sup>3</sup> URI for Leonardo da Vinci, a TGN<sup>4</sup> URI for Paris, a link to a Wikipedia page, etc) .
- **Unit of Work (UoW)**. A UoW consists of the objects and their descriptions that form a single contribution for Europeana, from the user point of view. This contribution is not yet completed and therefore it exists as a UoW that the user can work on at any time. A UoW may include several objects, but these objects must form a connected graph.
- **Workspace** is the set of data structures that a user uses while interacting with UGC service. It consists of: an Inbox, an Outbox and a set of UoWs.
- **Inbox** is the class of all in-boxes. Each Inbox holds the messages that Europeana sends to the user of the WS where the Inbox belongs. A message may contain the results of a User query to Europeana, or a notification from Europeana to the User.
- **Outbox** is the class of all out-boxes. Each Outbox holds the SIPs that the user of the WS where the Outbox belongs has submitted to Europeana. Through Outboxes, Europeana will be able to harvest the user created SIPs. The classes Inbox and Outbox support communication between Europeana and the user. At implementation level, the Assets server might contain a single Outbox/InBox for all users by providing appropriate mechanisms for the identification of the messages sender/receiver.
- **Submission Information Package (SIP)** is the class of all SIPs. SIPs are in one-to-one relationship with UoW, in the sense that each SIP holds a UoW, and each UoW is held by one SIP. Essentially, a SIP is an XML file containing the objects and the descriptions created by the user in the format requested by Europeana. This format relies on the ORE<sup>5</sup> model for representing complex objects and their descriptions. We refer to the EDM Primer for an introduction to the EDM.
- **Message** is a notification (e.g. an error/warning/info/results) sent by the Europeana

<sup>3</sup> Virtual International Authority File; <http://viaf.org/>

<sup>4</sup> Getty Thesaurus of Geographic Names: <http://www.getty.edu/research/tools/vocabularies/tgn/index.html>

<sup>5</sup> Open Archives Initiative – Object Reuse and Exchange: <http://www.openarchives.org/ore/0.1/datamodel>



Server. It may include a Query Results if the notification has been generated by the execution of a query. Investigations on Europeana notification mechanism (if exists) are needed. The message will contain a message type (e.g. error/warning/info), a message code (e.g. RepoxError200) and a description.

- **Query.** A query could be simple or advanced and can generate a large amount of resulting records. In specifying a query should be given: 1) its type (simple/advanced), 2) the string containing the query expression, 3) the initial record in the result, and 4) the maximum number of records to return as a response (to avoid flooding the network with huge shipments). For a simple query the minimum score may be expressed as a constraint. The query expression is a Lucene expression. For advance search the query expression contains condition in Lucene style (e.g. (Title:Nicola AND Year:1915) OR Contributor:Cesare).
- **Query Result.** Is an Europeana generated Message that includes the results of a query execution in the form of an XML file (JSON based) containing: 1) the totalResultRows generated by the query execution, 2) the currentCallRows i.e. the amount of returned records (<= maxRow expressed in the query), 3) the execution time expressed in milliseconds, 3) the query expression and an array of FullDocs.

### Associations

- **describes:** this is a ternary association that links a Description (OD) to an Object in a given Unit of Work. An Object can have different Descriptions and a Description only describes a single Object; an Object can belong to many different Units of Work and a Unit of Work can have many Objects that belong to it. But an Object has exactly one Description in one Unit of Work. This means that there is a functional dependency from Unit of Work and Object to Description. The three sub-associations are:
  - **has Object:** this association links an Object (whether an Europeana object or an UGC object) to a triple in describes. An Object can appear in one to many triples.
  - **has Unit of Work:** this association links a Unit of Work to a triple in describes. A Unit of Work can appear in one to many triples
  - **has Description:** this association links a Description to a triple in describes. A Description appears in exactly one triple.
- **consists of:** this association links the Description (OD) and the Attribution classes. A Description is defined by (consists of) a set of attributions (at least one, as can be seen from the cardinality and optionality constraints).
- **has Value:** links an Attribution to the value. One Attribution can have many Values, since metadata properties are by default multi-valued. A Value can appear in many Attributions. This is a non-mandatory association, as there may exist Attributions with no value.
- **has Property:** links an Attribution to the property. One Attribution has exactly one Property, whereas a Property can appear in many Attributions. This is a mandatory association, as there cannot exist Attributions without a Property.
- **lives:** a Unit of Work lives in exactly one Workspace, and a Workspace can be the living place of more than one Unit of Works.
- **is transformed:** this association highlights the fact that a SIP is obtained from a Unit of Work through a transformation process, upon completion of the user work. A Unit of



