DELIVERABLE

Project Acronym:	**EuDML**
Grant Agreement number:	**250503**
Project Title:	**The European Digital Mathematics Library**

Deliverable 4.3 – EuDML global system functional specification and design- Revision

Revision: 1.0

Authors:

Aleksander Nowiński (ICM)

Wojtek Sylwestrzak (ICM)

Krzysztof Wojciechowski (ICM)

José Borbinha (IST)

Contributors:

Gilberto Pedrosa (IST)

João Edmundo (IST)

Project co-funded by the European Commission within the ICT Policy Support Programme

Last changed: 10/02/2013

Dissemination level:	Public (PU)

Revision History

Revision	Date	Author	Organisation	Description
v0.01	10.01.13	Aleksander Nowiński	ICM	First draft
v0.02	1.02.13	Glberto Pedrosa	IST	Content review
v0.03	1.02.13	João Edmundo	IST	Content review
v0.04	08/02/13	Alan Sexton	UB	Internal Review
v0.05	09/02/13	Aleksander Nowinski	ICM	Updates regarding D4.1
v1.00	10-02-13	José Borbinha	IST	Final version

The document versions are controlled by the EuDML SVN version control system. The above revisions are for informational purpose only and do not reflect the actual SVN version numbers.

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise.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

Table of Contents

Executive summary

The document describes the EuDML architecture as a service oriented, document centric system. The system is divided into three main components: *web interface* – which provides user interface and REST services, *backend* – which provides services and *REPOX* – which is responsible for harvesting data from data providers and serving contents of the system over the OAI-PMH protocol. Components communicate using HTTP-invoke protocol (encapsulated by Spring Remoting).

The *Web Interface* component is based on the YaddaWeb2 system, and has been modified and adapted to match specific EuDML system requirements (alternative metadata format, mathematical component, new annotation component etc.). It delivers a web UI to the users, but also manages REST services interface (see **D6.4: Web and Service Interface Implementation – Service Interface**) for the machine clients. The security (authentication and authorization) is implemented and enforced within this layer. Web UI communicates directly with the backend module.

The *Backend module* hosts and orchestrates all the services, including, most importantly: storage, search, browse, workflow, user catalog, similarity and annotation. Documents (both publication metadata and content) are stored in the form of records identified by EuDML Id. A workflow processing service is used to process documents from storage for indexing or enhancement (see **D4.4 – The EuDML Information Life Cycle Process** and **D7.4: Toolset for Image and Text Processing and Metadata Enhancements — Final Release** for details). The *browse*, *search* and *similarity* services are used to locate elements in the storage.

REPOX is a single component responsible for data management with OAI-PMH protocol, harvesting, transforming and serving. It is also responsible for full-text harvesting.

The system is developed as J2EE web application and deployed within apache-tomcat server, but some components are implemented in other languages (python, OCAML) and integrated in an appropriate way (either as invoked binary programs or as separate servers).

The base document describes the architecture of the solution and the role of the components in the global system, and is accompanied by technical descriptions of the components used.

The document also revises technical specification described in D4.1 as still relevant with minor changes only (removed user groups, simplified browsing).

1. Introduction

This document is an revision and update of the Deliverable D4.2: - EuDML global system functional specification and design. It follows the original text whenever appropriate, while all updates are clearly marked.

Functional specification of the system as described in D4.1 requires only minor revision described in separate section.

This document describes the overall architecture and the key components of the EuDML software system. Together with its complementary appendix, it is intended as the general reference document for development purposes. Therefore it is expected that the document will be continuously updated to reflect the evolution of the system.

This is a concise document, focused on the main concepts of the EuDML architecture. More detailed descriptions of individual services and other components are provided in the companion document "Deliverable 4.4 – EuDML global system functional specification and design – Appendix".

1.1. Updates in functional specification

The system development has been driven by functional specification as described in D4.1. Over the time document has been the main requirement document for the system development, and only minor changes has been done to original requirements. Those changes are:

- Full list of registered users is not available to any user
- Concept of user groups has been abandoned as overcomplicated, and confusing

Rest of the requirements are still valid, but not all of them have been fulfilled. Especially in terms of browsing – browsing authors and dates is not available separately, but rather as the part of the advanced search and faceted search. That requirements are expected to be slowly developed after the project end, in the maintenance phase.

1.2. Design concepts and general architecture

The EuDML system implements a Service Oriented Architecture. The functionality offered to the user is backed by custom services, which encapsulate internal data persistence and access. The services are defined by their interfaces, and the implementation is, by design, hidden from the user, so that it can be easily replaced with an alternative one.

The presentation layer (web user interface) has been explicitly separated from the service layer, it is run in a separate application container and communicates with system services using remote invocation over network. This layer contains the logic for the presentation and security. It also implements REST services for accessing the data, which are nothing but an alternative view to the default system views.

Logic responsible for accessing and serving OAI-PMH (and more broadly – massive data transfer) is also separated into a module, independent from the User Interface module (REPOX). This module also uses only service interface and has no direct access to the underlying infrastructure.

This default deployment scenario is expected to be used throughout the system's lifespan. The separation of the presentation layer and the service layer ensures that user interface may use only service interfaces and never interferes with the internal implementation.

1.3. EuDML general architecture

The key components of the EuDML system are the Backend, the User Interface and the External Services Interface (Figure 1).

1.3.1. Backend

The *Backend* is a collection of services providing all the internal functionality of the system. This includes both basic repository services and more complex services, including some oriented towards the user interaction. The basic repository functionality covers metadata storage, content storage (document and extracted plain text for indexing), searching and relation browsing. It also includes technical aspects of the data processing, such as indexing documents.

Other services in the backend are responsible for user interaction (user registry, annotation service), metadata enhancements, as well as other internal aspects of the system.

The backend is accessed only by components of the EuDML system - web user interface module and by external services interface components – it is never accessed directly by the remote clients. Therefore to avoid unnecessary complication on the communication level, the decision has been made that security will be applied on the upper level – in the web interface module and in the external service layer.

For some software components there was no reasonable method to deploy them within the backend service container. One such services is gensim similarity, which runs as a set of separate processes and is invoked remotely by the backend component.

