LoCloud

Local content in a Europeana cloud

D2.2: Modified MINT prototype

Authors:
Natasa Sofou (NTUA)
Nassos Drosopoulos (NTUA)
Vassilis Tzouvaras (NTUA)
Arne Stabenau (NTUA)

Version: Final

LoCloud is funded by the European Commission's
ICT Policy Support Programme

cip competitiveness and innovation framework programme 2007–2013

Revision History

Revision	Date	Author	Organisation	Description
1.0	01/02/2014	Natasa Sofou	NTUA	Initial draft
1.1	15/02/2014	Vassilis Tzouvaras	NTUA	Updates and corrections
1.2	25/02/2014	Natasa Sofou , Nasos Drosopoulos	NTUA	More details
1.3	28/02/2014	Natasa Sofou, Arne Stabenau, Nasos Drosopoulos		Update

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This deliverable contains original unpublished work except where clearly indicated otherwise.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.

Contents

1 Executive summary

This main objective of this deliverable is to describe the MINT ingestion platform presenting its architecture and the technologies used for its implementations. Additionally, the MINT workflow is explained along with the main modifications of MINT platform presented in new release. The role of MINT in LoCloud project is to assist content providers in the transformation of their in house metadata to Europeana Data Model and their publication to Europeana. MINT addresses the ingestion of metadata from multiple sources, the mapping of the imported records to the intermediate metadata schema and the transformation and storage of the metadata in a repository.

The main role of the MINT ingestion platform in the LoCloud project is to enable users to

- Provide metadata records in a range of “source” formats
- Convert metadata to LIDO, Carare2.0, and latest version of EDM (that is used as an intermediate standard)
- Monitor the progresses of content provision

while its key functionalities include:

- Organization and user level access rights and role assignment.
- Collection and record management (XML serialisation).
- Direct import and validation according to registered schemas (XSD).
- OAI-PMH based harvesting and publishing.
- Visual mapping editor for the XSLT language.
- Transformation and previewing (XML and HTML).
- Repository deployment and remediation interfaces.

MINT allows providers to perform mappings of their metadata through a very user-friendly interface (see figure below).

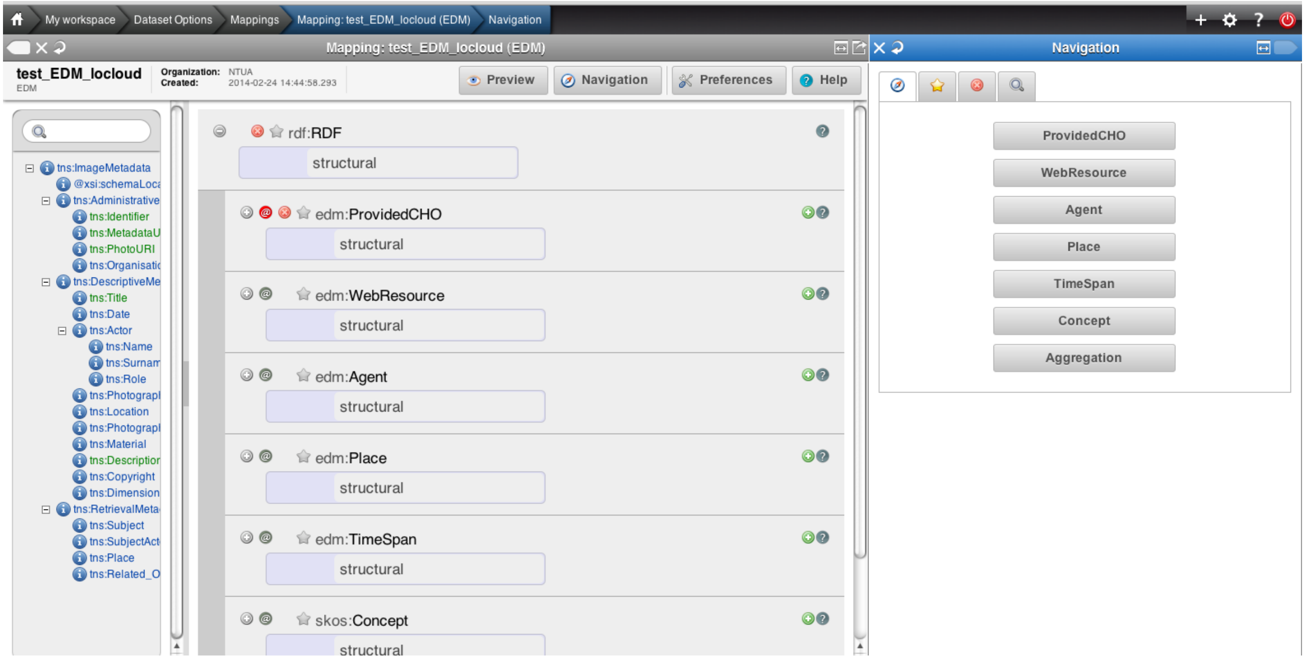


Figure 1-1 MINT mapping editor

The structure that corresponds to a user's specific import is visualized in the mapping interface as an interactive tree that appears on the left hand side of the editor. This tree is extracted from the user's import and represents the snapshot of the XML schema that is used as input for the mapping process. The user is able to navigate and access element statistics and also to search the tree by using the text field on the top.

On the right hand side, buttons correspond to high-level elements of the target schema and are used as links for accessing their corresponding sub-elements. These are visualized on the middle part of the screen as a tree structure of embedded boxes, representing the internal structure of the complex element. The user is able to interact with this structure by clicking to collapse and expand every embedded box that represents an element, along with all relevant information (attributes, annotations) defined in the XML schema document. To perform an actual (one to one) mapping between the input and the target schema, a user has to simply drag a source element from the left and drop it on the respective target in the middle.

After the creation of a valid mapping, content providers metadata are transformed to the LIDO, Carare2.0 or EDM (depending on the selected target schema) and then transmitted to NTUA's OAI-PMH server.

The MINT ingestion platform has been used in many European projects for delivering metadata to Europeana, however many specializations have been implemented to meet the special needs of each project content providers.

In detail, the MINT instance that was set up for the LoCloud project differs from MINT instances used in other projects in the following

- It has been customized so as to assist the content providers to easily map their metadata to LIDO, Carare2.0 and EDM with the aid of element's bookmarks.
- The MINT mapping editor has been totally re-constructed taking into account the experience gathered from the usage of MINT in other projects and also the feedback from content providers. The new mapping editor is based on a new JSON library (JsonSmart) that is much faster and responsive than one used in previous versions. It also permits the creation of more complex mappings by the use of structural mappings on any element, and if-else conditional mappings making in that way the implementation of crosswalks from any provider's input schema to selected target schema.
- It fully supports the Europeana Data.
- The repository exposes the records in RDF and OAI-DC fully supporting in that way the harvesting in RDF and the OAI-PMH interface.
- Progress reports through MINT per content provider for monitoring of the metadata production and publication to our (OAI-PMH) repository.

2 Introduction

MINT services compose a web based platform that was designed and developed to facilitate aggregation initiatives for cultural heritage content and metadata in Europe. It is employed from the first steps of such workflows, corresponding to the ingestion, mapping and aggregation of metadata records, and proceeds to implement a variety of remediation approaches for the resulting repository. The platform offers a user and organization management system that allows the deployment and operation of different aggregation schemes (thematic or cross-domain, international, national or regional) and corresponding access rights. Registered organizations can upload (http, ftp, oai-pmh) their metadata records in xml or csv serialization in order to manage, aggregate and publish their collections.

A reference metadata model serves as the aggregation schema to which the ingested (standard or proprietary) schemata are aligned to. Users can define their metadata crosswalks with the help of a visual mappings editor for the XSL language. Mapping is performed with simple drag-and-drop or input operations which are then translated to the corresponding code. The mappings editor visualizes both the input and target XSD, in an intuitive interface that provides access and navigation of the structure and data of the input schema, and the structure, documentation and restrictions of the target one. It supports string manipulation functions for input elements in order to perform 1-n and m-1 (with the option between concatenation and element repetition) mappings between the two models. Additionally, structural element mappings are allowed, as well as constant or controlled value (target schema enumerations) assignment, conditional mappings (with a complex condition editor) and value mappings between input and target value lists. Mappings can be applied to ingested records, edited, downloaded and shared as templates between users of the platform.

Preview interfaces present to users the steps of the aggregation including the current input xml record, the XSLT of their mappings, the transformed record in the target schema, subsequent transformations from the target schema to other models of interest (e.g. Europeana's metadata schema), and available html renderings of each xml record. Users can transform their selected collections using complete and validated mappings in order to publish them in available target schemas for the required aggregation and remediation steps.

