DISMARC
discovering music archives

eContentplus Programme
Targeted Project
Grant agreement Number ECP-2005-CULT-038202

D2.1 Report and technical specification for each site configuration

Technical Document WP 2

Version 2

November 2007

eContent*plus*

DISMARC
discovering music archives

D2.1 Report and technical specification for each site configuration

Technical Document TD 2.1

November 2007

Project acronym	DISMARC	**Contract nr.**	ECP-2005-CULT-038202
Type and Number	D2.1		
Work package	2		
Date of delivery	2007-11-07		
Code name	DM_D2.1_v01	**Version** 2 draft ☐ final ☐	
Objective	Technical Document		
Distribution Type	Internal		
Distributed to	DISMARC partners		
Authors (Partner)	AIT		
Abstract	The aim of this report is to set forth the functional and technical requirements of the DISMARC system, and to describe the DISMARC core application, the MetaDataManager.		
Keywords List	DISMARC MetaDataManager, Functional Requirements, Technical Requirements, Components, Procedures		
Version log	Version 1: internal version October 2007 Version 2: 1st official version of D2.1		

ANGEWANDTE INFORMATIONSTECHNIK
Forschungsgesellschaft mbH

eContent*plus*

Table of contents

1 Abstract

The aim of the deliverable D2.1 within the DISMARC project is to provide a report of the functional and technical requirements of the DISMARC system, and to describe the DISMARC core application, the MetaDataManager. All DISMARC partners may run an instance of the MetaDataManager at their local sites given that the described software and hardware requirements are met. The data from these local instances will be harvested by the central DISMARC store that will in future communicate with other metadata stores like the TEL (The Euroepan Library) network.

The description of the DISMARC functional requirements is an ongoing cooperative process within the consortium that started right from the start of the project. Therefore these descriptions are outlined in a separate document, the Technical Document TD2.7 which is part of D2.1 and attached to this document.
The technical descriptions within D2.1 make use of UML2.0 diagrams as described at http://www.uml.org by the Object Management Group (http://www.omg.org) and uses software engineering and design vocabularies introduced by the Gang-of-Four ("Design Patterns - Elements of Reusable Object-Oriented Software").

2 Introduction

2.1 The DISMARC Vision

The purpose of the DISMARC project is to establish a metadata platform for the discovery and diffusion of knowledge concerning music data. The project addresses problems of interactivity between different systems, which are mainly caused by varying methods of metadata description. The fields from the partners' databases are mapped onto a project-defined DISMARC metadata standard format based on the international Dublin Core Standard.
The DISMARC metastore provides a one-stop search facility for different user groups (the academic community, the general public and the media) and helps them to find and access the vast and often undisclosed number of audio and audio-related data held in various sound archives.

Furthermore DISMARC is seen as the audio pillar for the European Community's TEL portal (The European Library, common search access for the national libraries in Europe). All DISMARC metadata is exposed to be harvested by the TEL-harvester and copied onto the central TEL-System which allows integration of the DISMARC-metadata into the overall system (indexing, searching, browsing). By this means DISMARC metadata (coming from all partner catalogues) can be searched simultaneously together with all other TEL catalogues' metadata.

2.2 Short Overview on the DISMARC System

The DISMARC system is designed to be simple to use, quick and flexible, in order to become the preferred contact instrument among the various partners of a network offering access to a virtual sound archive.

The DISMARC Internet portal offers a system for content management, a validation system, a metastore and - also - an archive for keeping documents (audio samples).
Together with the metadata the digitized sound files are optionally stored in low quality in the metastore, and are available through the portal. It is not the objective of the DISMARC metastore to store the original objects (with all their administrative information) and the original audio files.
The metastore offers concise general information on the objects (the object metadata description and a short preview if possible), thus allowing users an initial selection of the objects that attract their attention. More detailed information (the original data) can then be retrieved directly from the supplying partner via a link.
The user may search for objects via a so-called "Google search" and a Simple search which smoothly becomes an Advanced search when adding further metadata fields to the search.

DISMARC is also planning to tailor a service to the needs of those archives that do not dispose of an own system but nevertheless want to join DISMARC. Such archives will be able to add information on their collections to the DISMARC metastore via the DISMARC collection description tool. They will be searchable on collection level.

Users can access the metastore via a standard Web browser. An authentication procedure enables individualized access.