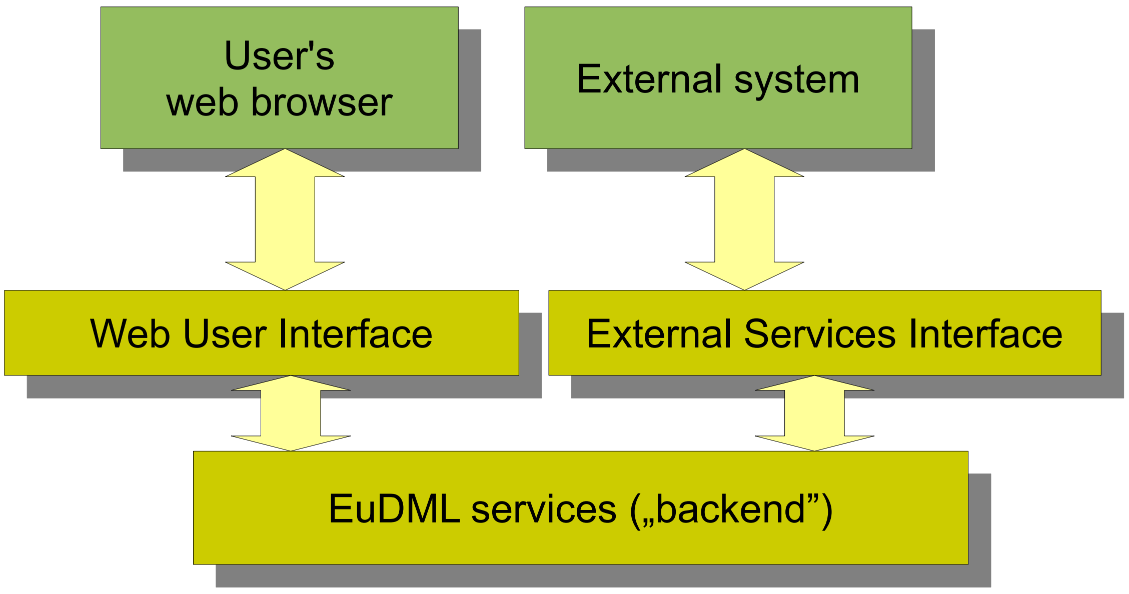


Figure 1: High-level system architecture view

1.3.2.User interface

The *User Interface* component is a web application, which is responsible for communicating with users using web browsers. It offers a REST service interface, a part of the External Service Interface. This component uses backend services to read/write the data, and exposes the metadata repository contents in a user friendly way. Security is applied in this layer, ensuring that users may perform only authorized actions.

1.3.3.External Services Interface

The *External Services Interface* is split into two parts. The first is carried out by the REPOX system, which manages the harvesting processes of the metadata and full-text provided by the data providers, together with the processes of data transformation and normalization. It also comprises an OAI-PMH server for sharing metadata with external parties. Also in this component security is enforced, so remote clients are properly authorized while performing actions.

The second part is the REST service API, which offers rich functionality for searching and accessing metadata for machine clients in a machine accessible way. For a description of the offered REST services refer to **D6.4: Web and Service Interface Implementation – Service Interface**. REST services are hosted within User Interface module.

1.4. EuDML as an XML record based system

EuDML is designed as a record-oriented system. The system manages a collection of records, where each record represents a single information entity uniquely identified by an EuDML identifier. An information entity will usually be a publication (such as an article or a book). Each record is a collection of XML metadata files and related content files.

This approach is the opposite of the typical relational-database driven approach, where records are decomposed into rows in multiple tables. In such systems maintaining the model is a difficult part of the development process, with a constant struggle between flexibility and performance of the queries.

The benefits of the EuDML approach include flexible metadata format support and easy adaptation to new requirements or extensions of the existing metadata formats. During the last 6 months of the project it has been obvious that it would be beneficial to adopt JATS as metadata schema, which is an extension of the NLM previously used. Thanks to the model used, the application transition required only limited resources and was possible even in such a late stage of the project without affecting stability or functionality.

It is also beneficial in a case of complex or rich metadata, when full relational decomposition would require a number of tables and rows to map a single publication.

1.4.1.Relations and indexes separated from the data

As the data is stored in the form of isolated records, the relationship between the records and the browsing indexes must be kept separate in dedicated services. This is a necessary trade-off for the benefit of an XML record driven architecture. Two core components of the system are used to maintain this task in EuDML: the general purpose Search Service and the Browse Service.

1.4.2. Workflow processing

The fact that the services are separated from the records themselves implies that XML source records must be processed to extract the necessary information for the indexes and the relations services. To this end a workflow processing engine is used. The processing engine is capable of iterative application of the specific workflow to a collection of the records in the system. As a result, the relations and search index entries are generated, but the same technology is used to perform metadata enhancement and clustering.

1.5. Communication layer

As a base communication layer for the system, a HTTP Invoke remote invocation mechanism has been chosen. To provide efficient and lightweight configuration of the communication layer the Spring Remoting framework is used. The System does not make assumption about mechanism used to invoke remote services, and HTTP invoke may be easily replaced by any other mechanism provided by Spring Remoting (RMI, WebServices etc.).

As architecture of the solution implies, that services are handled only internally, it is easy to adopt custom remoting protocol, and it does not affect the clients, as they use dedicated layer to access data.

2. Overview of the technology used

All the existing technologies to be used in EuDML have been carefully selected not only to perform well but also to be both open-source and open-licensed. The following set of technologies has been chosen to support the EuDML solutions.

- *Java* – the system is Java based, developed and tested with SUN/Oracle Java v. 1.6
- *PostgreSQL* is used as a relational database engine
- *JDBC* and *Spring JDBC* are mostly used in low level services to access database storage
- *Hibernate ORM* - used by some services for persistence layer
- *Webapp* - core components are packaged as standard Java webapps, using WAR packaging
    - *Tomcat webapp container* is used for operational running
    - *Jetty webapp container* - used extensively in the development process
    - *Apache http server* - used as a proxy in complex deployments
- *Spring framework* (v2.5) - used as a core framework of the project
    - *IOC container* - used to define deployment and software component orchestration
    - *Spring remoting* as core communication layer between components (configured to use HTTP invoke as transport layer)
    - *Spring MVC* is a core technology of the web UI
    - *Spring security* - manages UI security and login process
    - *Spring integration framework* is used as a base for workflow processing service
- *Solr search engine* - YADDA search engine is based on Solr
- *Apache Lucene* - used by YADDA similarity service
- *Neo4J* – a graph database used by new implementation of the browse service.
- *YADDA framework* - core services are either plain YADDA services, or are developed to fit into YADDA framework
    - YADDA services (storage/search/browse/process/user db)
    - YADDA service architecture & remoting
- *REPOX Service* (harvesting data from content providers)
    - *Google Web Toolkit* – Java software development framework for user interfaces
    - *Xalan-Java* – an XSLT processor for transforming XML documents
- *Apache Maven* – software project management and comprehension tool
- *Python* – some components of the system are written in python. Among them most important is GensimEuDML – a similarity engine
- *OCAML* – maxtract has been written in OCAML and is delivered in a binary format.

3. EuDML services and components

The EuDML system follows the Service Oriented Architecture paradigm, and consists of a number of inter-communicating services. The basic set of EuDML services is introduced in this section. In the future the set can and is intended to be extended to support new or enhanced functionalities of the system.

For detailed description of individual services consult the Appendix to this document.

Figure 2 depicts the general dependencies between the services.

