ICT-PSP Project no. 270905

LINKED HERITAGE

Coordination of standard and technologies

for the enrichment of Europeana

Starting date: 1st April 2011

Ending date: 31st October 2013

Deliverable Number:	D 5.3
Title of the Deliverable:	Metadata gateway
Dissemination Level:	Public

Contractual Date of Delivery to EC:	Month 12
Actual Date of Delivery to EC:	June 2012

Project Co-ordinator

Company name :	Istituto Centrale per il Catalogo Unico (ICCU)
Name of representative :	Rosa Caffo
Address :	Viale Castro Pretorio 105, I-00185 Roma
Phone number :	+39.06.49210427
Fax number :	+39.06. 06 4959302
E-mail :	rcaffo@beniculturali.it
Project WEB site address :	http://www.linkedheritage.org

Context

WP 5	Technical Integration
WP Leader	NTUA
Task 5.1	Gateway
Task Leader	NTUA
Dependencies	

Author(s)	Kostas Pardalis, Nasos Drosopoulos
Contributor(s)	Nikos Simou
Reviewers	Regine Stein, Gordon McKenna
Approved by:	

History

Version	**Date**	**Author**	**Comments**
0.1	20/03/2012	Nasos Drosopoulos	First Draft
0.2	10/04/2012	Kostas Pardalis	Second Draft
0.3	18/04/2012	Nasos Drosopoulos	Third Draft
1.0	11/06/2012	Nasos Drosopoulos	Final Version

TABLE OF CONTENTS

INTRODUCTION

This report documents the technological solution for the establishment of the Linked Heritage Metadata Gateway (Task 5.3). The service complements the project's technology platform (Task 5.1, Deliverable D5.1) offering the technical components necessary for metadata remediation and in particular for the delivery of metadata to Europeana (ESE/EDM compliant[1]). Following the modeling requirements of WP2, WP3 and WP4 and, close collaboration with the Europeana Office ensures interoperability of the Linked Heritage Technology Platform with the Europeana Ingestion infrastructure and the Europeana portal. The Linked Heritage Metadata Gateway handles and remediates XML metadata for the execution of the ingestion plan and RDF resources for prototyping of semantic web-enabled metadata services.

BACKGROUND

WP5 'Technical Integration' is responsible for the semantic alignment of metadata schemas which content providers use to annotate their items, to a common, well-defined, machine understandable schema. This in turn will allow for the ingestion of aggregated content in the Europeana portal.

Europeana provides integrated access to digital objects from the cultural heritage organizations in all the nations of Europe. It contains material from museums, libraries, archives and audio-visual archives with the aim of making Europe’s multicultural and multilingual riches discoverable in a common on-line environment. To do this Europeana harvests and indexes the descriptive metadata associated with digital objects. As there is no universal metadata standard used in the participating domains, a set of metadata elements (ESE – Europeana Semantic Elements) was developed. This allowed a common set of information to be supplied which supports the functionality desired by the user, and operation of the underlying system.

To provide metadata in the specified format, it is necessary for contributors to map elements from their own metadata format to ESE. Also it is necessary, to enable machine readbility, for a normalisation process to be carried out on a range of elements. In the initial implementation of the Europeana infrastructure much of the mapping and normalisation was carried out centrally in the Europeana Office. However this is now being shared with data providers and aggregators.

The Europeana Data Model (EDM) is replacing ESE. EDM aims to be an integration medium for collecting, connecting and enriching the descriptions provided by Europeana content provider. The purpose of the open structure of EDM is to enable the linking of data, placing it in the vanguard of semantic web developments. ESE was developed to constitute the lowest common denominator of the different data standards used for the heritage sectors. In contrast, EDM reverses this reductive approach and attempts to transcend the respective information perspectives of the sectors that are represented in Europeana. The strength of EDM lies in the fact that its development was not based on a specific standard but rather adopted an open, cross-domain Semantic Web based framework. It can accommodate several rich standards like LIDO for museums, EAD for archives or METS/MODS/MARC for digital libraries.

The use of EDM in production is currently (May 2012) being prototyped by Europeana in close collaboration with NTUA, WP5 leader. Moreover, Linked Heritage constitutes a significant use case, aiming to deliver resources fully modeled in EDM through the use of LIDO, which includes the CIDOC CRM concepts which are adopted in EDM (i.e. event-oriented representation).

1 http://pro.europeana.eu/technical-requirements

ROLE OF THIS DELIVERABLE IN THE PROJECT

The Linked Heritage project follows a content delivery plan for the ingestion (executed in WP6) providing an appropriate web-based tool for the aggregation and remediation of cultural metadata belonging to providers (T5.1, D5.1 Linked Heritage Technology Platform[2]). The process guarantees semantic interoperability across numerous data repositories of varied thematic categories, technical features and capabilities. It allows the seamless ingestion of diverse content and knowledge and, their subsequent harvesting by the Europeana Office through the OAI-PMH protocol.