Work can generate a single SIP and a SIP can be generated by a single Unit of Work.

- **delivered in:** This association links SIPs and the OutBoxes where they belong. At a given time instant, an OutBox can contain more than one SIP, waiting to be harvested. Conversely, a SIP can be placed only on one OutBox.
- **is included:** this association links an Inbox or an OutBox to a Workspace. An Inbox or an OutBox are included in exactly one Workspace, and a Workspace includes exactly one Inbox and exactly one OutBox.
- **send:** A User can submit Queries to the Europeana server in order to be able to use the resulting objects for the creation of new objects. A User can submit more than one Query and conversely the same Query could be sent by more than one User. This association is not stored.
- **produced:** this association relates a Query to the (possibly empty) resulting Query Result in the form of a Message. A Query has exactly one Query Result, while the same Query Result could be generated by different Queries. This association is not stored.
- **receives:** system notifications and query results are delivered by Europeana in the user Inbox. At a given instant, an Inbox can contain zero to many Messages. Conversely, a Message can be received by a single Inbox.
- **own:** each Assets Workspace is owned by a single User and viceversa.

Service Name	Content creation by re-use
Responsibility	<ol style="list-style-type: none"> <li>1. Work space management;</li> <li>2. Session Management;</li> <li>3. Media File management;</li> <li>4. Inbox Management;</li> <li>5. Outbox management;</li> <li>6. Query management;</li> <li>7. Object management</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. OutBox;</li> <li>2. Inbox;</li> <li>3. Workspace;</li> <li>4. MediaObjectRepository</li> </ol>
Dependencies	Session Management, Media File management (ASSETS-Commons)

Interface Name	MailBox (for both OutBox and Inbox)
Key Concepts	Message, QueryResults
Operations	<ul style="list-style-type: none"> <li>• getMessage,</li> <li>• getMessages,</li> </ul>



	<ul style="list-style-type: none"> <li>• putMessage,</li> <li>• removeMessage,</li> <li>• removeMessages,</li> <li>• removeAllMessages</li> </ul>
--	---

Interface Name	Workspace
Key Concepts	UnitOfWork, Query
Operations	<ul style="list-style-type: none"> <li>• list,</li> <li>• createUnitOfWork,</li> <li>• getUnitOfWork,</li> <li>• removeUnitOfWork,</li> <li>• renameUnitOfWork,</li> <li>• executeQuery</li> </ul>

Interface Name	MediaObjectRepository
Key Concepts	MediaObject
Operations	<ul style="list-style-type: none"> <li>• add,</li> <li>• getObjects,</li> <li>• getObject,</li> <li>• removeObject</li> </ul>

#### Domain Interfaces

Interface Name	Message
Key Concepts	QueryResult, SipIdentifier, MessageIdentifier
Operations	<ul style="list-style-type: none"> <li>• getIdentifier / setIdentifier,</li> <li>• getFrom / setFrom,</li> <li>• getTo / setTo,</li> <li>• getSubject / setSubject,</li> <li>• getReturnCode / setReturnCode</li> <li>• getMessage / setMessage</li> <li>• getSip / setSip</li> <li>• getQueryResults / setQueryResults</li> </ul>