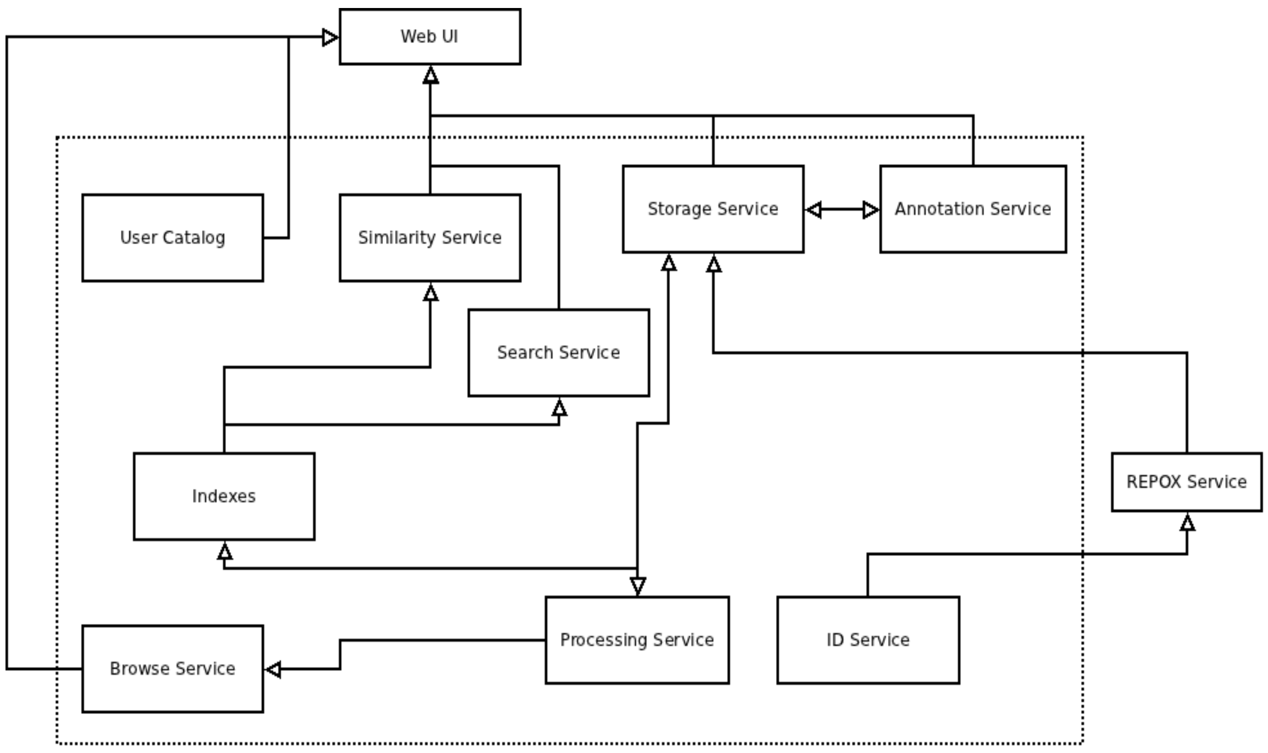


Figure 2: EuDML services interdependencies diagram.

3.1. Workflow Processing Service

As the system is generally record-oriented and XML-based, there is a need to perform iterative processing of the records within the system. This processing typically is:

- indexing data for a search service
- building specific relations in other services (e.g. browse service)
- performing metadata enhancement and clustering

The YADDA Processing Service is used as a workflow service.

The Processing Service orchestrates and applies the EuDML workflows to the data. It operates on the following basic concepts: The workflow defines a number of nodes and the data flow. The process is the operation of applying selected workflow to a number of messages. A message is an atomic processing unit, which is passed into a workflow. The Workflow nodes are operations applied to messages, and the flow defines the order of the node execution. During processing, messages are passed between nodes according to the workflow definition. The workflow is defined by channels linking nodes and routers, which may alter the flow (conditional selection of channel, starting parallel execution on different channels etc.) Typically, a message is a publication record, and nodes are single operations - like building index document or extracting text from PDF files. A message may be transformed by nodes or consumed (e.g. stored into persistent storage).

Figure 3 depicts the general processing concept for an example workflow. It starts with an iteration over a store. Messages are then passed through *processing nodes* and possibly *routers*. The input and output of each node can be of different type but in any given flow the input of a subsequent node must match the preceding node's output. The final messages are stored by *writers* in some kind of stores.

Workflows are defined statically in Spring configuration files.

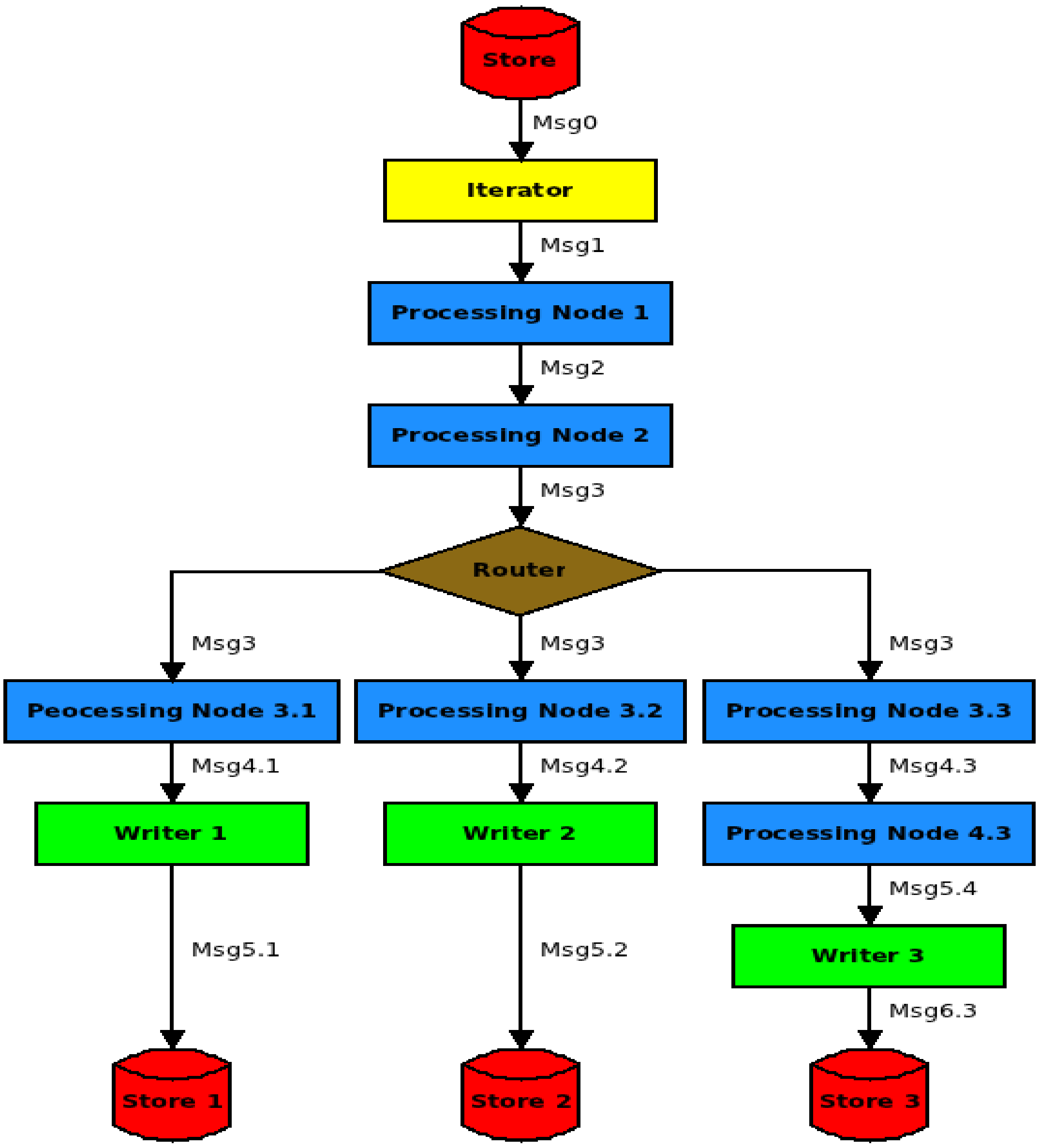


Figure 3: General processing workflow

The most important workflows used in EuDML are:

- indexing workflow – indexing and processing all the relations for new or updated publications
- enhancement workflow – a workflow to apply all possible enhancements to the publications, including metadata and text extraction, building external links etc. This workflow is described in detail in (**D7.4: Toolset for Image and Text Processing and Metadata Enhancements — Final Release**)