The content providers participating in the Linked Heritage aggregation are mainly museums, but there are also libraries and archives from both the public and private sector. As it is reported in WP2[3], and taken from the various user requirements and analysis studies and meetings that took place during the first year of the project, there are many metadata standards used by the providers. There is also a vast variety, in the level of detail, in annotation. This ranges from the use of a small flat-structured set of elements to defining and using complex schemata that support various levels of annotation, item and concept relations and connections with controlled vocabularies and thesauri.

T5.3 implements the Linked Heritage Metadata Gateway fulfilling project milestone MS7 'Ingester ready for delivery of content to Europeana' and allowing the validation of the platform by WP6 and the subsequent execution of the Content Deliver plan (T6.1). WP5 also supports the design and implementation of demonstrators that enable linked cultural heritage data as those are identified and specified in WP2 (T2.3 and 2.4). MINT offers services for handling RDF data, linking and publishing Linked Data and will introduce a prototype semantic repository.

APPROACH

Following the paradigm of the ATHENA project, the work of WP2 and WP5 identified the need for an intermediate standard that would serve as the point of interoperability between the providers, as well as between the project's repository and outside sources. The adoption of LIDO (Lightweight Information Describing Objects), an XML Schema for contributing content to cultural heritage repositories implemented through a joint effort of the CDWA-Lite, museumdat and SPECTRUM communities, meets the requirements for publishing metadata of all of the cultural object classes. It integrates the relevant concepts of its source schemas into a single schema addressing several important issues that include an event-oriented approach, refined subject element, full support of multilingualism, revised handling of display and index elements and, revised identifier handling (for semantic web needs). In the light of recent and future Europeana developments, LIDO also allows the contribution of metadata along with digital representations of items, delivers information in a 'self-contained' way, includes links to original repository, distinguishes between display and indexing elements and provides references to controlled vocabularies and authorities.

The adoption of a reference metadata model, coupled with the loosely defined providers' input schemas, leads to an aggregation and mapping workflow which allows for an elaborate, visually guided ingestion of metadata in the repository. Its fundamental principles include the disassociation of input metadata from existing metadata standards. This avoids ambiguity over interpretation and the ability to create and manage transformations that will apply to the actual metadata records, which subsequently (re-)define the input schema in a semantic, machine understandable way, based on its mapping to LIDO.

The Linked Heritage Technology Platform provides a user friendly ingestion environment that allows for the extraction and presentation of all relevant and statistical information concerning input metadata together with an intuitive mapping service that illustrates LIDO. It also provides all the functionality and

2 http://mint-projects.image.ntua.gr/linkedheritage

3 Deliverable D2.1 - *Best practice report on cultural heritage linked data and metadata standards*

documentation required for the providers to define their crosswalks. Transformations are editable and reusable and can be applied incrementally to user input while providing, throughout all steps. Best practice examples, previews and visual indications to illustrate and guide user actions. One of the key capabilities lies in the ability to semantically enhance user metadata through conditional mapping of input elements using transformation functions (e.g. concatenate) that allows for the addition and enrichment of semantics even when those are not specifically stated in the input.

The tool also determines the operational work flow processes which bring the amalgamated content of the partner institutions into Europeana and to create, manage and execute, with the European Digital Library Office, the implementation plan (T6.1) to ensure that the content is visible in Europeana. The platform supports the export of the aggregated metadata to several established standards concerning presentation and archive management. The primary effort is directed to the transformation of the aggregated content to the Europeana Semantic Elements schema and the Europeana Data Model and, to the maintenance of an OAI-PMH repository to facilitate harvesting by Europeana.

The ingestion tool is based on the MINT metadata aggregation platform that was developed and is maintained by the leader of the WP, IVM Laboratory of the National Technical University of Athens. Its design is based on a specialisation of a general metadata ingestion work-flow, as it is described in detail in D5.1.

STRUCTURE OF THE DOCUMENT

Chapter 2 introduces the practices and technologies used for handling and remediation of metadata serialized in XML (and for RDF/XML). Section 2.1 presents the overall XML Metadata Gateway architecture. Section 2.2 the XML processing infrastructure (MINT PI), a scalable mechanism for structured data processing. Section 2.3 outlines the MINT OAI-PMH Server.

Chapter 3 addresses the handling of cultural heritage resources modelled in RDF in the light of efforts to enable linked cultural heritage data. It discusses approaches for the Linked Data pilots (Section 3.1) that will be carried out in the project as well as for the Semantic Repository (Section 3.2) that will be prototyped in the Linked Heritage Technology Platform. Finally, Chapter 3 summarizes the conclusions of the report.

XML METADATA HANDLING AND REMEDIATION

MINT XML METADATA GATEWAY

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), is a low barrier mechanism for repository interoperability. Data providers setup repositories that expose structured metadata, to which service providers make OAI-PMH service requests to harvest that metadata. The protocol consists of a set of six verbs or services that are invoked within HTTP. In the context of an aggregation, OAI-PMH provides a mechanism for technical interoperability between the ingestion platform and other modules or platforms (e.g. Europeana United Ingestion Mechanism).