The platform has been deployed for a variety of aggregation workflows corresponding to the whole or parts of the backend services. Specifically, it has served the aggregator of museum content for Europeana (and one of the largest in volume and significance), the ATHENA project, that has ingested and aligned to the LIDO format over 4 million items from 135 organizations. The resulting repository offers an OAI-PMH interface exposing the records in the Europeana Semantic Elements schema. The use of a reference model allowed the rapid support of updated ESE versions that were introduced in the duration of the project (2008-2011), with minimal input from providers. Users effort to align their data to an adopted domain model also motivated them to update their collection management systems and improve the quality of their annotations in order to take advantage of a well defined, machine understandable model and, subsequently, control and enrich their organization's contribution and visibility through the aggregator and Europeana.

This deliverable describes the MINT ingestion platform that is used in the LoCloud project for the large-scale ingestion of metadata with final aim the delivery to Europeana. As explained earlier in this section, the development of MINT started in the ATHENA[1] project when the NTUA team integrated all the necessary components for ingesting, mapping and publishing metadata to Europeana into a common technology platform, while it evolved through its use in other Europeana-feeder projects like Linked Heritage[2], EuScreen[3], ECLAP[4], Carare[5] Fashion[6], EuropeanaPhotography[7] and others. The MINT platform provides content holders with the ability to perform the required mapping of their own metadata schemas intto LIDO, Carare2.0 and EDM. It addresses the ingestion of metadata from multiple sources, the mapping of the imported records to a target metadata schema and the transformation and storage of the metadata in a repository. Although its deployment is also guided by expediency, the system has been developed using established tools and standards, embodying best practices in order to animate familiar content provider procedures in an intuitive and transparent way also for newcomers.

2.1 Background

Metadata records are critical to the documentation and maintenance of interrelationships between information resources and are being used to find, gather, and maintain resources over long periods of time. The consistent application of a descriptive metadata standard improves the user's search experience and makes information retrieval within a single collection or across multiple datasets more reliable. Descriptive, administrative, technical, and preservation metadata contribute to the management of information resources and help to ensure their intellectual integrity both now and in the future. In parallel with other domains, many researchers in the digital cultural heritage community recognized the need to lower the barriers for the management and aggregation of digital resources, by implementing some measure of interoperability among metadata standards and then with proprietary data structures. There is a wide range of proposed solutions, including crosswalks, translation algorithms, metadata registries, and specialized data dictionaries.

A crosswalk provides a mapping of metadata elements from one metadata schema to another. The prerequisite to a meaningful mapping requires a clear and precise definition of the elements in each schema. The primary difficulty is to identify the common elements in different metadata schemas and put this information to use in systems that resolve differences between incompatible records. Crosswalks are typically presented as tables of equivalent elements in two schemas and, even though the equivalences may be inexact, they represent an expert's judgment that the conceptual differences are immaterial to the successful operation of a software process that involves records encoded in the two models. A crosswalk supports the ability of a retrieval mechanism to query fields with the same or similar content in different data sources; in other words, it supports semantic interoperability.

Crosswalks are not only important for supporting the demand for single point of access or cross-domain searching; they are also instrumental for converting data from one format to another. However, aggregating metadata records from different repositories may create confusing display results, especially if some of the metadata was automatically generated or created by institutions or individuals that did not follow best practices or standard thesauri and controlled vocabularies.

[1] http://www.athenaeurope.org/

[2] http://www.linkedheritage.eu/

[3] http://euscreen.eu/

[4] http://www.eclap.eu/drupal/?q=en-US

[5] http://www.carare.eu

[6] http://www.europeanafashion.eu/

[7] http://www.europeana-photography.eu/

Mapping metadata elements from different schemas is only one level of cross walking. Another level of semantic interoperability addresses datatype registration and formatting of the values that populate the metadata elements, e.g. rules for recording personal names or encoding standards for dates, and the alignment between local authority files and adopted terminologies.

The MINT ingestion platform implements an aggregation infrastructure offering a crosswalk mechanism to support subsequent critical activities:

- harvesting and aggregating metadata records that were created using shared community standards or proprietary metadata schemas ;
- migrating from providers' models (whether standard or local) to a reference model

3 MINT Ingestion Workflow

The developed system facilitates the ingestion of semi-structured data and offers the ability to establish crosswalks to the reference schemas (LIDO, Carare2.0, EDM) in order to take advantage of a well-defined, machine understandable model. The underlying data serialization is in XML, while the user's mapping actions are registered as XSL transformations.

The main role of the MINT ingestion platform in the LoCloud project is to enable users to

- Provide metadata records in a range of "source" formats
- Convert metadata to selected target schema (LIDO, Carare2.0, EDM - used as an intermediate standard before publishing to Europeana)
- Monitor the progresses of content provision

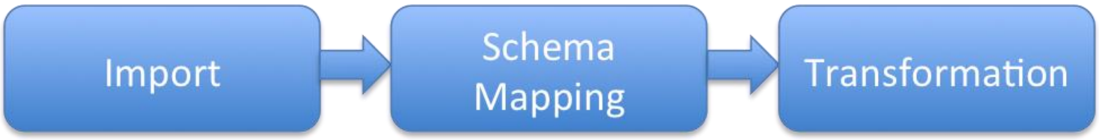


Figure 3-1 Ingestion workflow

The metadata ingestion workflow, as illustrated in Figure 3-1 Ingestion workflow, consists of three main steps. First is the Import of provider's metadata using common data delivery protocols, such as OAI-PMH, HTTP and FTP. Following is the Schema Mapping procedure, during which the imported metadata are mapped to one of the intermediate schemas used in LoCloud: LIDO, Carare2.0 or EDM. A graphical user interface assists content providers in mapping their metadata the preferred intermediate schema, using an underlying machine-understandable mapping language. Furthermore, it provides useful statistics about the provider's metadata also supporting the share and reuse of metadata crosswalks and the establishment of template transformations. The third step is the Transformation procedure, during which provider's metadata is transformed to the selected schema (LIDO, Carare2.0, and EDM) by using the mapping they made in the previous step.

4 Mapping editor

Metadata mapping is the crucial step of the ingestion procedure. It formalizes the notion of a metadata crosswalk, hiding the technical details and permitting semantic equivalences to emerge as the centrepiece. It involves a user-friendly graphical environment, as shown in Figure 4-1 (example of mapping opened in the editor), where interoperability is achieved by guiding users in the creation of mappings between input and target elements. User imports are not required to include the respective schema declaration, while the records can be uploaded as XML or CSV files. User's mapping actions are expressed through XSLT style sheets, i.e. a well-formed XML document conforming to the namespaces in XML recommendation. XSLT style sheets are stored and can be applied to any user data, exported and published as a well-defined, machine understandable crosswalk and, shared with other users to act as template for their mapping needs.

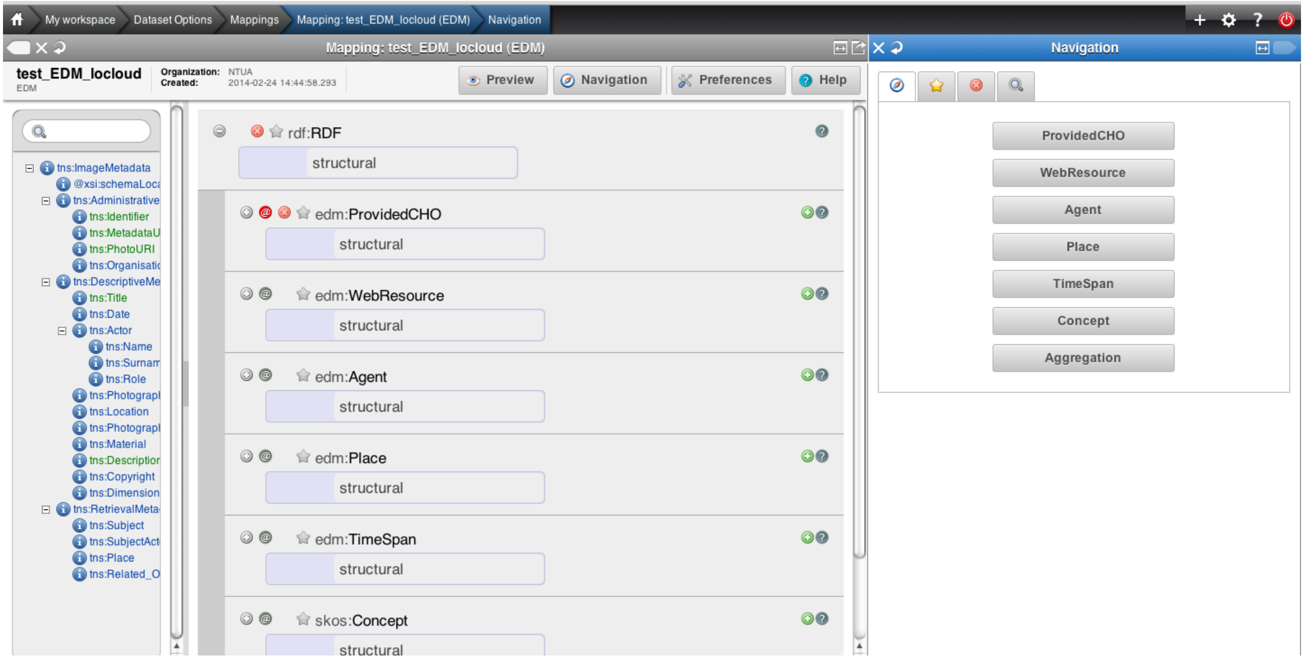


Figure 4-1 Screenshot of the mapping editor

The structure that corresponds to a user's specific import is visualized in the mapping interface as an interactive tree that appears on the left hand side of the editor. The tree represents the snapshot of the XML schema that is used as input for the mapping process. The user is able to navigate and access element statistics for the specific import while the set of elements that have to be mapped can be limited to those that are actually populated. The aim is to accelerate the actual work, especially for the non-expert user, and to help overcome expected inconsistencies between schema declaration and actual usage.