3.2. Content Aggregation Service

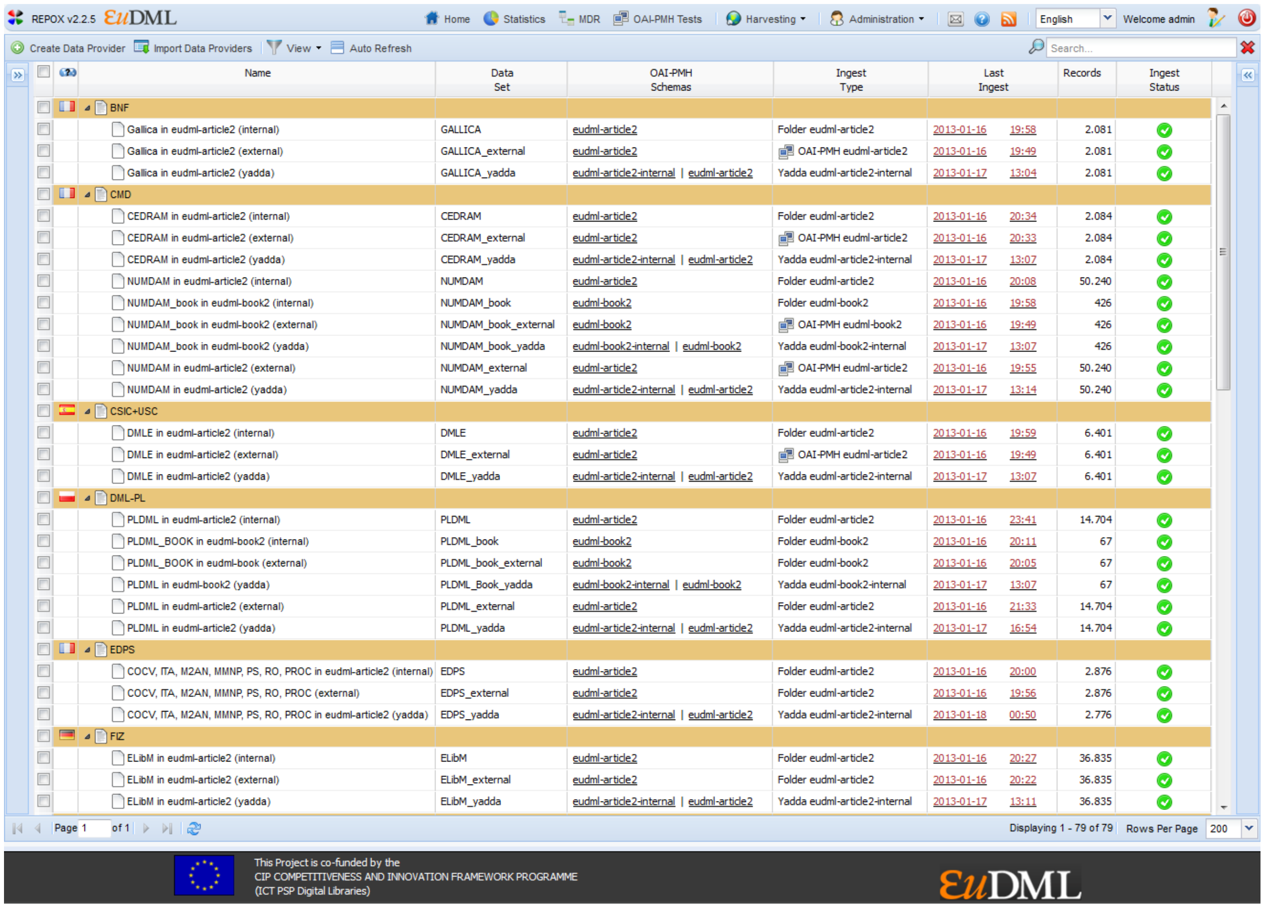
The Content Aggregation Service uses REPOX technology to manage the harvesting processes of the metadata and full-text provided by data providers, as well as the processes of data transformation and normalization.

In brief, REPOX provides the following main functions:

- Registration of data providers, their collection descriptions (a single data provider can make more than one collection available to EuDML), and the configurations for the harvesting of the related metadata and full-texts.
- The automatic and manual harvesting of the metadata by OAI-PMH (according to configurations and options provided by the data providers and the decisions of the administrators of the central service) or by HTTP or FTP. Support is provided for multiple metadata formats (OAI-DC is assumed by default, but any other format is also possible). The harvesting of the full-text is expected to be done through HTTP or FTP.
- The delivery of metadata to external partners (in this way, records that have been enhanced by the EuDML specialized services can be returned to their original providers).
- Monitoring of the quality of service of the OAI-PMH servers, including statistics.

The REPOX service has its own graphic user interface (example in Figure 4) to be used by humans.

REPOX also provides to the outside a set of REST services for data providers, data sources and records.



Name	Data Set	OAI-PMH Schemas	Ingest Type	Last Ingest		Records	Ingest Status
BNF							
Gallica in eudml-article2 (internal)	GALLICA	eudml-article2	Folder eudml-article2	2013-01-16	19:58	2.081	✓
Gallica in eudml-article2 (external)	GALLICA_external	eudml-article2	OAI-PMH eudml-article2	2013-01-16	19:49	2.081	✓
Gallica in eudml-article2 (yadda)	GALLICA_yadda	eudml-article2-internal | eudml-article2	Yadda eudml-article2-internal	2013-01-17	13:04	2.081	✓
CMD							
CEDRAM in eudml-article2 (internal)	CEDRAM	eudml-article2	Folder eudml-article2	2013-01-16	20:34	2.084	✓
CEDRAM in eudml-article2 (external)	CEDRAM_external	eudml-article2	OAI-PMH eudml-article2	2013-01-16	20:33	2.084	✓
CEDRAM in eudml-article2 (yadda)	CEDRAM_yadda	eudml-article2-internal | eudml-article2	Yadda eudml-article2-internal	2013-01-17	13:07	2.084	✓
NUMDAM in eudml-article2 (internal)	NUMDAM	eudml-article2	Folder eudml-article2	2013-01-16	20:08	50.240	✓
NUMDAM_book in eudml-book2 (internal)	NUMDAM_book	eudml-book2	Folder eudml-book2	2013-01-16	19:58	426	✓
NUMDAM_book in eudml-book2 (external)	NUMDAM_book_external	eudml-book2	OAI-PMH eudml-book2	2013-01-16	19:49	426	✓
NUMDAM_book in eudml-book2 (yadda)	NUMDAM_book_yadda	eudml-book2-internal | eudml-book2	Yadda eudml-book2-internal	2013-01-17	13:07	426	✓
NUMDAM in eudml-article2 (external)	NUMDAM_external	eudml-article2	OAI-PMH eudml-article2	2013-01-16	19:55	50.240	✓
NUMDAM in eudml-article2 (yadda)	NUMDAM_yadda	eudml-article2-internal | eudml-article2	Yadda eudml-article2-internal	2013-01-17	13:14	50.240	✓
CSIC+USC							
DMLE in eudml-article2 (internal)	DMLE	eudml-article2	Folder eudml-article2	2013-01-16	19:59	6.401	✓
DMLE in eudml-article2 (external)	DMLE_external	eudml-article2	OAI-PMH eudml-article2	2013-01-16	19:49	6.401	✓
DMLE in eudml-article2 (yadda)	DMLE_yadda	eudml-article2-internal | eudml-article2	Yadda eudml-article2-internal	2013-01-17	13:07	6.401	✓
DML-PL							
PLDML in eudml-article2 (internal)	PLDML	eudml-article2	Folder eudml-article2	2013-01-16	23:41	14.704	✓
PLDML_BOOK in eudml-book2 (internal)	PLDML_book	eudml-book2	Folder eudml-book2	2013-01-16	20:11	67	✓
PLDML_BOOK in eudml-book (external)	PLDML_book_external	eudml-book2	Folder eudml-book2	2013-01-16	20:05	67	✓
PLDML in eudml-book2 (yadda)	PLDML_Book_yadda	eudml-book2-internal | eudml-book2	Yadda eudml-book2-internal	2013-01-17	13:07	67	✓
PLDML in eudml-article2 (external)	PLDML_external	eudml-article2	Folder eudml-article2	2013-01-16	21:33	14.704	✓
PLDML in eudml-article2 (yadda)	PLDML_yadda	eudml-article2-internal | eudml-article2	Yadda eudml-article2-internal	2013-01-17	16:54	14.704	✓
EDPS							
COCV, ITA, M2AN, MMNP, PS, RO, PROC in eudml-article2 (internal)	EDPS	eudml-article2	Folder eudml-article2	2013-01-16	20:00	2.876	✓
COCV, ITA, M2AN, MMNP, PS, RO, PROC (external)	EDPS_external	eudml-article2	Folder eudml-article2	2013-01-16	19:56	2.876	✓
COCV, ITA, M2AN, MMNP, PS, RO, PROC in eudml-article2 (yadda)	EDPS_yadda	eudml-article2-internal | eudml-article2	Yadda eudml-article2-internal	2013-01-18	00:50	2.776	✓
FIZ							
ELibM in eudml-article2 (internal)	ELibM	eudml-article2	Folder eudml-article2	2013-01-16	20:27	36.835	✓
ELibM in eudml-article2 (external)	ELibM_external	eudml-article2	Folder eudml-article2	2013-01-16	20:22	36.835	✓
ELibM in eudml-article2 (yadda)	ELibM_yadda	eudml-article2-internal | eudml-article2	Yadda eudml-article2-internal	2013-01-17	13:11	36.835	✓