The Linked Heritage Metadata Gateway is capable of managing heterogeneous collections of metadata records while exposing services for mapping and transforming from one metadata schema to another. In order to extend the functionalities of the OAI-PMH protocol and thus to expose metadata through an interoperable mechanism, MINT implements the defined OAI-PMH verbs on top of the underlying, domain-specific data layer. An issue that arises in the case of aggregations is that while being able to manage collections of metadata records, the OAI-PMH verbs operate on an item level, something that makes the implementation of the appropriate verbs, directly on top of a collection-based data layer a challenging task. For this reason, and also to design and introduce a set of robust, enhanced metadata processing services, an export mechanism is introduced in the MINT platform, facilitating scalable and reliable data delivery and exchange between different data layers and repositories.

The overall architecture for the delivery of content to Europeana, depicted in Figure 1, consists of a data processing layer responsible for iterating the records that are stored inside the Linked Heritage Platform repository, transform and manipulate them, together with a data layer which is exposed to Europeana by using an implementation of the OAI-PMH protocol. The overall architecture is depicted in *Figure* 1.

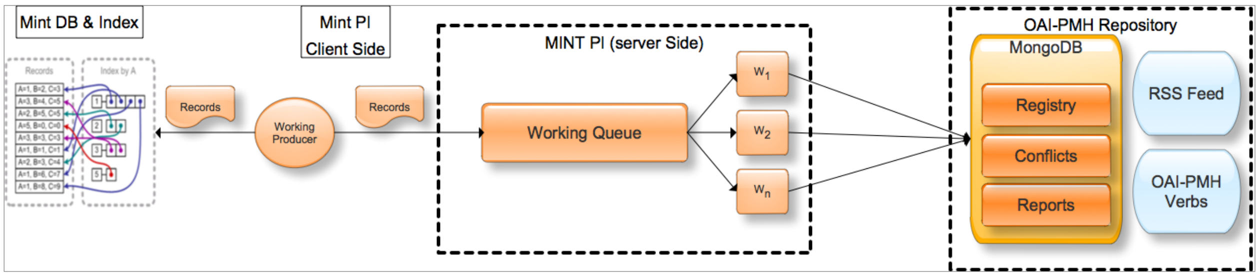


Figure 1*: Data Delivery* Architecture

MINT PROCESSING INFRASTRUCTURE (MINT PI)

MINT PI is part of the metadata interoperability services suite and offering a scalable mechanism for structured data processing. It is built using RabbitMQ[4] acting as a message broker in its core and utilizes a number of message queue patterns based on the AMQP[5], a standard wire-level protocol and semantic framework for high performance enterprise messaging. AMQP is an Open Standard for Messaging Middleware.

4 http://www.rabbitmq.com/

5 http://amqp.org/

By complying with the AMQP standard, middleware products written for different platforms and in different languages can send messages to one another. AMQP addresses the problem of transporting value-bearing messages across and between organisations in a timely manner. The approach of using message queues instead of a mechanism like Hadoop, was decided based on the requirement to scale on both large and small datasets without reducing the efficiency of the overall system. The overall architecture of MINT PI is depicted in Figure 2.