On the right hand side, buttons correspond to high-level elements of the target schema and are used to access their corresponding sub-elements. These are visualized on the middle part of the screen as a tree structure of embedded boxes, representing the internal structure of the complex element. The user is able to interact with this structure by clicking to collapse and expand every embedded box that represents an element, along with all relevant information (attributes, annotations) defined in the XML schema document. To perform an actual (one to one) mapping between the input and the target schema, a user has to simply drag a source element from the left and drop it on the respective target in the middle.

The user interface of the mapping editor is schema aware regarding the target data model and enables or restricts certain operations accordingly, based on constraints for elements in the target XSD. For example, when an element can be repeated then an appropriate button appears to indicate and implement its duplication. Several advanced mapping features of the language are accessible to the user through actions on the interface, including:

- String manipulation functions for input elements;
- m-1 mappings with the option between concatenation and element repetition;
- Structural element mappings;
- Constant or controlled value assignment;
- Conditional mappings (with a complex condition editor);
- Value mappings editor (for input and target element value lists).

Mappings can be applied to ingested records, edited, downloaded and shared as templates. Users can transform their selected collections using complete and validated mappings in order to publish them in available target schemas for the required aggregation and remediation steps.

5 MINT Modifications for LoCloud

The previous sections presented the overall workflow as well as the main functionality of the MINT mapping tool. Although similar description of functionality and usage of MINT ingestion platform is also available in other deliverables authored by the NTUA team for other European aggregation projects like LoCloud, it still remains vital for the content providers of this project. Other deliverables where MINT workflow and functionality is described include:

- Athena D7.1 “First version of the semantic interoperability plug-ins”,
- ECLAP D4.1 “Metadata descriptors Identification and Definition”,
- Linked Heritage D5.1 “Linked Heritage Technology Platform”,
- EuropeanaPhotography D5.2 “The MINT Mapping tool”,
- AthenaPlus D3.1 “MINT ingestion Platform” .

However even if the main functionality of the MINT mapping tool – that is the transformation of the metadata extracted from the provider’s metadata management systems in various standards to a common metadata standard for the project – remains the same, many specializations are implemented to meet the special needs of the content providers of each project. In this section we present the specializations that were implemented for the LoCLoud project.

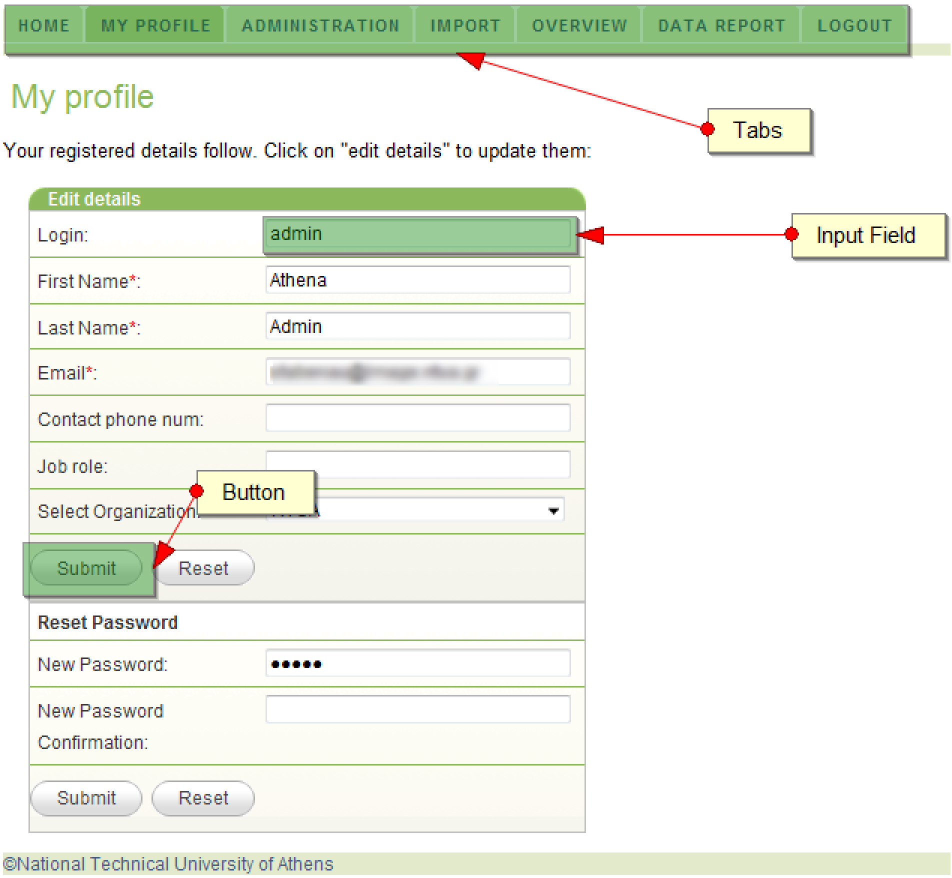
5.1 User Interface

A major development in the MINT release used for the LoCloud version is its new user interface. The main objective was to redesign the MINT’s mapping tool user interface in a way that would permit to the various content providers to easily use it and better understand the overall workflow towards transformation, with final aim the delivery to Europeana. The following subsections present in more detail the updated user interface.

5.1.1 Aggregation workflow guided interface

The user interface of the MINT release used in previous Europeana-feeder projects like Athena, Linked Heritage, ECLAP, EuScreen and others was implemented using the Javascript library YUI[8]. The interface approach followed there was tab-oriented as shown below:

[8] http://yuilibrary.com/



My profile

Your registered details follow. Click on "edit details" to update them:

Edit details

Login:	admin
First Name*:	Athena
Last Name*:	Admin
Email*:	[redacted]
Contact phone num:	
Job role:	
Select Organization:	

Reset Password

New Password:	•••••
New Password Confirmation:	

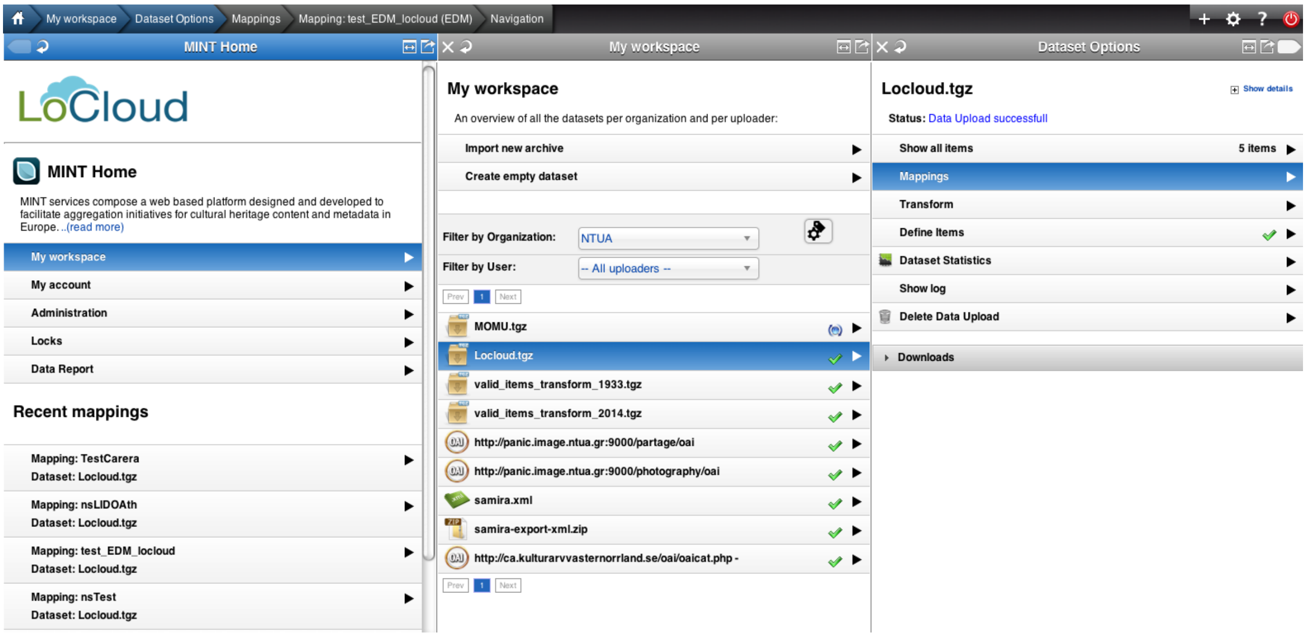
©National Technical University of Athens