Interface Name	UnitOfWork
Key Concepts	DLObject, SIP
Operations	<ul style="list-style-type: none"> <li>• getName / setName,</li> <li>• getSip / setSip</li> <li>• submit</li> <li>• getObjects / setObjects,</li> <li>• getObject</li> <li>• updateObject,</li> <li>• addObject,</li> <li>• removeObject</li> </ul>

Interface Name	DLObject
Key Concepts	EuropeanaObject, UGCOject
Operations	<ul style="list-style-type: none"> <li>• getEuropeanaObjects / setEuropeanaObjects,</li> <li>• getUGCOjects / setUGCOjects</li> </ul>

Interface Name	EuropeanaObject
Key Concepts	ObjectDescription
Operations	<ul style="list-style-type: none"> <li>• getId / setId,</li> <li>• getObjectType / setObjectType,</li> <li>• getDescription / setDescription</li> </ul>

Interface Name	UGCOject
Key Concepts	ObjectDescription
Operations	<ul style="list-style-type: none"> <li>• getName / setName,</li> <li>• getOriginalFileName / setOriginalFileName,</li> <li>• getCreationDate / setCreationDate</li> <li>• getModifiedDate / setModifiedDate</li> </ul>

Interface Name	ObjectDescription
Key Concepts	Attribution
Operations	<ul style="list-style-type: none"> <li>• getAttributions / setAttributions,</li> <li>• addAttribution / getAttribution,</li> <li>• removeAttribution</li> </ul>

Interface Name	Attribution
Key Concepts	Property
Operations	<ul style="list-style-type: none"> <li>• getValueType / setValueType</li> <li>• getProperty / setProperty</li> <li>• getResourceValue / setResourceValue</li> <li>• getLiteralValue / setLiteralValue</li> </ul>

Interface Name	QueryResults
Key Concepts	AssetsBriefDoc
Operations	<ul style="list-style-type: none"> <li>• getTotalResultRows / setTotalResultRows,</li> <li>• getCurrentCallRows / setCurrentCallRows,</li> <li>• getMilliseconds / setMilliseconds,</li> <li>• getBriefDocs / setBriefDocs</li> </ul>

Interface Name	SessionInterface
Key Concepts	User, Privileges, Query, QueryIdentifier
Operations	<ul style="list-style-type: none"> <li>• void login (User user),</li> <li>• void logout(),</li> <li>• Privileges checkPrivileges (User user),</li> <li>• QueryIdentifier execute (Query query)</li> </ul>

The following figure shows the UML Class Diagram for the above definitions.