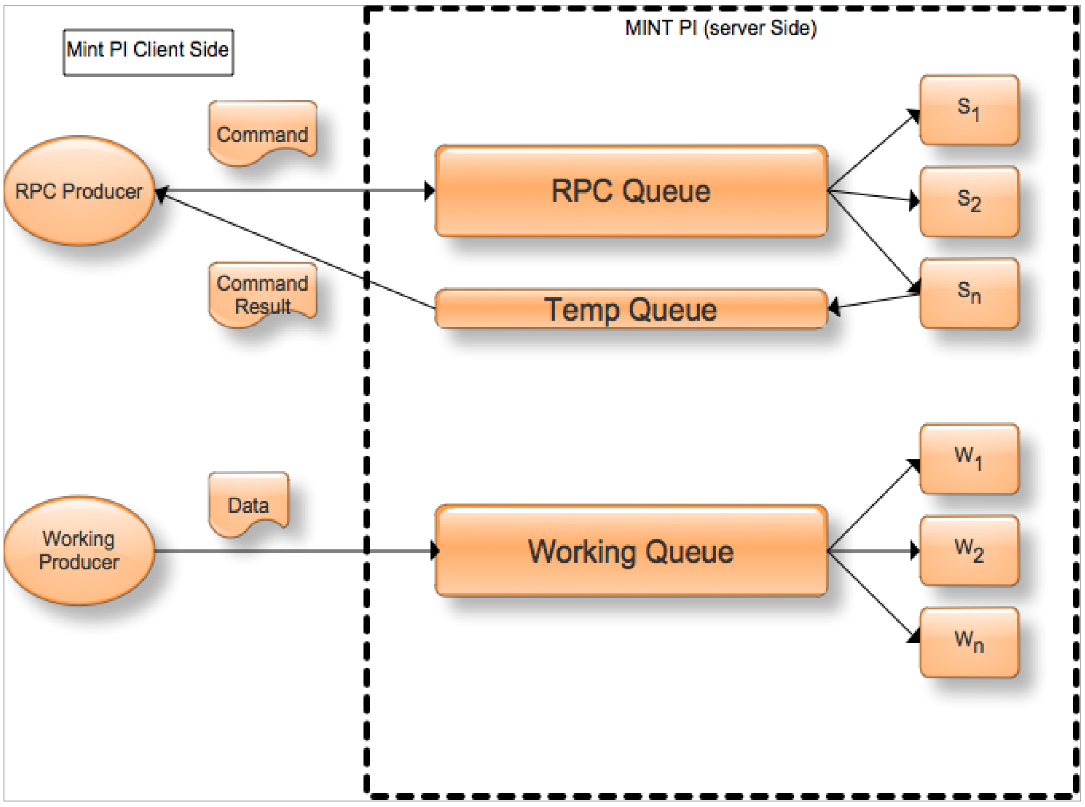


Figure 2: The Mint PI Architecture

MINT PI is using two distinct queue patterns: A RPC Queue pattern which is used for cases where the client desires to block while the processing is executed and also awaits for a response in a pre-defined format and a Working Queue pattern which is used for non blocking processing where the client submits the data for processing and does not wait for a response. The first queue pattern is mainly used for the implementation of specific commands, e.g. for implementing the cleaning or deletion of a repository, while the second is used for bulk processing of raw data, e.g. data transformation and enrichment of records.

Scalability is achieved by the parallel processing of many workers for the case of the working Queue and the existence of number of RPC Consumers that are running concurrently for the case of the RPC Queue. In both cases the workers and the RPC consumers might be running on different nodes of the cluster that is materialized. More workers and consumers can be added to the system at any time and thus increasing the processing power of the system. At the same time the RabbitMQ broker is also scalable. New nodes can be added to the broke, thus increasing the message per second ratio that can be handled by the system. In this way an overall scalable architecture for message delivery is defined which can be extended with minimal administrative effort.

MINT PI is not limited in a certain type of processing or data schema that is delivered using the messages. This is achieved by utilizing a software design pattern called the Strategy or Policy Pattern. Using this pattern it is possible to select different algorithms for execution at runtime. A UML diagram that represents the Strategy or Policy pattern is shown in Figure 3. An additional benefit of this design pattern is the abstraction introduced between the messaging layer of the system and the implementation of various algorithms for processing. A developer does not have to know about the intrinsic details of the messaging system in order to implement another algorithm for data processing by MINT PI.

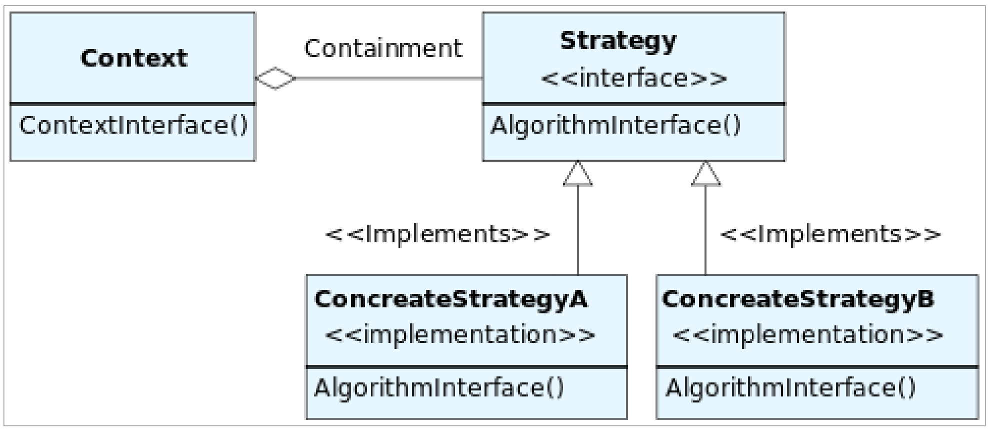


Figure 3: The Strategy or Policy Design Pattern

By following the architectural principles of MINT PI, a number of strategies can be implemented in order to support hosting and remediation of Linked Heritage cultural resources. More specifically the following strategies can be implemented to support the functionalities required to execute the content ingestion plan and to facilitate the semantic web prototypes of the project:

1. A strategy for applying the LIDO to ESE v3.4 Schema XSLT to the metadata records.
2. A strategy for enriching the records with multilingual subjects using the Linked Heritage terminologies.
3. A strategy for checking submitted URLs included inside the metadata records for their validity.
4. Finally, a strategy that stores the records in the appropriate format inside the OAI-PMH repository that is used as an interoperability API between Linked Heritage and Europeana.