Page 1 of 1

Displaying 1 - 79 of 79 Rows Per Page 200

This Project is co-funded by the
CIP COMPETITIVENESS AND INNOVATION FRAMEWORK PROGRAMME
(ICT PSP Digital Libraries)

Figure 4: REPOX Graphical User Interface

3.3. Storage Service

The EuDML Storage Service (EuDML store) is the central part of the system. It encapsulates the logic of the record format and layout and implements it using lower layer services (currently YADDA MD-Storage and YADDA Archive). Its main role is to store and furnish both metadata and content (both full-texts and plain-texts, typically being PDF and text files) of the information entities maintained by the system. The internal structure of the service assumes that each entity consists of a number of parts, and both metadata, and contents can be stored in many concurrent formats. For example, almost all items in EuDML will have their corresponding metadata in Zentralblatt Math. So the Storage Service keeps its ZBMath metadata alongside with the metadata harvested from the content provider's OAI-PMH server and converted to NLM format by the Content Aggregation Service.

The main reason to store the items' contents (payloads) is to have them readily available for subsequent iterative mass-processing, e.g. text mining. The second reason is to keep backup copies for redundancy and possible future long term preservation purposes. The EuDML system is currently not intended to serve publication contents (full-texts) from its stored copies but instead links to the original sources at the data provider's site. The only exception to this rule is that EuDML serves accessible content versions after enhancements to make it accessible for disabled users. Technology to do this is offered by WP10 an is described in (**D10.4: Enhancement and Deployment of Final Toolset**)

In a number of cases it may be useful to keep full-texts in different formats. The EuDML system can at the same time keep parts of a publication in PDF, and other (or actually the same) parts in plain-text or XML structures extracted by specific tools. If a data provider supplies EuDML with e.g. LaTeX source files, they may be stored as well in order to avoid the need to extract the structure and details from the rendered files. Finally, keeping the original and processed full-texts allows EuDML maintainers and developers to gradually improve processing tools in order to obtain better quality of the derived data.

Other EuDML services communicate with the Storage Service in order to obtain data to be enhanced, presented (displayed) or to store data after being enhanced. At a lower level, persistence of the data is assured by the Storage Service by a PostgreSQL database.

3.4. User Directory Service

The User Directory Service is dedicated to store information about users, groups and relations between them. This service is also used for storing additional information about users e.g. email address, biography etc. It can also authenticate users using pluggable token/credential mechanism.

The stored information concerning users includes:

- user domain (the User Registry implementation supports multiple domains)
- user identifiers (from different namespaces)
- user attributes
- user flags
- user credentials (used in user authentication process)
- groups to which user belongs

The stored information concerning groups is:

- group domain
- group name
- group roles
- whether the group has a parent group (from which it inherits roles)

In EuDML there is a custom service, EuDML UserDB, which uses a User Directory to authenticate users. This EuDML UserDB provides user/password authentication implementation over the generic authentication mechanism. It may also be used to keep other information provided by the users e.g. their email address, biography etc.

3.5. Search and Browse Services

There are two aims that the Search Service serves. Firstly, it is used as a document indexer – it stores documents' metadata (title, authors, abstract...) and full-texts in the index. Secondly, as a document searcher – it provides an API to search the index for documents corresponding to a specified query (using provided Java query model).

The Search Service is based on Apache Solr (Figure 5). It serves as a simplified outer layer (completely hiding the Solr API and configuration) which provides:

- simple Java API for document indexing and searching
- simple XML (and Java) index metadata configuration (index fields declaration)
- fully automatic Solr instance configuration and initialization

It uses many of the Solr advanced capabilities and extends them through the mechanism of plug-ins. Its main features include:

- advanced Solr search functions, such as: Faceted Search, Filtering, Dynamic Fields

- Solr features with improved or customized processing logic, such as: Sorting, Highlighting, Wildcard Search
- new features, designed to improve search in such cases as: authors name search, full-text search
- special plug-in for mathematical formulae (in MathML format) indexing and searching

The implementation used uses embedded Solr instead of stand-alone server, which makes deployment and the management of the system easier.

The Browse Service is a supporting service, responsible for dealing with relations between objects. This service allows to define relations, index them for fast access, and query the data in these relations. This service is used to provide effective browsing over data objects within the storage. It also allows efficient querying of aggregated data, for example fetching the count of objects fulfilling specific criteria, by maintaining lazily materialized aggregated data views. The Browse Service supports numeric, boolean and timestamp data as well as case-sensitive and insensitive text strings and string arrays. The browse service used is a Yadda2 browse service and it is implemented using the Neo4J graph database. The browse service is used for:

- browsing journals
- browsing collections
- browsing categories
- browsing journal contents