Figure 5-1 Previous release MINT interface

The content provider could navigate to MINT's functionalities through the different tabs that appeared on the top of the working window. The main problem with that approach however was that individual actions were not related and the users did not have any guidance thought the aggregation process. Therefore the fact that a provider first had to import his metadata by using the import preview, then to click on the Overview preview to locate its import and after that to perform mappings and transformations was not clear, and even after training sessions many of the providers had problems in following the right steps. In the new release of MINT platform, for the improved user interface of the mapping tool, a different approach has been followed where the jQuery library Kaiten[9] have been used. The main difference in the new approach is that interactions between the user and the application are a stack of contiguous screens where each screen is presented in columns as illustrated in Figure 5-2.

This way the content provider can only perform specific actions depending on his starting point. For example when clicking on the workspace button from MINT's start page he can either make an import (that is highlighted) or select one of the existing ones. In a similar manner after selecting an import the following screen appears illustrating to the provider his possible options.

[9] http://www.officity.com/kaiten/



MINT services compose a web based platform designed and developed to facilitate aggregation initiatives for cultural heritage content and metadata in Europe…(read more)

An overview of all the datasets per organization and per uploader:

Filter by Organization: NTUA

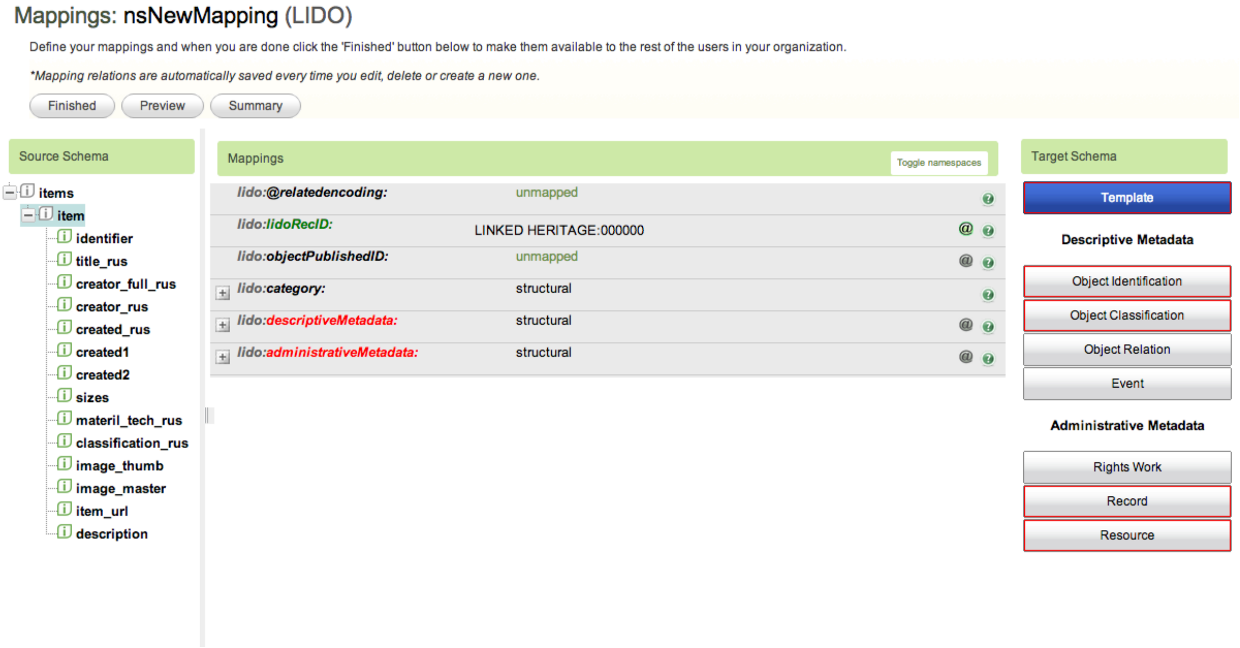
Filter by User: -- All uploaders --

Status: Data Upload successfull

Figure 5-2 MINT new user interface

5.1.2 Improved Schema browsing

One of the major difficulties that content providers experienced when producing mappings using the previous version of MINT was when browsing the input and target schema.


Mappings: nsNewMapping (LIDO)

Define your mappings and when you are done click the 'Finished' button below to make them available to the rest of the users in your organization.

*Mapping relations are automatically saved every time you edit, delete or create a new one.

lido:@relatedencoding:	unmapped
lido:lidoRecID:	LINKED HERITAGE:000000
lido:objectPublishedID:	unmapped
lido:category:	structural
lido:descriptiveMetadata:	structural
lido:administrativeMetadata:	structural

Figure 5-3 MINT's old mapping editor

The MINT release used in previous projects had limited functionality for browsing schemas as shown in Figure 5-3, only permitting providers to:

- explore the input schema by using its hierarchy
- browse the target schema by clicking on some predefined buttons that categorized elements

In order to make the discovery of schema's elements easier and facilitate the mapping process the new MINT interface allows providers to search elements in both input and target schema. In addition for further assisting providers to meet the project requirements and to get familiarized with the target schema bookmarks can been set up for the LoCloud recommended elements.