Initially the producer is iterating metadata records directly from the Lucene index which is maintained in MINT. In this way it is possible to filter the records based on the data provider, their imports and status. These records are encapsulated inside a message in JSON serialization including the appropriate XSLT to be applied and various instructions for the complete set of strategies that will be applied on each one. Each worker thread fetches one message from the queue and applies in a predefined sequence the implemented strategies. The first step is to apply the XSLT and generate the ESEv3.4 record, the next step is to enrich the records with introduced thesauri terms for selected LIDO elements. When the transformation and enrichment is done, the final steps are to check the resulting records regarding the accessibility of the digital resources, e.g. thumbnails, and finally to store the records inside the OAI-PMH repository and thus making them accessible for harvesting by Europeana.

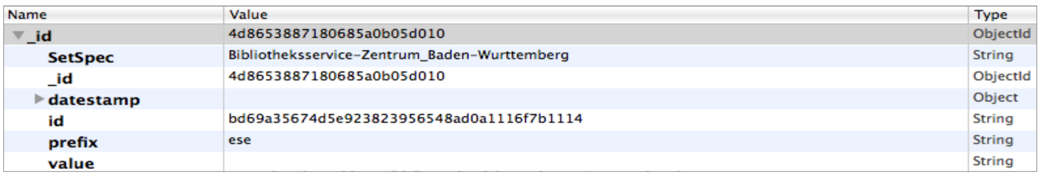
MINT OAI-PMH SERVER

A core characteristic of the MINT ingestion platform is that of it being agnostic with respect to the schema of an imported dataset. This characteristic is also inherited by the OAI-PMH service. For this to be achieved the data layer of the platform has to be able to handle heterogeneous metadata schemas. Based on this assumption, it was decided to implement the underlying data layer using a NoSQL solution that does not enforce any particular schema, and so is able to adapt to varying metadata models. The NoSQL solution that is used for the MINT OAI-PMH Platform is the MongoDB[6] document database. MongoDB is designed around the concept of documents that are internally implemented as JSON document and internally stored using the BSON format, which is a binary serialization of the JSON model. It allows the existence of JSON documents in the same database or even a collection having different fields, and so it does not enforce any specific data model schema. Finally it provides a rich set of native implementations of drivers for communicating with the database while the JSON format provides added value in the development of web applications because the stored data do not have to be transformed to a different format in order to be consumed by the applications.

The Linked Heritage OAI-PMH Platform Data Layer is designed around three distinct collections that exist inside a MongoDB data base. Collections can be perceived as tables of typical SQL databases, although it is not required that each document conforms to a specific data model and set of fields. These collections are the following:

1. Registry: in this collection the actual metadata records are stored and accessed by the implemented OAI-PMH verbs.
2. Conflicts: every time a new dataset is imported in the OAI-PMH repository, it is checked for the existence of duplicate records. If any exist, they are reported and stored in a different collection while they are also associated with a specific Report document.
3. Reports: every time an operation occurs on the OAI-PMH Repository, a report is created which includes any useful information regarding the operation. For example, if any conflict is identified between the items it is logged and reported for further reference.

The Registry collection constitutes the core collection of documents for the OAI-PMH repository. It contains all the records that it is desired to be exposed via the OAI-PMH repository. Each record is stored inside a JSON document which contains other information that might be useful to the platform. More specifically, the record document contains information regarding the organization to which the item belongs to, a unique hash key that is generated by calculating the SHA1 hash of the string representation of the item, a datestamp which represents the date and time the record was inserted and finally a namespace “prefix” value which is required by the OAI-PMH specification. An example of a Registry Document instance is depicted in Figure 4.



Name	Value	Type
▼ _id	4d8653887180685a0b05d010	ObjectId
SetSpec	Bibliotheksservice-Zentrum_Baden-Wurttemberg	String
_id	4d8653887180685a0b05d010	ObjectId
▶ datestamp		Object
id	bd69a35674d5e923823956548ad0a1116f7b1114	String
prefix	ese	String
value		String

Figure 4: Structure of the Registry Document.

As was mentioned earlier, another important collection of documents that are stored as part of the OAI-PMH repository is the Reports collection. The documents that are stored in this collection represent a set of valuable information that corresponds to specific actions of the repository platform. These actions are stored as values in the type attribute of the document and take one of the following values:

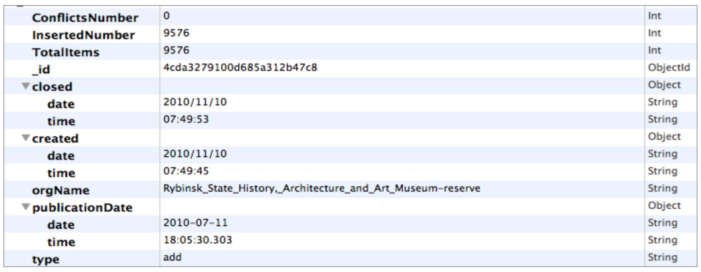
1. Add: this type represents an action in which records are added in the registry.

6 http://www.mongodb.org/

2. Update: this type represents an update action in which a set for a specific import already exist and it is updated by adding new metadata records.