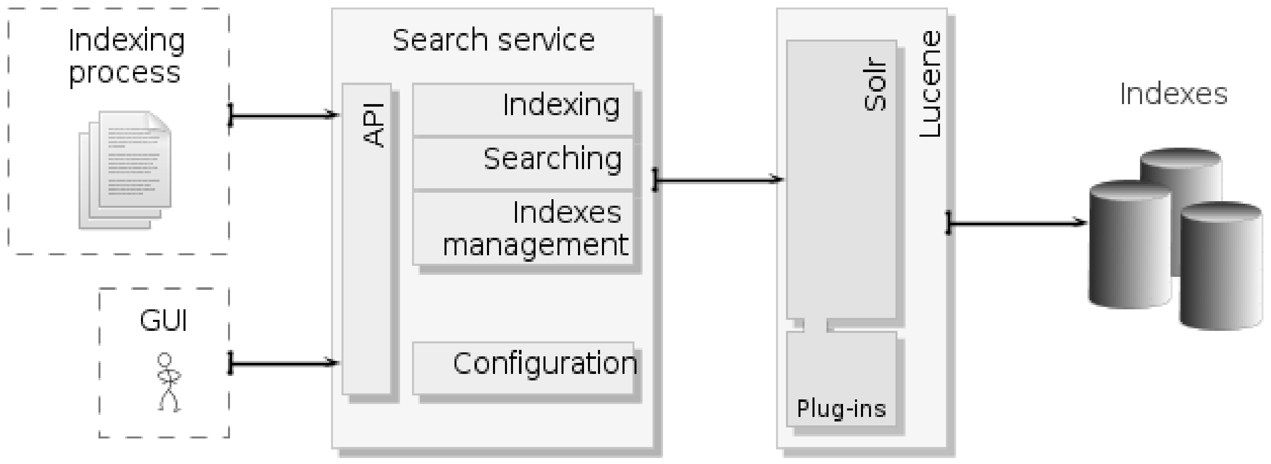


Figure 5: Search Service Implementation

3.6. Similarity Service

The Similarity Service detects similarities between text documents and it is used to offer “similar documents” functionality in UI. A given document, either already stored in the service or specified in the request (in the latter case, document's text must be contained in the query), is used as a basis for a similarity query to find other similar documents stored earlier in the service. What exactly “similar” means depends on the specific implementation.

EuDML has evaluated two existing implementations of the service: YADDA similarity and Gensim and finally it has been decided that Gensim results are better.

Gensim is written in python, with some native C++ code, and it would be expensive to deploy this service within a backend application container. Therefore Gensim runs as a separate process and offers a service interface over the network, which is used by the backend service set.

3.7. Annotation Service

The Annotation Service is dedicated to store pieces of information related to a particular target item having a URI. In EuDML every identified information entity can be annotated. This allows users to express their observations concerning any given item and store them in order to be shared with other, possibly future, users interested in them.

The annotation bodies are stored in YADDA MD-Storage and annotation relations (ownership, target, state, visibility, language...) in Sesame RDF storage. Serialization is realized via Xstream. The annotation service is used not only to comment, but also to request moderation, suggest a topic for the item etc. Reading lists are implemented using the annotation service. For detailed description of the annotation component refer to **D9.3 Annotation Component Integration**.

3.8. ID Service

The ID Service is responsible for managing assigned identifiers (in the form of URIs) and ensuring that all the identifiers are unique and assigned only once. The service manages and keeps track of the identifiers assigned to the objects.

It is an important functional requirement that all information entities in the EuDML system have persistent, unique, long-term identifiers. It is assumed that if a publication has been visible to the users under a certain id for even a short period of time, it may have been linked from external pages or otherwise referred to and thus this id cannot disappear or point to another resource in the future. As the EuDML data space is not a static collection but a living system which grows all the time with new data being constantly imported, those identifiers must be assigned with care. Imports often are performed multiple times with the same data, and it has to be assumed that a new version of an existing object may appear in source repository. The same information entity can also be imported independently from multiple sources. To handle these cases the ID management service has been developed.

The service operates on the assumption that each document has a number of identifiers of various types. These identifiers may be well recognized unique identifiers (like DOI, Zentralblatt) or repository specific ids. The ID service is called whenever a document needs an identifier assigned (i.e. it is imported into EuDML system). At this moment, the service is provided with a complete list of all known document identifiers, in form of pairs (type, value). The service checks if at least one of the identifiers is already known to the system. If so, all identifiers are added as equal to the known one and the already assigned EuDML identifier is returned. If none of the identifiers is known yet to the service, then a new identifier is assigned, and the provided identifiers are stored as equal to newly created identifier.

Service data is persisted within a database and is a key to proper management of identifiers. In the case of re-importing all the data into EuDML (e.g. to a new instance of the system) it is sufficient to migrate this service database to ensure that during the import process all the publications will have the same identifiers, as they had in original system.

The identifier assignment policy is carried out according to the EuDML identifiers assignment rules. Identifiers start at 1000 and each new identifier is generated as a simple increment of the latest identifier assigned. This protects the 'golden identifiers' (short ones) for special purposes, and uses the namespace efficiently, generating identifiers that are as short as possible.

3.9. User Interface

The EuDML User Interface is a typical Java based web application, developed with use of widely acknowledged Open Source tools including JSP, Spring Framework, Tiles and Apache Commons tools. In addition to these standard tools, the EuDML UI makes use of the YaddaWeb2 framework developed by ICM, which allows easy implementation of repository frontends. The EuDML UI accesses the EuDML Repository through the HTTP Invoke remoting protocol. The following services of the Repository are used:

- EuDML store - which serves metadata and contents
- Browse Service - which serves lists of journals
- Similarity service – which identifies collections of similar items
- Search Service - which is responsible for performing searches
- User Registry Service - which stores user accounts and is used to authenticate users
- Annotation Service - which stores user created content

Authorization and Authentication in EuDML UI is based on the Spring Security Framework. The framework is customized in order to use EuDML specific authentication data source (User Registry Service). Spring Security gives off-the-shelf support for features like OpenId integration and authentication through persistent cookies. Since all infrastructure of EuDML is maintained by the participants of EuDML project, there is no need to implement service-to-service security between backend and User Interface. Because of that, security enforcement is performed only in the EuDML UI application and on the external service interfaces.

3.9.1. YaddaWeb2 framework

YW2 (YaddaWeb2, formerly YaddaWebLite) is a set of components supporting easy implementation common repository frontend functionalities:

1. Searching for publications
2. Page lists retrieved from different sources
3. Displaying details of particular publications in convenient way
4. Filter and enrich text entities from the publication metadata to be displayed on the web page (highlight search texts, convert domain specific formatting into HTML)
5. Consistently handle system errors, access violations and trials to access non-existent resources.
6. Display AJAX tree with hierarchy of publications

YW2 is a Java framework intended to be used along with Spring and Spring MVC frameworks. It is distributed as a set of JAR packages which may be included in a particular Java Web application. Having included YW2 in an application, it is possible to make use of concrete classes or components to implement particular functionalities. In most cases it will be necessary to implement some domain-specific environment of particular components like datasources, metadata deserialization components, view templates etc. Since all components belonging to YWL are loosely coupled, it is possible to use only some of them. A typical YaddaWebLite based application may look like that in Figure 6.