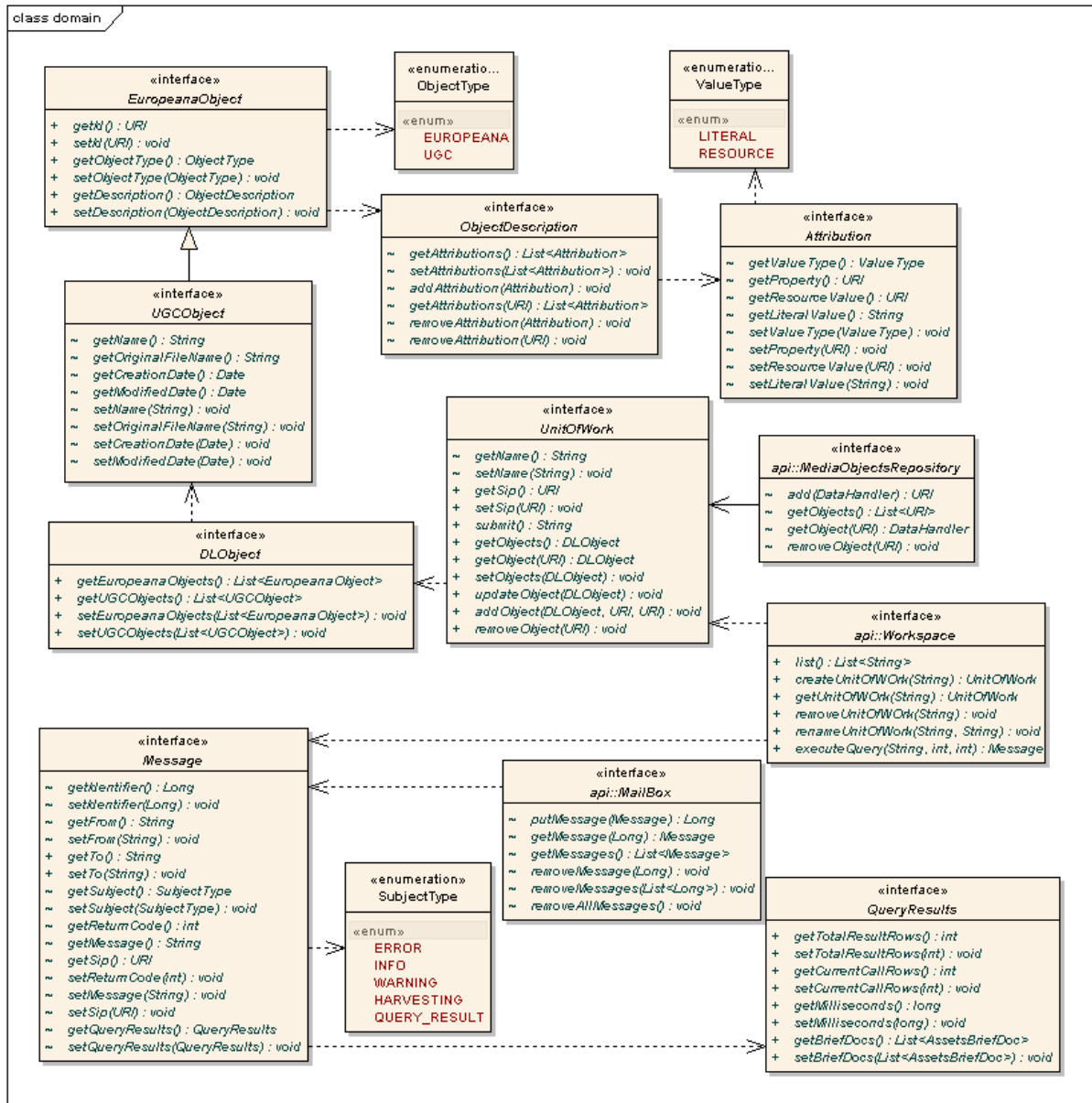


Figure 80 - User Generated Content Data Model

#### 4.7.2 Taxonomy-based Notification Service Models and Interfaces

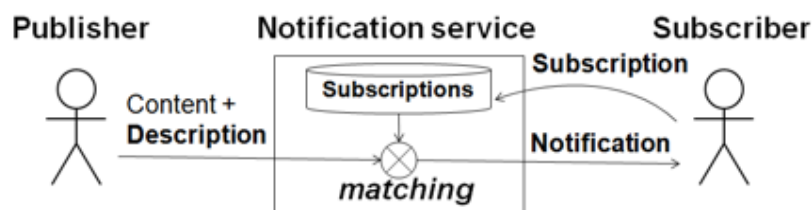
The purpose of this service is to allow users to be notified when objects of interest to them are ingested or updated in Europeana. This is done by first allowing users to subscribe to profiles, made up of terms from a controlled vocabulary, or taxonomy used for the description of objects. After the subscription, when the user logs in the myEuropeana service, he will find an alert signaling that there are newly added objects to Europeana which satisfy one of the profiles to which the user subscribed and therefore are of interest to the user.

The taxonomy based notification service will implement the Publish/Subscribe approach to messaging. According to this approach Publishers and Subscribers may dynamically publish or subscribe to the content hierarchy. The system takes care of distributing the messages arriving from a topic's multiple publishers to its multiple subscribers. Topics retain messages only as long as it takes to distribute them to current subscribers.

For that reason, the taxonomy-based notification, as well as the preservation notification, shares models and interfaces defined in section 4.2.5 for the Common Notification.