3. Delete: deletion of records from a specific import that already exists in the registry.

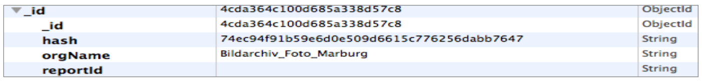
Apart from the action type, a number of other values are also stored as part of the Report Document. More specifically: A set of valuable statistics are stored; the number of conflicted items that were identified; the total number of the inserted records; and the total number of items which corresponds to sum of the inserted records plus the conflicts. Two datestamps are also stored as part of the Report document, one corresponding to the time of creation of the document and the other to the time of closing of the document. In this way it is possible for the repository to calculate the time it took to import a whole data set into the database. Finally the date the import was published in the ingestion platform is stored together with the name of the organization it belongs to. A visual representation of the Report document structure is depicted in Figure 5.



ConflictsNumber	0	Int
InsertedNumber	9576	Int
TotalItems	9576	Int
_id	4cda3279100d685a312b47c8	ObjectId
▼closed		Object
date	2010/11/10	String
time	07:49:53	String
▼created		Object
date	2010/11/10	String
time	07:49:45	String
orgName	Rybinsk_State_History,_Architecture_and_Art_Museum-reserve	String
▼publicationDate		Object
date	2010-07-11	String
time	18:05:30.303	String
type	add	String

Figure 5: Structure of the Report document.

The last collection of the OAI-PMH repository database contains conflict logs. The documents stored contain any metadata records that at the time of the exporting of an import from the ingestion platform to the OAI-PMH repository were found to be conflicted. These documents are quite simple in their structure. They contain an SHA1 hash of the conflicted item that was found together with the record and a reference to the Report document that it belongs to. In this way it is possible for someone to browse the actions that were made on the repository (e.g. additions, deletions and updates) and directly view the items that were found as conflicted for the cases of additions and updates. An example of a conflict document is depicted in Figure 6. It has to be noted that the whole procedure of creating unique hash codes and identifying conflicted items is an important functionality of the OAI-PMH repository platform because it provides a mechanism for creating unique ids for the metadata records and also a mechanism for identifying duplicate.



▼_id	4cda364c100d685a338d57c8	ObjectId
_id	4cda364c100d685a338d57c8	ObjectId
hash	74ec94f91b59e6d0e509d6615c776256dabb7647	String
orgName	Bildarchiv_Foto_Marburg	String
reportId		String

Figure 6: Structure of the Conflict document

On top of the data layer, that was described above, a set of functionalities is built, and more specifically an RSS Feed based on Atom, and, of course, the implementation of the actual OAI-PMH verbs. The implementation of the verbs is based on the customization of the OAICat[7] web application, which provides an abstract implementation of the OAI-PMH v2.0 specified verbs that can be customized in order to

7 www.oclc.org/research/activities/oaicat/default.htm

operate on top of different data layer technologies (e.g. flat XML files and relational databases). The verbs that were implemented in order to work with the current OAI-PMH Repository implementation are the following:

- **Identify**: This verb provides basic information regarding the running instance of the OAI-PMH v2.0 data repository such as, contact details of the admin of the repository, the base URL that can be used by a harvester and, a sample of an identifier among others. For a complete list of the information provided by the Identify verb someone can refer to the OAI-PMH specification. The information served by this verb does not have to be stored in the underlying data layer but is part of the configuration files of the verbs implementation.
- **GetRecord**: Given the identifier of a record and the desired namespace prefix, this verb fetches from the MongoDB database the corresponding metadata record and delivers it as a response to the harvesting client. In the current implementation a query is executed based on the prefix which is part of the Registry Document and the unique ID which is generated by the SHA1 hashing of the initial Metadata Record, this query corresponds to an exact match on the database.
- **ListIdentifiers**: Given a set name and a namespace prefix, this verb responds with a list of identifiers for items that correspond to these criteria. The set name is identical to the name of the organization. In this way it is possible to organize the records of the repository around organizations/providers. Again, the namespace prefix is matched with the prefix field of the Registry Document.
- **ListRecords**: This verb operates in a similar way to the ListIdentifiers with the main difference being that instead of returning only the identifiers, the complete Metadata Record is served.
- **ListMetadataFormats**: This verb is implemented by aggregating all the unique prefix values that are stored in the data layer by executing an aggregation for uniqueness query on the Registry collection. The resulting response that is served by this verb contains all the unique namespace prefixes that exist in the OAI-PMH repository and can be used for accessing the Metadata Records.
- **ListSets**: This verb returns a list of all the sets that exist in the OAI-PMH repository. Sets are named after the organizations that provide metadata records to the OAI-PMH repository, in this way it is possible for someone to retrieve only the records that are associated with a specific organization. The way the values are extracted is similar to the ListMetadataFormats verb.

In every case the verbs are implemented in such a way that they support paging through the mechanism of resumption tokens as it is defined by the OAI-PMH specification. The number of the returned items is specified through the configuration files of the OAICat running instance. Finally, by being a servlet implementation, the OAICat specific instance can be served through any of the available servlet containers that exist, e.g. Tomcat, JBoss, and Jetty. Currently it is served via a running Apache Tomcat instance.