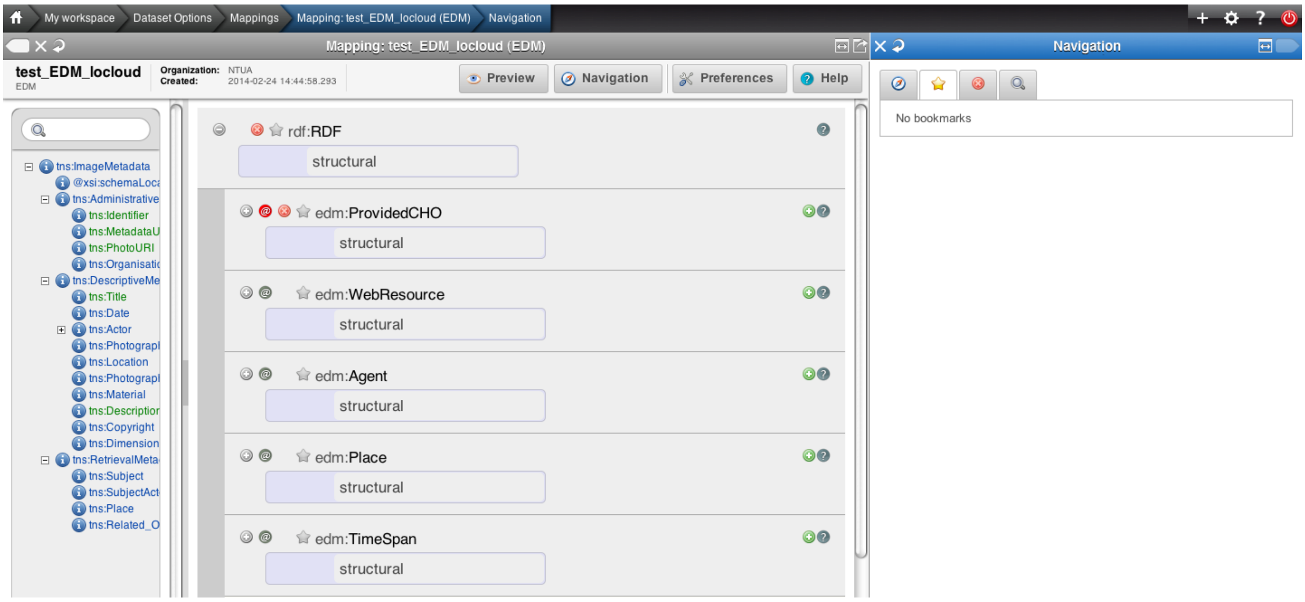


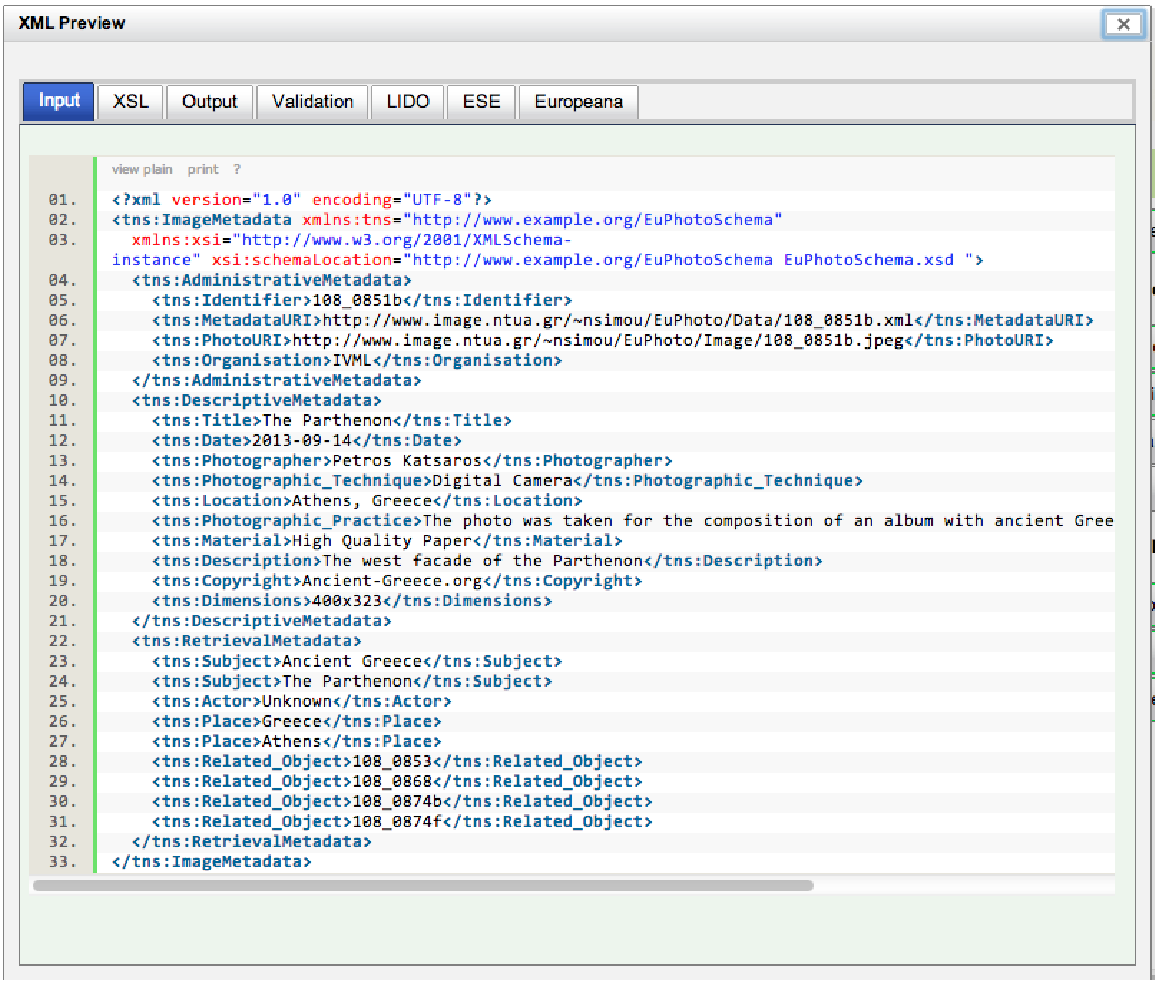
Figure 5-4: MINT's new mapping editor

5.1.3 Improved item preview interface

Another update of the MINT release used for LoCLoud is the new item preview interface. In the previous version of the tool the user could only see the preview of one item while working on a mapping, as in *Figure 5-5*. This was not always very convenient, since different items of an import may vary on the information (elements of input metadata) they had. For example the first item of an import could have element 'Author' while the rest of the imports may have missed that. Cases like were resulting in transformations with many errors simply because the user did only have feedback of his mapping for the first record.

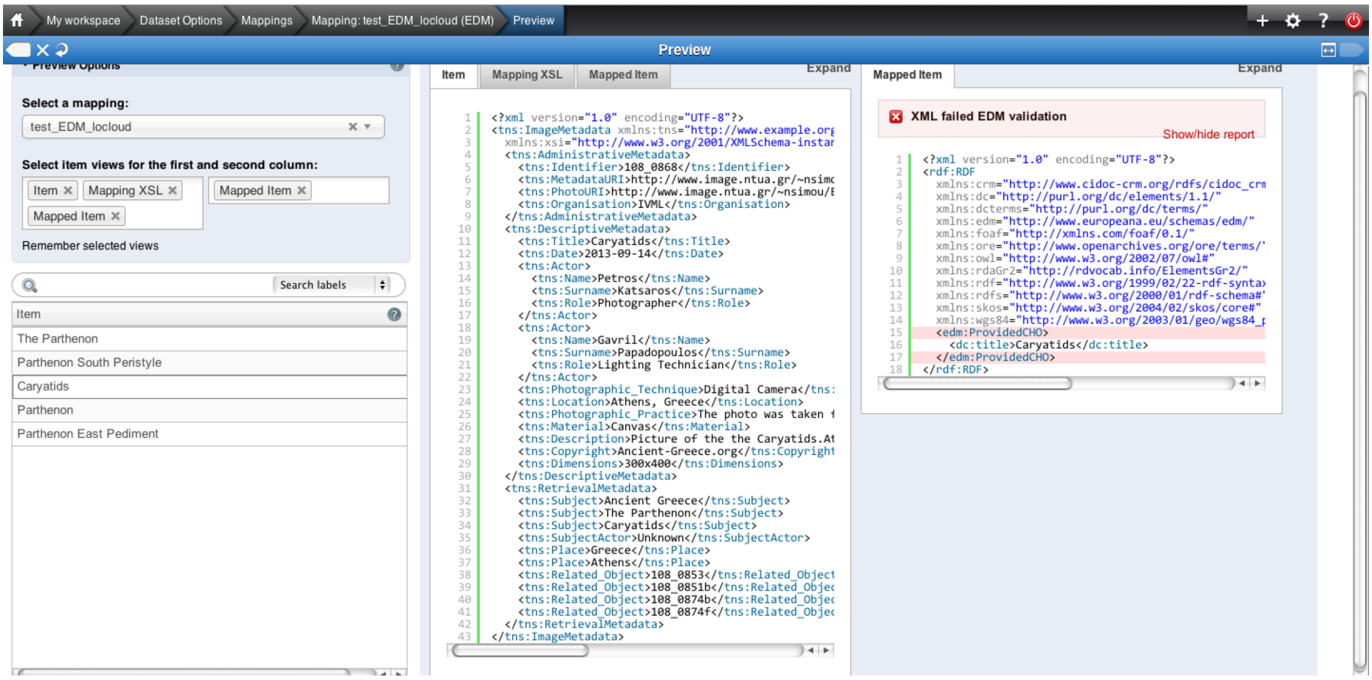
The new Item Preview interface as illustrated in *Figure 5-6* does not suffer from these limitations. In detail, the new preview interface enables the user to:

- preview all his items while working on a mapping;
- select the previews he prefers to see in the two preview panes;
- get inline validation for each preview according to its' related XML schema;
- instantly select a different mapping and preview its result for the same item;
- search the items he wishes to preview (this feature is still under development).



```xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:ImageMetadata xmlns:tns="http://www.example.org/EuPhotoSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.example.org/EuPhotoSchema EuPhotoSchema.xsd ">
  <tns:AdministrativeMetadata>
    <tns:Identifier>108_0851b</tns:Identifier>
    <tns:MetadataURI>http://www.image.ntua.gr/~nsimou/EuPhoto/Data/108_0851b.xml</tns:MetadataURI>
    <tns:PhotoURI>http://www.image.ntua.gr/~nsimou/EuPhoto/Image/108_0851b.jpeg</tns:PhotoURI>
    <tns:Organisation>IVML</tns:Organisation>
  </tns:AdministrativeMetadata>
  <tns:DescriptiveMetadata>
    <tns:Title>The Parthenon</tns:Title>
    <tns:Date>2013-09-14</tns:Date>
    <tns:Photographer>Petros Katsaros</tns:Photographer>
    <tns:Photographic_Technique>Digital Camera</tns:Photographic_Technique>
    <tns:Location>Athens, Greece</tns:Location>
    <tns:Photographic_Practice>The photo was taken for the composition of an album with ancient Gree
    <tns:Material>High Quality Paper</tns:Material>
    <tns:Description>The west facade of the Parthenon</tns:Description>
    <tns:Copyright>Ancient-Greece.org</tns:Copyright>
    <tns:Dimensions>400x323</tns:Dimensions>
  </tns:DescriptiveMetadata>
  <tns:RetrievalMetadata>
    <tns:Subject>Ancient Greece</tns:Subject>
    <tns:Subject>The Parthenon</tns:Subject>
    <tns:Actor>Unknown</tns:Actor>
    <tns:Place>Greece</tns:Place>
    <tns:Place>Athens</tns:Place>
    <tns:Related_Object>108_0853</tns:Related_Object>
    <tns:Related_Object>108_0868</tns:Related_Object>
    <tns:Related_Object>108_0874b</tns:Related_Object>
    <tns:Related_Object>108_0874f</tns:Related_Object>
  </tns:RetrievalMetadata>
</tns:ImageMetadata>
```