The advantage of the taxonomy-based notification is that it allows users to define subscriptions more flexibly and with less cost. In this approach, the hierarchy of terms in a taxonomy is used for determining "is-a" relationships among descriptions and subscriptions. For instance, users can receive alerts about "German watercolor", "Italian watercolor portrait" and "French gouache in 19th century" by subscribing only "European watercolor". The taxonomy-based notification service implements an efficient algorithm to compare descriptions and subscriptions by taking account a taxonomy.

**Domain Model:** Publishers and Subscribers



**Figure 81 - Taxonomy-based notification users**

We can distinguish users in two main categories of actors:

- **publisher:** (typically Europeana) publishes content into the portal together with its description. This description comprises a set of terms coming from a controlled vocabulary, called the taxonomy, which is organized as a structured hierarchically.
- **subscriber:** uses the content published for satisfying some information need, described as a set of terms from the taxonomy. Such a set of terms is called a subscription. The goal of the taxonomy-based notification service is to retrieve a set of new content that matches a given subscription. This is done in two phases:
  - The user declares a subscription to the notification service, which stores it into the system.

- The notification service compares a given subscription with content descriptions published in a specified period of time. Whenever a description matches to the subscription, a notification is sent to the user.

### Overall Architecture

Usually the interaction between users and a notification service can be implemented according two models:

- Pull:
  - the component implementing the notification service periodically asks to publishers for changes related to topics
  - subscribers periodically queries the service to check if there are notification messages
- Push:
  - publishers sends notifications to the service
  - the notification service sends notifications to subscribers

The Taxonomy based Notification service will be deployed on the ASSETS Server and will implement a pull model. In our case the publisher is the Europeana server, therefore the notification service will periodically inspect the Europeana database asking for changes related to the topics defined in the taxonomy. Subscribers will check for notifications in the following way: once they log in the ASSETS server all notifications (if any) will be shown in their client GUI, during the session the client will implement a "refresh" of the list in a transparent way by periodically contacting the service and updating the notifications list.

### Implementation Approach

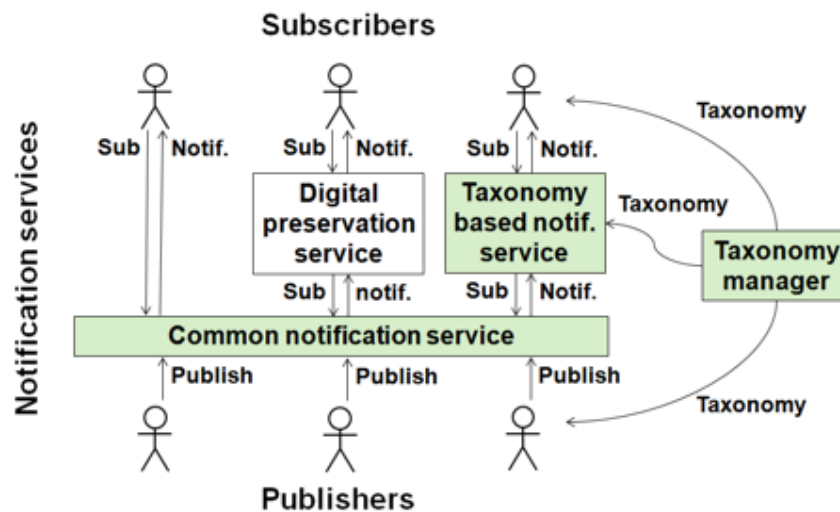
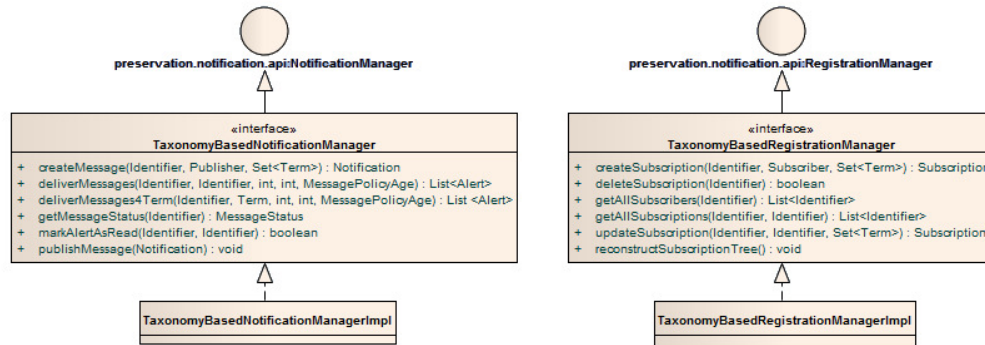