The RSS feed is implemented following the Atom Syndication Format which is an XML language used for web feeds while the Atom Publishing Protocol (APP) is a simple HTTP based protocol for creating and updating web resources. The purpose of an RSS feed in the current OAI-PMH repository implementation is to provide a mechanism for notifying metadata consumers for the occurrence of specific actions, for example when new items are added or updated to the repository in an automatic way. This is achieved by creating the RSS Feed on top of the Reports collections that was described earlier.

In this way every time a new report is generated the feed is automatically updated and the subscribers are informed for the associated action. The implementation of the RSS Feed service is based on the Apache Abdera[8] project a functionally-complete, high performance implementation of the IETF Atom

8 http://abdera.apache.org/

Syndication Format (RFC 4287) and Atom Publishing Protocol (RFC 5023) specifications. The current implementation of the RSS Feed communicates directly with the MongoDB based data layer of the OAI-PMH repository. Every time a new Report document is inserted, it is also transformed into the Atom protocol XML representation and published on the Atom Feed that is maintained through the API of Apache Abdera. The RSS Feed can be served via any of the available servlet containers, for performance reasons it was decided to use a Jetty servlet container in the current implementation.

METADATA IN RDF

ENABLING LINKED CULTURAL HERITAGE DATA

The representation of cultural heritage resources and metadata in RDF is essential for establishing machine understandable semantic content. Metadata structures and values guide the conversion to resources that can then also be linked to external data sources like DBpedia. The transformation is not trivial due to the different purpose that each format is intended to serve. XML is used for collecting metadata describing cultural content, while RDF is employed to make statements about web resources in the form of subject-predicate-object expressions, the so called triples. Therefore, during the RDFization the things described in the XML document have to be firstly identified, together with the statements about them, before proceeding with the instantiation of the RDF model.

Although RDF provides a generic, abstract data model for describing resources using subject-predicate-object triples, it does not provide any domain-specific terms for describing classes of things in the world or how they relate to each other. This function is served by taxonomies, vocabularies and ontologies expressed in SKOS (Simple Knowledge Organization System), RDFS (the RDF Vocabulary Description Language, also known as RDF Schema) and OWL (the Web Ontology Language). Hence, a decision that has to be made in accordance with the resources described in the project is the selection of the vocabularies used for the RDF representation. Task 2.3 and 2.4 are going to identify interesting vocabularies for the implementation of Linked Data pilots.

Once the selection of the appropriate vocabularies for the RDF representation is complete, resources can be instantiated for the things described in the xml document. Thus, the described objects are matched to a URI. In detail, for every XML document an RDF file is constructed and named after the unique identifier ID, found in the XML accompanied by RDF file extension. The same ID can then be used in order to construct the resource, identifying each resource served under the domain. For example, an item assigned with the unique identifier 'UK-DC-7036a5f' would be mapped to a resource with URI http://mint.image.ece.ntua.gr/resource/UK-DC-7036a5f that will dereference to the UK-DC-7036a5f.rdf file. The next step is mapping appropriate XML elements to RDF properties using the selected vocabularies.

Another aspect that is very important when converting metadata to resources is the provenance of information. Therefore, for every RDF representation of an item, provenance metadata should be included to denote the publication date and the creator, allowing in that way consumers to track the origin of particular data fragments.

SEMANTIC REPOSITORY

In order to store the data produced by the RDFization process we use databases that were purpose-built for the storage and retrieval of RDF metadata. These databases are called triplestores and can be set up on the provided infrastructure. In the following we present some indicative, popular triplestores that have been tested with MINT.

Bigdata[9] is a clustered RDF store for ordered data (B+Trees) and designed to run as a server on commodity hardware. It is available under the GNU General Public License (GPL) and is designed for UNIX systems (Linux) with client connectors available for Java (Sesame). Additional scale can be achieved by simply plugging in more data services dynamically at runtime, which will self-register with the centralized service manager and start managing data automatically. Scale-out is achieved via dynamic key-range partitioning of the B+Tree indices.

9 http://www.systap.com/bigdata.htm

OWLIM[10] is a family of commercial RDF storage solutions, provided by Ontotext. It is available in two different editions: SwiftOWLIM is designed for medium data volumes (up to 100 million triples) since reasoning and query evaluation are performed in main memory, while BigOWLIM is designed for large data volumes and uses le-based indices that allow it to scale up to billions of RDF triples supporting dynamic configuration of cluster. Additionally, OWLIM is available as a SAIL (Storage and Inference Layer) for the Sesame RDF framework. SwiftOWLIM source code is provided free of charge for any purpose under a GNU LGPL license. BigOWLIM is free to use for research, evaluation, and testing purposes; for commercial applications an appropriate license is required. Both versions do not provide a dedicated extensibility mechanism but allow the definition of custom rules and rule languages for the inferencing process. BigOWLIM is in use for a large number of Semantic Web and Linked Data applications.