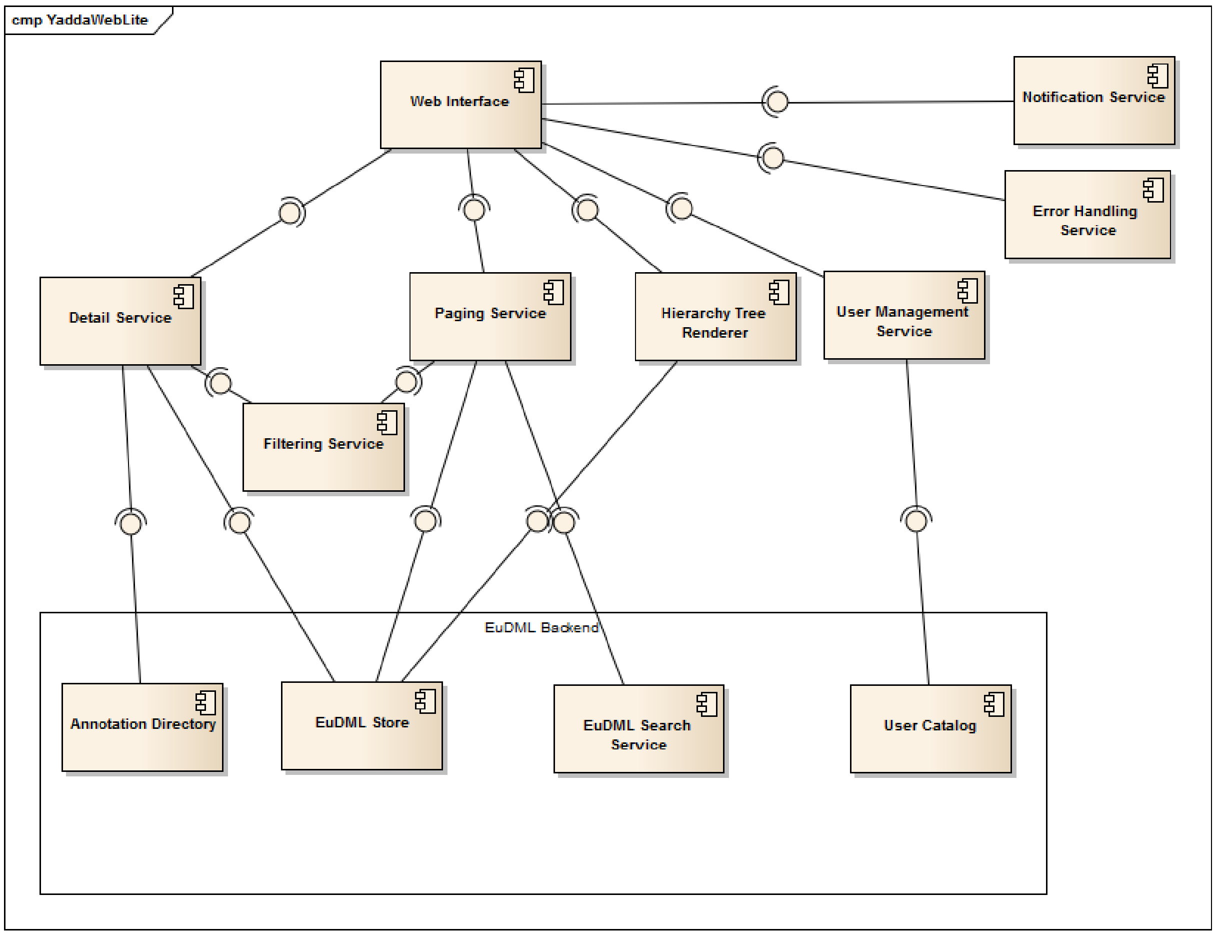


Figure 6: YaddaWebLite with EuDML Backend Services

3.10.External Services Interface

The External Services Interface is intended to provide data and service-level communication between EuDML and third party applications or services. A set of external interfaces contains:

- REST services for querying publications
- OpenSearch REST service
- OAI-PMH server
- services to query Ids of the documents
- services to query citations among documents
- other services, as described in **D6.4: Web and Service Interface Implementation –**
- **Service Interface**

The services has been proven useful and are used with success by a number of the remote systems.

4. Metadata and data handling

For a digital library, proper handling of metadata and data is, obviously, a crucial point of its design. EuDML handles object's metadata, content (associated files) and plain-text. Each object's metadata are imported from content providers, converted, aggregated, stored and maintained locally together with the related plain-text. The other object's content files (e.g. PDF) will be downloaded from their providers and stored within EuDML for subsequent processing, but the system is not intended to serve them to the users, as long as they are available in from the original provider's site. It may be possible to serve the content from the EuDML archive, though, if the original site is inaccessible.

4.1. EuDML Identifier (eID) format

One of the goals of the EuDML is to offer standard, long living identifiers and URLs for the objects (publications), in order to offer a unified way of online referencing of mathematical works. These identifiers and URLs must be short enough to be usable and must share a common format and assignment strategy.

In EuDML IDs are URNs with custom namespace: **eudml**. This way we offer reasonable standard format of the identifier, which is the custom string will be composed of two parts:

- object type
- object identifier (an string in format dependent on the object type)

We have decided to incorporate object types into identifiers. With this, we would like to preserve document identifier space and not to waste it on other, less important objects, like annotations.

EuDML identifiers have the following format:

```
urn:eudml:<type_id>:<object_id>
```

Specific objects have special rules for the proper id format, as described in the table below:

Object type	type_id	Identifier format	example
Publication	doc	For objects with content ("leafs") a numeric string as assigned by ID service	`urn:eudml:doc:12345`
Annotation	an	<random uuid>	`urn:eudml:an:63ce7b10-01f0-11e0-a976-0800200c9a66`
User	User	Random	`urn:eudml:user:63ce7b10-01f0-11e0-a976-0800201c9a66`
Bookshelf	bookshelf	Random	`urn:eudml:bookshelf:63ce7b10`

Note that only document and annotation type identifiers are abbreviated. We have decided that this is optimal, as in fact only those two types of objects are typically addressed externally, and those identifiers are expected to be passed outside of the system, so short notation is an advantage for external system users.

4.2. Short form

The short eID form is an URN without the namespace (**urn:eudml:**) prefix so it only has a type string and a proper identifier. The short form will be used inside the system and in URLs when appropriate.

4.3. URNs and URLs

As the UI will offer stable addressing, a persistent mapping (URN->URL) has been defined as the URL of the document in EuDML system. This mapping can be queried with appropriate REST service (See D6.4, service *All pointers*), and for the publications it is simple: EuDML ID in form urn:eudml:doc:<number> is mapped to: **https://eudml.org/doc/<number>**. This mapping is constant, and URLs of the objects are not supposed to change during service lifetime.

4.4. Document identifiers assignment

As the same publication may appear in multiple providers collection and may be imported multiple times to the system, we must track identifiers carefully. The namespace of the identifiers has no limit, but users expect that the same publication has the same EuDML identifier regardless of the import source and time.

To keep track of the identifiers we have chosen id assigning strategy based on other document identifiers.

Each imported document has one or more identifiers known during the import time, including provider's internal identifier. Each of those identifiers is described as a pair of text strings (identifier type, identifier value), which are assumed to be unique. Other than providers' identifiers, there are cross-system identifiers like DOI, Zentralblatt or AMS citations. Whenever a document is imported into a system, an ID service is consulted and all known documents identified are passed as a request. If the corresponding EuDML ID for at least one of the identifiers is known, then this EuDML ID is returned, and all identifiers passed to the request are assumed to be equal to this identifier. Otherwise, if no identifier is known to the ID Service, then a new identifier is assigned as a first free identifier (i.e. next in sequence), and all passed identifiers are stored as equal to this one. The ID Service database contains complete information on all assigned IDs, and ensures that if all the contents are imported again, then the same identifiers will be assigned to the same publications.

4.5. Duplicate handling

As providers collections overlap it was inevitable that problems of duplicate detection and handling would arise and must be resolved. Duplicate handling is very special, due to requirements of the reference library. The problem is quite subtle: as the duplicate detection algorithm improves, at some point two existing records may be found to be duplicates of a single publication. In such case there is a need to join them. On the other hand users may already have links to each of the duplicate items, and removing such items would violate the principles of the reference library. To solve this, a new item is created and all existing duplicates are merged into this item. It obtains a new EuDML ID, while both old records are marked as duplicates of the item, and from this moment all requests to these items are redirected to the newly created record. If the decision about merging was wrong, then old records are restored under their original Ids, and incorrect records are replaced with information about the event and links to the original elements. All this is done in the storage level, and allows handling of the deduplication process even in complex scenarios.