Figure 82 – Taxonomy-based Implementation Outline

This service will be implemented the top of the (common) notification service, which is a part of the digital preservation service (Task 2.3.3). The main aim of the taxonomy based notification service is extend functionality of the common notification service to perform description-subscription matching by taking account a taxonomy. Therefore, the taxonomy

based notification service will be implemented as a client of the common notification service. Due to compatibility among notification services, the taxonomy-based notification service provides an API that is completely based on the common notification service. The service provides two service interfaces that directly inherit NotificationManager interface and RegistrationManager interface.



**Figure 83 – Taxonomy-based Notification Service API model**

Service Name	Taxonomy-based Notification
Responsibility	<ol style="list-style-type: none"> <li>1. Allows subscription for topics of interest</li> <li>2. Allows notification of changes</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. TaxonomyBasedNotificationManager</li> <li>2. TaxonomyBasedRegistrationManager</li> </ol>
Dependencies	Session Management, ASSETS Common, Common Notification

Interface Name	TaxonomyBasedNotificationManager (inherits interface NotificationManager from Common Notification)
Key Concepts	Message, Taxonomy
Operations	<ul style="list-style-type: none"> <li>• createMessage</li> <li>• publishMessage</li> <li>• deliverMessages</li> <li>• deliverMessages4Term</li> <li>• getMessageStatus</li> <li>• markAlertAsRead</li> </ul>

Interface Name	TaxonomyBasedRegistrationManager (inherits interface RegistrationManager from Common Notification)
Key Concepts	Subscriber, Subscription, Taxonomy



Operations	<ul style="list-style-type: none"> <li>• createSubscription</li> <li>• updateSubscription</li> <li>• deleteSubscription</li> <li>• getAllSubscriptions</li> <li>• getAllSubscribers</li> <li>• reconstructSubscriptionTree – rebuild a subscription tree</li> </ul>
------------	---

### 4.7.3 Personalisation service Models and Interfaces

Service Name	Preference management service
Responsibility	<ol style="list-style-type: none"> <li>1. Stores user preferences into the ASSETS server;</li> <li>2. Retrieves user preferences stored in the server;</li> <li>3. Updates stored preferences</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. UserPreferenceManager</li> </ol>
Dependencies	Session management and ASSETS common's storage to store user preferences

Interface Name	UserPreferenceManager
Key Concepts	User and User preference
Operations	<ul style="list-style-type: none"> <li>• getPreference</li> <li>• postPreference</li> </ul>

Service Name	Personalized query service
Responsibility	<ol style="list-style-type: none"> <li>1. Create new personalized query session;</li> <li>2. Evaluates a query with a preference;</li> <li>3. Allows users to navigate in blocks of retrieved items;</li> <li>4. Close personalized query sessions after predefined duration</li> </ol>
Provided Interfaces	<ol style="list-style-type: none"> <li>1. PersonalizedQueryService</li> </ol>
Dependencies	ASSETS common's simple query service and Session management

Interface Name	PersonalizedQueryService
----------------	--------------------------

Key Concepts	User preference, Query, Result set (block), Metadata of block and Personalized query session
Operations	<ul style="list-style-type: none"><li>• evaluatePersonalizedQuery;</li><li>• getResultBlockMetadata;</li><li>• getResultBlock;</li><li>• closePersonalizedQuerySession</li></ul>



## Appendix 1 - Enrichment Services Training Data Format

This page<sup>6</sup> provides information on how CPs have to format their data in order to submit training data to the enrichment services of WP2.1:

- Metadata cleaning
- Knowledge extraction
- Metadata classification

The training data file format is XML. For each task an XML schema file, i.e., an XSD file, is provided. CPs should use the schema for each task to produce their training data files (one file for each task).