4Store[11] is an RDF database developed by Garlik Inc. It is implemented in ANSIC99 and available under the GNU General Public License (GPL), version 3. It is designed for UNIX-like systems (Linux, Mac OS) and runs as a server on a single-machine or in cluster-mode on 64bit machines. Client connectors are available for PHP, Ruby, Python, and Java. Dedicated extensibility mechanisms are not foreseen.

SHARD[12] which stands for Scalable, High-Performance, Robust and Distributed is an open source cloud-based triplestore technology that enables scalable data processing and analysis based on the Cloudera Distribution for Hadoop implementation of the MapReduce formalism. SHARD triple-store persists data as standard RDF triples and runs queries over this data using the standard SPARQL query-language.

Dydra[13] is a cloud-based RDF store, a database-as-a-service. It is a multi-tenant data store and query engine. The user does not have to estimate the data size, worry about clusters, nodes, resource use, or make big licensing commitments up front.

Sesame[14] is an open source Java framework for storing, querying and reasoning with RDF and RDFS. It can be used as a database for RDF and RDF Schema, or as a Java library for applications that need to work with RDF internally.

In order to select a triplestore for the storing of the data in Linked Heritage the following requirements are addressed:

- *Distributed*. This means that the storage devices are not connected to the same CPU, but they may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers.
- *Open Source Software*. An open-source license is a copyright license for computer software that makes the source code available for everyone to use. This allows end users to review and modify the source code for their own customization and/or troubleshooting needs. Open-source licenses are also commonly free, allowing for modification, redistribution, and commercial use without having to pay the original author.
- *SPARQL 1.1 support.* An RDF query language designed to meet the use cases and requirements identified by the RDF Data Access Working Group.
- *Accessibility.* Availability of APIs and ability to maintain through web interfaces.

10 http://www.ontotext.com/owlim

11 http://4store.org/

12 http://shard-3store.sourceforge.net/

13 http://dydra.com/

14 http://www.openrdf.org/

In the case of the Linked Data pilots of the project, Sesame and 4store will be the primary candidates for the semantic repository, allowing full scale experimentation with open source tools.

CONCLUSION

This deliverable reports on the technological solution for the establishment of the Linked Heritage Metadata Gateway (Task 5.3). This service complements the project's technology platform (Task 5.1, Deliverable D5.1) offering the technical components necessary for metadata remediation and in particular for the delivery of metadata to Europeana (ESE/EDM compliant). The content providers and aggregators in the Linked Heritage consortium, guided by the processes of WP6, will use the Linked Heritage Technology Platform and respective validation environment in order to process large amounts of metadata content. The Linked Heritage Metadata Gateway then processes and remediates XML metadata for the execution of the Europeana ingestion plan and RDF resources for prototyping of semantic web-enabled metadata services.

APPENDIX: DEFINITIONS OF TERMS AND ABBREVIATIONS

Reader is assumed to have basic understanding of XML, XSLT, RDF and the MINT metadata aggregation platform. Following is a glossary of technical terms and abbreviations used in the document.

AMQP (Advanced Message Queuing Protocol) is an open standard application layer protocol for message-oriented middleware. The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security.

APACHE LUCENE is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.

ATOM PUBLISHING PROTOCOL is a simple HTTP-based protocol for creating and updating web resources.

ATOM SYNDICATION FORMAT is an XML language used for web feeds.

HADOOP is an open source project from the Apache Software Foundation that provides a software framework for distributing and running applications on clusters of servers. It is inspired by Google's MapReduce programming model as well as its file system.

MESSAGE BROKER is either a complete messaging system or software that works with existing messaging transports in order to add routing intelligence and data conversion capabilities. A rules engine analyzes the messages and determines which application should receive them, and a formatting engine converts the data into the structure required by the receiving application.

MESSAGING MIDDLEWARE refers to software that provides an interface between applications, allowing them to send data back and forth to each other asynchronously.

MONGODB is a scalable, high-performance, open source NoSQL database

OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) is a protocol developed by the Open Archives Initiative. It is used to harvest (or collect) the metadata descriptions of the records in an archive so that services can be built using metadata from many archives. OAI-PMH uses XML over HTTP.

OAICAT open source software project is a Java Servlet web application providing a repository framework that conforms to the OAI-PMH v2.0. This framework can be customized to work with arbitrary data repositories by implementing some Java interfaces.

RPC (remote procedure call) is an inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space (commonly on another computer on a shared network) without the programmer explicitly coding the details for this remote interaction.

RSS (originally RDF Site Summary, often dubbed Really Simple Syndication) is a family of web feed formats used to publish frequently updated works—such as blog entries, news headlines, audio, and video—in a standardized format. An RSS document (which is called a "feed", "web feed", or "channel") includes full or summarized text, plus metadata such as publishing dates and authorship.

STRATEGY PATTERN (also known as the policy pattern) is a particular software design pattern, whereby algorithms can be selected at runtime. Formally speaking, the strategy pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

WIRE-LEVEL PROTOCOL is a low-level interface (wire level being the hardware level; actual wires, circuits, etc.). It typically refers to programming interfaces (APIs) in a network directly above the physical layer that are used strictly for transport or interconnection.