4.6. Stored metadata

As the record may appear in multiple source forms, within the system record is stored in multiple formats, including all source forms. This is important, as we know that it is not always clear, that we can choose best version of the record, and therefore merging of the metadata during conversion to NLM format may be desired. Even if we have only one provider for the record, we still often have a Zentralblatt record version, often containing extra metadata not available on the original source.

To unify this, within the store all source metadata are stored, and during import an EuDML metadata (in NLM format) is generated according to specific conversion rules. Those rules are pluggable and may be adopted according to the particular situation. This generated NLM is the so called 'base NLM' record, which contains the best possible metadata converted from all source providers and merged into a single record. This format does support storing full-text within NLM.

4.7. Formats used

Metadata are kept in EuDML in XML based formats. Every format has its own XML Schema. All provider source format records are stored within EuDML.

The main operational format is JATS, a successor of NLM format, adapted to special requirements of the service. It is described in the full detail in Deliverables D3.2 with further modifications described in D3.6. This format is the base for all operations and displaying the data to the user through the UI. Aside from NLM, the Zentralblatt MATH internal format is used to deal with Zentralblatt data.

For storing mathematical expression, MathML is primarily used. As an alternative to MathML, the service also keeps mathematical expressions as LaTeX code (as a special tag inside NLM files) because this format (though not XML-based) is widely used by mathematicians and people using mathematics in their work.

For full-text items, the service mainly uses PDF files (both OCR-ed and digitally born) as this is the de-facto standard in digital libraries. Technically there are no problems with keeping other formats as well (e.g. DjVu, TIFF).

4.8. Handling contents

Content (metadata, plain-text, and full-text) is placed in Storage Service by REPOX Service in available source format. During this process, REPOX carries out conversion from content providers' specific formats to the JATS format used in EuDML. In this step, persistent identifiers are generated by the ID Service. The converted JATS file contains no plain-text and, if plain-text is present, it is stored separately. This is done for future processing efficiency, as JATS are used constantly, while plain-text files are used only once (during the indexing process) and may have significant size.

In the next step, metadata are enhanced with ZBMath metadata by lookup either by ZbMath identifier or, if that identifier is not known, using author, title etc. in the query. If ZBMath metadata is present, it is stored as another source metadata and NLM is generated once again, containing data from all sources, according to the merging procedure.

Later, if no plain-text is provided, extraction from PDFs is performed (it can be via OCR or another techniques in case of digital-born PDFs). In the next stage, plain text is enhanced with MathML, which data are used intensively by whole service. The latter step can be combined with PDFs extraction; it depends on the data that are available.

The two latest steps may be applied once again, if there is any hope of obtaining better results. The typical example is when the work is not yet present in ZBMath, but becomes available after a few months, upon which event it will be downloaded.

After finishing this stage, the system contains a full set of required data and internal processing starts: the Search Service indexes plain-text and metadata and other relations stored in the system are filled in with data. In this step, the similarity index may also be generated or updated.

In the next step, the matching references process tries to identify references in the documents with items. This is finally used to make links between references and objects.

As data comes from many independent sources there may exist duplicates in the final collection that forms EuDML. The last step is to find and merge duplicates. This is done at the end of handling contents because information gathered during the previous steps can influence this action.

Detection of duplicates is an ongoing process, even after data is presented to the public. If such late duplicate detection occurs (possibly supported with expert knowledge), then old references will redirect to one, unified record, so references to all duplicates will be supported.

4.9. Metadata enhancement

Metadata enhancement tasks shall be implemented by a number of partners, using a number of different technologies. Therefore, it is important to design a flexible enhancement architecture.

Each enhancer is implemented as a set of workflows that are to be sequentially run in the backend by a resource manager. Each process consists of a number of the so-called *processing nodes*. The first node in the workflow (*source node*) typically iterates over contents of a repository: it fetches items one-by-one and feeds them to the next node (or nodes) in the workflow. The terminal node (or nodes) in the workflow (*writer node*) typically stores incoming items in a repository.

Processing nodes are written in Java as implementations of certain interfaces (such as IProcessingNode<I,O>), and workflows are defined using Spring framework configuration files (XML format).

Figure 7 Shows an example of an enhancer implemented using two workflows. Note that workflows are allowed to reuse actions.

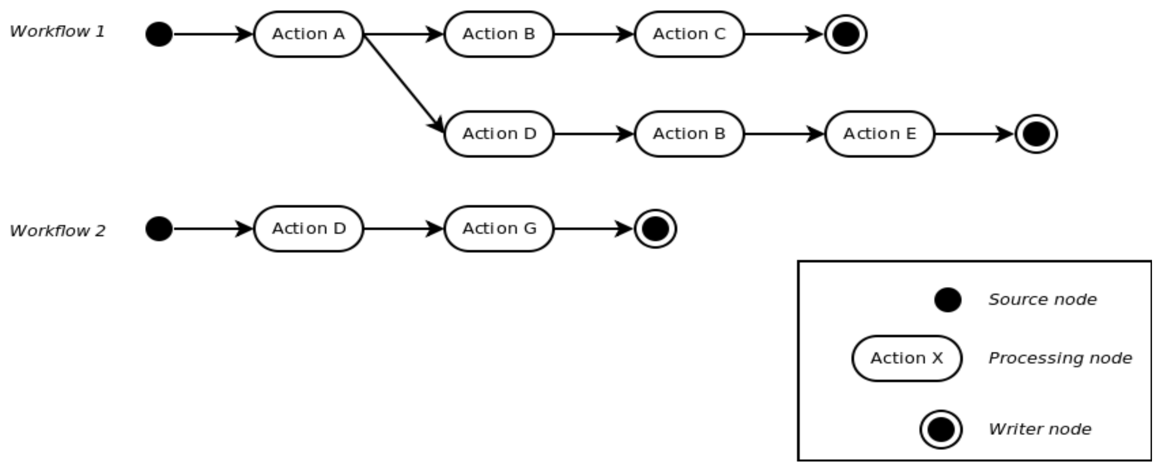


Figure 7: EuDML metadata enhancement workflow.

The complete list of the enhancer components together with detailed description of the enhancement workflow may be found in deliverables: D4.4 The EuDML Information Life Cycle Process, and D7.4: Toolset for Image and Text Processing and Metadata Enhancements — Final Release.

5. Practical experience from implementation

The architecture described in the previous sections has been successfully implemented, and during the project no major conceptual change has been required. Practical experience proved that most of the choices have been justified and the resulting system is manageable and extendable despite its complexity. We are more than satisfied with overall performance and flexibility, yet adding new features still requires a significant amount of work. On the other hand, maintenance of the system requires only minimal manpower, as most of the tasks are automated so future maintenance will not be expensive and is possible without significant resources.

The service oriented architecture was a good choice both for the system and for the development model of the project conducted by a number of distributed partners. One of the biggest efforts was not to deliver the components, but to integrate them into a single system. Separating and encapsulating delivered components as services or nodes in a workflow system allowed developers to achieve success in this matter. It was especially important during the quality assurance phase. As some components initially did not offer production-quality, thanks to the architecture shape it was easy to test, validate and improve them.

The document oriented architecture allowed introducing a complex metadata enhancement workflow, supporting easy maintenance of different versions of the document. This decision was also justified in the context of multiple metadata providers, as it allowed easy implementation of object merging. This choice also allowed easy modifications and updates in some critical phases, and we consider this one of the important achievements in this project.

Some services have not been worth the investments made. For example the browse service finally appears to be redundant to the search service capabilities and it is expected to be marginalized and removed in future releases.

Overall experience from implementing EuDML with the architecture described in this document is very positive, and we expect to follow this path in future work.