In order to simplify the process we also provide an XML example file for each task and a document with guidelines and comments. CPs could follow the guidelines and use the examples as a starting point to produce their training data files.

The wiki folder "Enrichment Services Training Data Guidelines" contains both the guidelines for the content providers and the xsd/xml files that are to be used in preparing the training sets. More in details, that folder contains the following files:

- cleaningSchema.xsd.txt: XML Schema Definition for providing training sets to task T2.1.1 "Metadata Cleaning". The file extension is txt because the wiki does not allow the upload of xsd files. The file should be renamed by removing the extension .txt.
- cleaningExample.xml: example of a well-formed training set for task T2.1.1. The content providers may modify this file for providing their training data.
- extractionSchema.xsd.txt: XML Schema Definition for providing training sets to task T2.1.2 "Knowledge Extraction". The file extension is txt because the wiki does not allow the upload of xsd files. The file should be renamed by removing the extension .txt.
- extractionExample.xml: example of a well-formed training set for task T2.1.2. The content providers may modify this file for providing their training data.
- classificationSchema.xsd.txt: XML Schema Definition for providing training sets to task T2.1.3 "Metadata Classification". The file extension is txt because the wiki does not allow the upload of xsd files. The file should be renamed by removing the extension .txt.
- classificationExample.xml: example of a well-formed training set for task T2.1.3. The content providers may modify this file for providing their training data.
- T2.1\_TrainingGuidelines.pdf: pdf document presenting and discussing the xml/xsd files above.
- T2.1\_Training.tgz": tgz pack containing all of the previous files (with the correct file extension).

All of the previous files are available on this wiki at the following URL:  
[http://www.assets4europeana.eu/web/portal/documents?p\\_p\\_id=20&folderId=35495](http://www.assets4europeana.eu/web/portal/documents?p_p_id=20&folderId=35495)

---

<sup>6</sup> <http://www.assets4europeana.eu/web/portal/wiki/-/wiki/Main/Enrichment%20Services%20Training%20Data%20Format>



## 5. References

---

1. ASSETS D2.0.1 "Requirements Specification" – Internal Document
2. ASSETS MS12 "System Architecture" – Internal Document
3. ASSETS MS27 "Digital Preservation Service Design"– Internal Document
4. Roy Thomas Fielding "Architectural Styles and the Design of Network-based Software Architectures", 2000 – available at [http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding\\_dissertation.pdf](http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf)
5. H. Belhaj-Frej, P. Rigaux, N. Spyrtos "User notification in taxonomy based digital libraries", SIGDOC '06 Proceedings of the 24th annual ACM international conference on Design of communication. Available at <http://portal.acm.org/citation.cfm?id=1166366>
6. Europeana Aggregator's Handbook - available at [http://www.version1.europeana.eu/c/document\\_library/get\\_file?uuid=94bcddbfc3625-4e6d-8135-c7375d6bbc62&groupId=10602](http://www.version1.europeana.eu/c/document_library/get_file?uuid=94bcddbfc3625-4e6d-8135-c7375d6bbc62&groupId=10602)
7. ASSETS D2.0.2 "Interface Specifications and System Design" – Internal Document
8. Nicolas Spyrtos. "A Functional Model for Data Analysis". Proc. of Flexible Query Answering Systems (FQAS) 2006, pp 51-64, 2006.
9. Eld Zierau "The Planets Approach to Migration Tools" – available at [http://www.planets-project.eu/docs/papers/Archiving2008\\_Zierau\\_Wijk.pdf](http://www.planets-project.eu/docs/papers/Archiving2008_Zierau_Wijk.pdf)