Figure 5-5 MINT’s old item preview interface



Select a mapping:

test_EDM_locloud

Select item views for the first and second column:

Item ✕ Mapping XSL ✕ Mapped Item ✕

Mapped Item ✕

Remember selected views

Item

The Parthenon

Parthenon South Peristyle

Caryatids

Parthenon

Parthenon East Pediment

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:ImageMetadata xmlns:tns="http://www.example.org
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instar
  <tns:AdministrativeMetadata>
    <tns:Identifier>108_0868</tns:Identifier>
    <tns:MetadataURI>http://www.image.ntua.gr/~nsimo
    <tns:PhotoURI>http://www.image.ntua.gr/~nsimou/E
    <tns:Organisation>IVML</tns:Organisation>
  </tns:AdministrativeMetadata>
  <tns:DescriptiveMetadata>
    <tns:Title>Caryatids</tns:Title>
    <tns:Date>2013-09-14</tns:Date>
    <tns:Actor>
      <tns:Name>Petros</tns:Name>
      <tns:Surname>Katsaros</tns:Surname>
      <tns:Role>Photographer</tns:Role>
    </tns:Actor>
    <tns:Actor>
      <tns:Name>Gavril</tns:Name>
      <tns:Surname>Papadopoulos</tns:Surname>
      <tns:Role>Lighting Technician</tns:Role>
    </tns:Actor>
    <tns:Photographic_Technique>Digital Camera</tns:
    <tns:Location>Athens, Greece</tns:Location>
    <tns:Photographic_Practice>The photo was taken f
    <tns:Material>Canvas</tns:Material>
    <tns:Description>Picture of the the Caryatids.At
    <tns:Copyright>Ancient-Greece.org</tns:Copyright
    <tns:Dimensions>300x400</tns:Dimensions>
  </tns:DescriptiveMetadata>
  <tns:RetrievalMetadata>
    <tns:Subject>Ancient Greece</tns:Subject>
    <tns:Subject>The Parthenon</tns:Subject>
    <tns:Subject>Caryatids</tns:Subject>
    <tns:SubjectActor>Unknown</tns:SubjectActor>
    <tns:Place>Greece</tns:Place>
    <tns:Place>Athens</tns:Place>
    <tns:Related_Object>108_0853</tns:Related_Object
    <tns:Related_Object>108_0851b</tns:Related_Objec
    <tns:Related_Object>108_0874b</tns:Related_Objec
    <tns:Related_Object>108_0874f</tns:Related_Objec
  </tns:RetrievalMetadata>
</tns:ImageMetadata>
```

☒ XML failed EDM validation

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:crm="http://www.cidoc-crm.org/rdfs/cidoc_crm
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:edm="http://www.europeana.eu/schemas/edm/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:ore="http://www.openarchives.org/ore/terms/'
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdaGr2="http://rdvocab.info/ElementsGr2/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-synta
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#'
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:wgs84="http://www.w3.org/2003/01/geo/wgs84_p
  <edm:ProvidedCHO>
    <dc:title>Caryatids</dc:title>
  </edm:ProvidedCHO>
</rdf:RDF>
```

Figure 5-6 MINT's new item preview interface

5.2 Mapping functionality

MINT's main functionality in LoCloud is mapping and transformation of the providers' in-house metadata to the LoCloud reference schema in order to facilitate its publication to Europeana. Content providers perform this process using MINT's mapping editor, which offers a wide range of mapping functionalities. Therefore, the development of new mapping functionalities always comprises part of NTUA's on-going work. More specifically the mapping editor has been totally reconstructed using a new JSON library (JsonSmart) that is much faster than the one used in previous versions. In some tests performed comparing the two libraries, it turned out that JsonSmart only required a couple of seconds for the parsing of a mapping while for the same mapping the previous library may have required a minute of parsing. The following sections describe in detail the latest mapping functionalities.

5.2.1 Negative closure for functional mappings

An important mapping functionality that has been implemented in the latest MINT release is the negation for all the conditions used in conditional mappings. By using this functionality a content provider can perform mappings in which his dataset is separated into two sets according to its values (e.g. those that start with, and those that don't start with) and each one of them is treated in a different way.

5.2.2 If else mapping

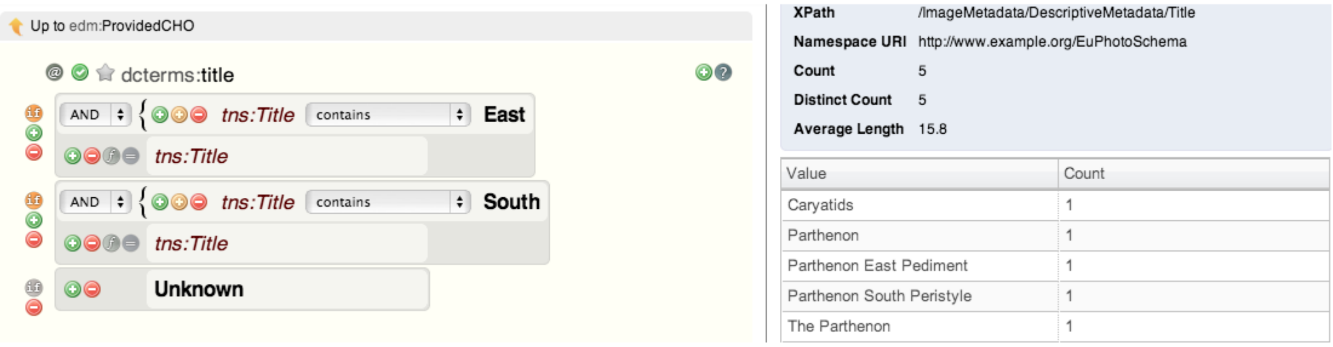
In addition to the negative closure for all the MINT functions a user can also make mappings using if else statements. This functionality can be better understood by the example shown below.

The table on the right displays all the values of element 'tns:title'. This XPath is first mapped to element 'dcterms:title'. After that a conditional mapping is made by clicking on the 'if'-button. Then by clicking on the '+'-button located below the 'if'-button additional conditions can be set. The last condition set is the closure of all the previous. In simple words the condition set in the example below is the following.

- if the value of element 'tns:Title' contains "East" then map the value of the element 'tns:Title' to 'dcterms:Title'
- else if the value of element 'tns:Title' contains "South" then map the value of the element 'tns:Title' to 'dcterms:Title'
- else map the value "Unknown" to 'dcterms:Title'

So for the items having 'tns:Title' values 'Caryatids', 'Parthenon' and 'The Parthenon' the value 'Unknown' will be mapped[10].

[10] The use of the "and" button in Figure 13 can be a bit confusing. The "AND" button refers to the condition set on the conditional mapping. The reader is reffered to section 10.3.5.6 for a better understanding of conditional mappings.



Up to edm:ProvidedCHO

dcterms:title

AND tns:Title contains East

tns:Title

AND tns:Title contains South

tns:Title

Unknown

XPath	/ImageMetadata/DescriptiveMetadata/Title
Namespace URI	http://www.example.org/EuPhotoSchema
Count	5
Distinct Count	5
Average Length	15.8

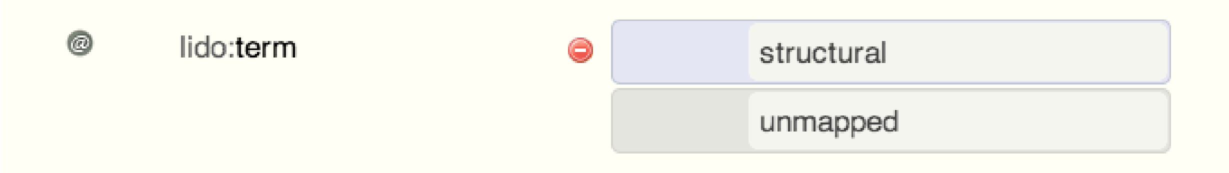
Value	Count
Caryatids	1
Parthenon	1
Parthenon East Pediment	1
Parthenon South Peristyle	1
The Parthenon	1

Figure 5-7: If else mapping

5.2.3 Structural on any element

Another very important mapping functionality available in the new MINT release is the “structural on any element”. The need for that functionality occurred from the complexity of some metadata standards. Some of the providers have metadata exports in schemas that had very extensive hierarchical structure. When attempting to map the very well defined information to a schema that was less expressive they could not exploit their in house structure in their mapping.

The “structural on any element” mapping functionality solves that problem by allowing content providers to threat each target schema’s simple element (i.e. ‘lido:term’) as a complex type, thus taking advantage of their in house metadata hierarchy and of the mappings that can be set to a structural element through MINT. The structural mapping area appears by clicking on the elements’ name.



lido:term

structural

unmapped

Figure 5-8: Structural on any element

5.2.4 SKOS Vocabularies Support

MINT’s functionality extended to support SKOS vocabularies by the development of an additional module. More specifically a semantic repository has been set up to which the SKOS vocabularies are stored. The communication of the MINT mapping tool with it is established by using SPAQRL 1.1 to retrieve the vocabularies’ terms based on the SKOS specification. Additional semantic properties can be added – if necessary – to the vocabularies for controlling selectable and non-selectable terms through the mapping tool (skos:member, skos:Collection) and also for selecting to display only subcategories of them (skos:inScheme, skos:ConceptScheme).

5.3 MINT OAI-PMH server

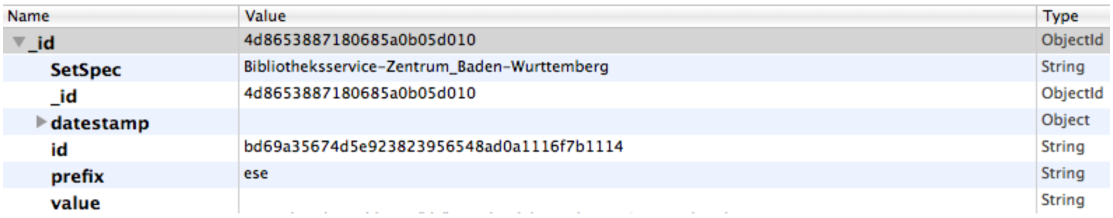
A core characteristic of the MINT ingestion platform is that of being agnostic with respect to the schema of an imported dataset, a characteristic that is also inherited by the OAI-PMH service. For this to be achieved the data layer of the platform has to be able to handle heterogeneous metadata schemata. Based on this assumption, the metadata layer is implemented using a NoSQL solution that does not enforce any particular schema, thus being able to adapt to varying metadata models.

The NoSQL solution that is used for the MINT OAI-PMH Platform is the MongoDB[11] 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 stands for a binary serialization of the JSON model. It allows the existence of JSON documents in the same database or even collection having different fields and thus 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 application because the stored data do not have to be transformed to a different format in order to be consumed by the applications.

The MINT's OAI-PMH repository is designed around three distinct collections that exist inside a MongoDB database. 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:

- Registry: in this collection the actual metadata records are stored and accessed by the implemented OAI-PMH verbs.
- Conflicts: every time a new dataset is imported in the OAI-PMH repository, it is checked for the existence of duplicate records, and if any exist, they are reported and stored in a different collection while they are also associated with a specific Report document.
- 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 the user wishes to be exposed via the OAI-PMH repositories. Each record is stored inside a JSON document, which also 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 date stamp 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 5-9.



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 5-9: Structure of the registry document

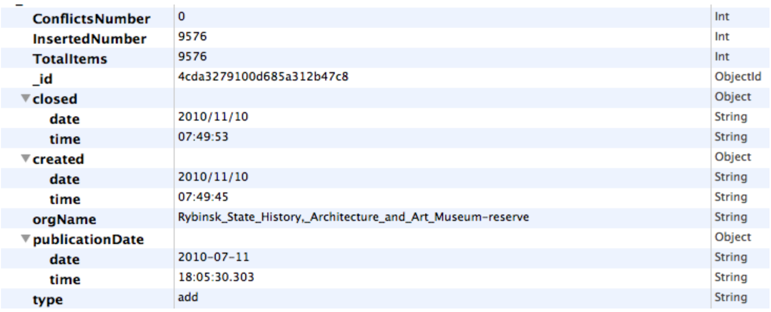
As it was mentioned earlier, another important collection of documents that is stored as part of the OAI-PMH repository is the Reports collection. The documents that are stored in this collection represent a set of valuable information that corresponds to specific actions of the repository

[11] http://www.mongodb.org/

platform. These actions are stored as values in the type attribute of the document and take one of the following values:

- Add: this type represents an action in which records are added in the registry.
- Update: this type represents an update action in which a set for a specific import already exists and it is updated by adding new metadata records.
- Delete: deletion of records from a specific import that already exists in the registry.

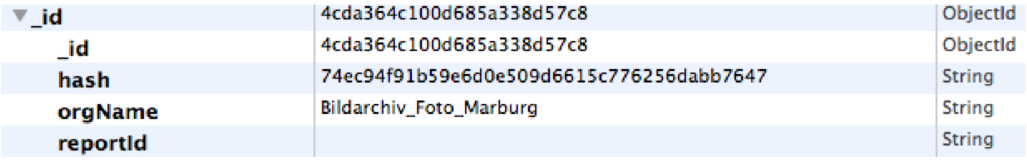
Apart from the action type, a number of other values are also stored as part of the Report Document. More specifically, a set of valuable statistics are stored; the number of conflicted items that were identified, the total number of the inserted records and the total number of items which corresponds to sum of the inserted records plus the conflicts. Two date stamps 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, thus enabling the calculation of 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 5-10.



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 5-10: 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 publishing of a dataset 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 inFigure 5-11. It should 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 since it provides a mechanism for creating unique ids for the metadata records and also a mechanism for identifying duplicates.



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

Figure 5-11: Structure of the conflict document

On top of the data layer described so far, a set of functionalities is built. More specifically there is 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[12] web application, which provides an abstract implementation of the OAI-PMH v2.0 specified verbs that can be customized in order to operate on top of different data layer technologies (e.g. flat XML files, relational databases etc.). 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 that 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 of 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 that is needed, 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

[12] www.oclc.org/research/activities/oaicat/default.htm

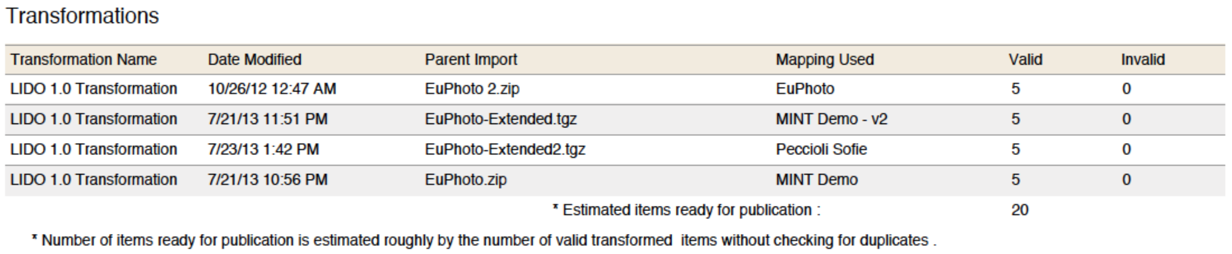
instance. Finally, by being a servlet implementation, the OAICat specific instance can be served through any of the available servlet contains that exist, e.g. Tomcat, JBoss, Jetty etc. Currently it is served via a running Apache Tomcat instance.

An 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 this 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[13] project, a functionally complete, high performance implementation of the IETF Atom 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.

5.4 Metadata progress reports

In order to improve the monitoring of the metadata production and facilitate thepublication to Europeana a new module has been implemented for the LoCloud MINT instance that allows content providers to download reports with their overall progress.

The Transformations table shows the imports that have been transformed and the resulting valid Items according to LIDO.



Transformations

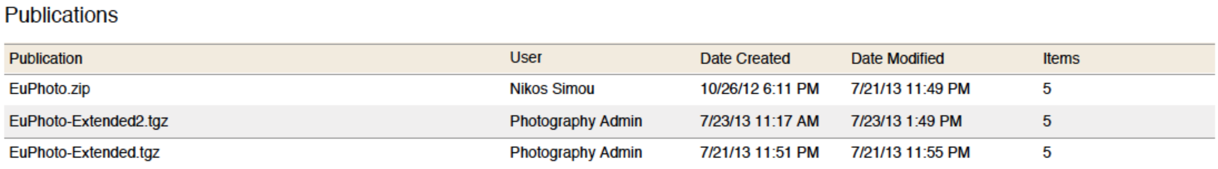
Transformation Name	Date Modified	Parent Import	Mapping Used	Valid	Invalid
LIDO 1.0 Transformation	10/26/12 12:47 AM	EuPhoto 2.zip	EuPhoto	5	0
LIDO 1.0 Transformation	7/21/13 11:51 PM	EuPhoto-Extended.tgz	MINT Demo - v2	5	0
LIDO 1.0 Transformation	7/23/13 1:42 PM	EuPhoto-Extended2.tgz	Peccioli Sofie	5	0
LIDO 1.0 Transformation	7/21/13 10:56 PM	EuPhoto.zip	MINT Demo	5	0
		* Estimated items ready for publication :		20	

* Number of items ready for publication is estimated roughly by the number of valid transformed items without checking for duplicates .

Figure 5-12: MINT progress report (Transformations)

The publications table presents the datasets that have been made available through MINT.

[13] http://abdera.apache.org/

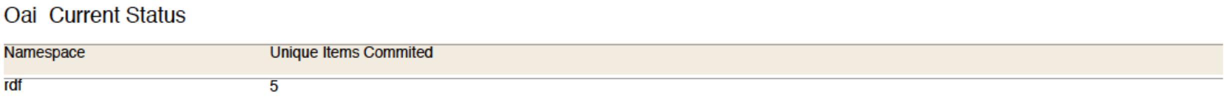


Publications

Publication	User	Date Created	Date Modified	Items
EuPhoto.zip	Nikos Simou	10/26/12 6:11 PM	7/21/13 11:49 PM	5
EuPhoto-Extended2.tgz	Photography Admin	7/23/13 11:17 AM	7/23/13 1:49 PM	5
EuPhoto-Extended.tgz	Photography Admin	7/21/13 11:51 PM	7/21/13 11:55 PM	5

Figure 5-13: MINT progress report (Publications)

The OAI current status table shows that number of unique records that are on NTUA's OAI-PMH server and are ready to be harvested by Europeana (in this case the same records have been published three times resulting to 5 unique records in OAI-PMH)

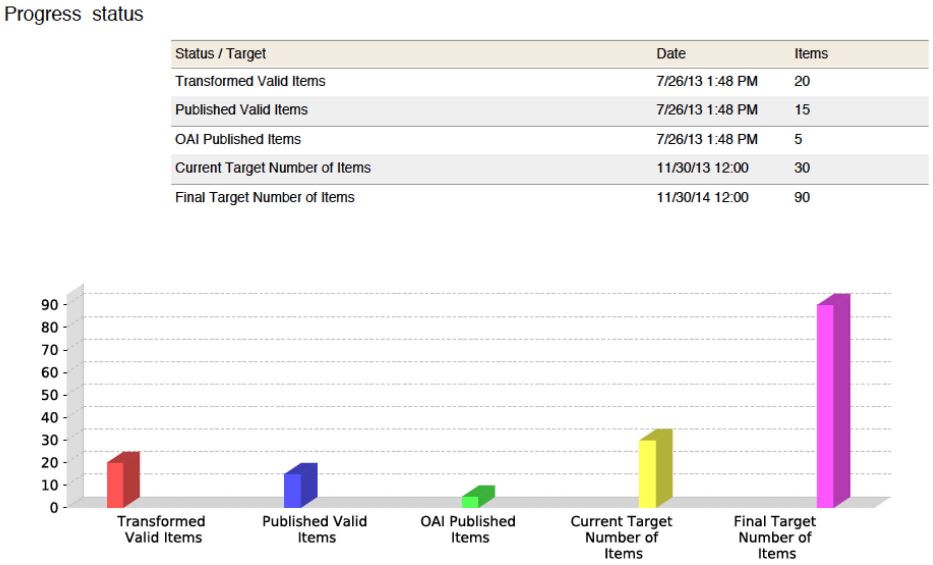


Oai Current Status

Namespace	Unique Items Commited
rdf	5

Figure 5-14: MINT progress report (OAI-PMH)

Finally, the progress status table shows the total number of transformed valid items, published (made available for LoCloud repository) valid items, OAI published items and also the current and the overall target assigned to content providers.



Progress status

Status / Target	Date	Items
Transformed Valid Items	7/26/13 1:48 PM	20
Published Valid Items	7/26/13 1:48 PM	15
OAI Published Items	7/26/13 1:48 PM	5
Current Target Number of Items	11/30/13 12:00	30
Final Target Number of Items	11/30/14 12:00	90

Figure 5-15: MINT progress report (Progress status)

6 MINT support

The transformation and the publication of providers' metadata to the selected target schema (LIDO, Carare2.0, EDM) is a process during which providers may encounter difficulties, even if they are well aware of the MINT's functionality. This is because the majority of the providers do not have a strong technical background and it may be hard for them to follow the ingestion workflow combining MINT's functionalities to reach the desired result. For that reason a help desk support system will be set up that in order to assist the content providers during all the phases of ingestion in the LoCloud project. More specifically the content providers will be able to get instructions on how to map their in-house metadata to the selected target schema for exploiting its full expressiveness, as well as technical information about MINT functionalities that will allow them to fit their metadata perfectly according to the LoCloud requirements. In addition, a set of explanatory documents concerning the various aspects of importing metadata to MINT and mapping to the target schema (LIDO, Carare2.0, and EDM) will be available online.

Online documentation and instruction of step-by-step usage of MINT can be found at: http://mint.image.ece.ntua.gr/mint2/documentation/.

7 Technical Specifications

7.1 Platform

The platform is developed using JAVA, JSP, HTML and Javascript. It uses PostgreSQL as an object-relational database with Hibernate as the data persistence framework, and mongoDB as a document-oriented database. MINT is also reusing other open source development frameworks and libraries according to specific deployments and customizations. MINT source code versions are released under a free software license (GNU Affero GPL).

The MINT platform offers the user an organisation management system that allows the deployment and operation of different aggregation schemes with corresponding user roles and access rights. A Restful web service is available for user management and authentication.

7.2 Ingestion

Registered users can upload their metadata records in XML or CSV serialization, using the HTTP, FTP and OAI-PMH protocols. Users can also directly upload and validate records in a range of supported metadata standards (XSD). XML records are stored and indexed for statistics, previews, access from the ingestion platform and subsequent services. Current developments aim to support relational database schemata and OWL/RDFS ontologies as input.

7.3 Processing

Handling of metadata records includes indexing, retrieval, update and transformation of XML files and records. XML processors (Apache Xerces, SAXON, Nux) are used for validation and transformation tasks as well as for the visualization of XML and XSLT. For issues of scalability with respect to the amount of data and concurrent heavy processing tasks, parts of the services are multi-threaded or use specific queue processing mechanisms.

7.4 Normalization & Vocabularies

Various additional resources such as terminologies, vocabularies, authority files and dictionaries are used to reinforce an aggregation's homogeneity and interoperability with external data sources. A typical usage scenario is the connection of a local (server) or online resource with a metadata element in order to be used during mapping/normalization. The vocabularies have to be represented in SKOS.

8 MINT Integration in LoCloud core infrastructure

MINT integration into LoCloud infrastructure aims at:

- Exposing MINT data and services via appropriate API
- Interaction with core infrastructure API (D2.1)
- Publication to Europeana through MORE (D2.3)

Data providers and Organizations that need to convert their data into the Carare 2.0, LIDO or EDM format can use the new MINT release to easily create the necessary mapping. Once the data transformed into the target schema and validated, MINT offers to transfer the data to the MORE system.

For this to work, MINT and MORE cooperate via a web service like API. MINT announces to MORE the finished dataset and provides a download token, which authorizes the MORE service to download the prepared dataset. This happens via a URL request to a specified MORE service. MORE then uses this token to download a tar-gzip archive of the published data. It contains a package.xml file to describe the whole publication, along with subdirectories for each published item. Each item is supplied with source and transformed xml, the generated and utilized XSL file and some general information contained in an info.xml file.

This integration has been set up and tested in previous projects (Carare, 3d-Icons) and is based on a standardized and extendable approach based on HTTP APIs. For the final deployment and integration in the project's cloud enabled infrastructure the aforementioned approach can be updated and/or extended according to new requirements that may arise during the core infrastructure implementation.

9 Conclusion

The present document constitutes the report of deliverable D 2.2 “Modified MINT prototype” that is made available online for validation and for the large-scale contribution of content to Europeana and for dissemination & training. The MINT ingestion platform uses once-only mappings and simple re-use of local “source” metadata and in that way takes full advantage of the funded project to make a very low-cost continuation possible. NTUA hosts the tool and the provision of these services has minimal additional cost.

The platform implements an aggregation infrastructure offering a crosswalk mechanism to support subsequent critical activities:

- harvesting and aggregating metadata records that were created using shared community standards or proprietary metadata schemas ;
- migrating from providers’ models (whether standard or local) to a reference model ;

9.1 Results

The objective of the deliverable is the deployment of the MINT ingestion platform (Task 2.2.1), that is available online to the project partners at http://mint-projects.image.ntua.gr/locloud