The metastore harvests the metadata from any different automated systems (providers) and provides powerful search capabilities for searching this diverse data. This eliminates the need to know how to search objects in individual systems.
Users can locate the objects by browsing and searching in different metadata fields or via full text searches. In addition several different controlled vocabularies (tree structures, lookup lists) are available. A user could then find objects by drilling down any of the word lists or vocabulary trees. If a user double-clicks on a specific tree (or list) item the term will be automatically added as a search term.

New metadata is harvested automatically in pre-defined intervals as soon as is provided by the content partners OAI provider.

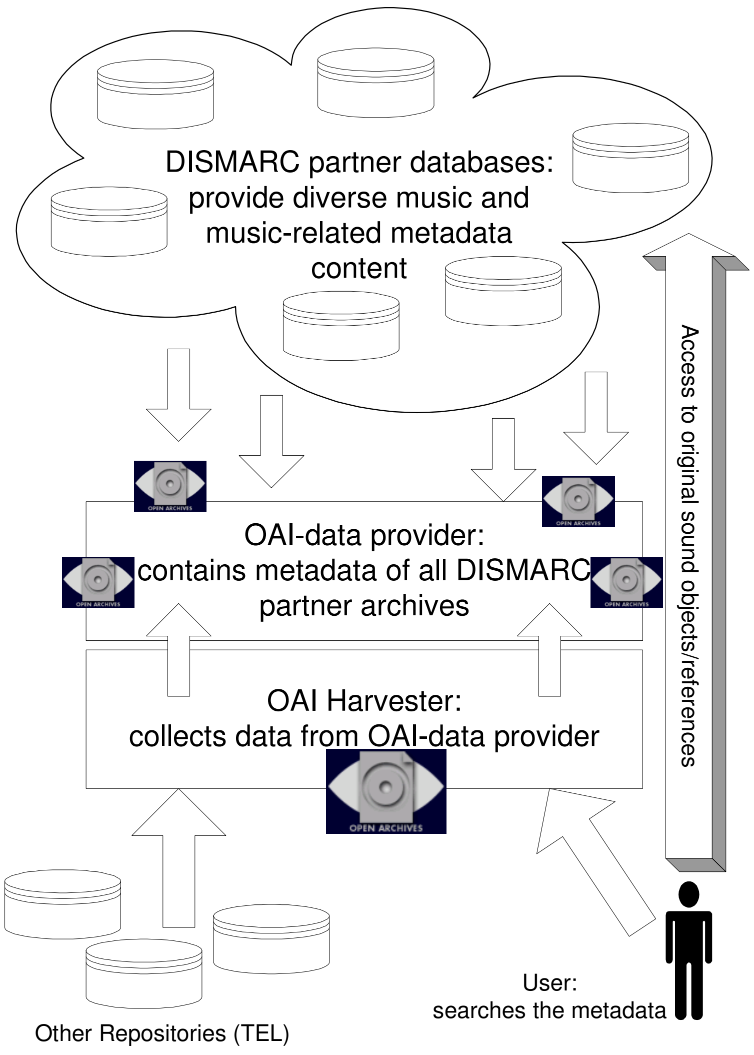
DISMARC partner databases: provide diverse music and music-related metadata content
OAI-data provider: contains metadata of all DISMARC partner archives
OAI Harvester: collects data from OAI-data provider
Access to original sound objects/references
Other Repositories (TEL)
User: searches the metadata
OPEN ARCHIVES

Figure 1 DISMARC Overview

3 Requirements

The requirements discussed in this document are stripped down the categories shown below. All other non functional requirements are not essential for the scope of this report and are therefore not depicted here.

3.1 Functional Requirements

The functional requirements for the DISMARC system are detailed within a separate document, the Technical Document TD2.7, which is constantly being updated and amended. The document describes the various use cases for the DISMARC browser and the administration area. The current status of TD2.7 is Version 5 (August 2007).
These are the contents of the available versions so far:

V01: The document describes the user requirements for the search&retrieval within the DISMARC MetaStore. (March 2007)
V02: The document incorporates several updates and the remarks received till 21-03-2007. (March 2007)
V03: The document incorporates updates in the search and retrieval section and the new Part B: Administration Area has been added. (May 2007)
V04: The document incorporates updates in the search and retrieval and the Administration Area section and the part of News Alert has been added. (July 2007)
V05: The document incorporates partner's proposals and improvements of details. (August 2007)

To date TD27 incorporates the following contents:

- General Functionalities
- DISMARC Components description
- User groups
- Security (2 Use Cases)
- Search & Retrieval (11 Use Cases)
- Administration Area (28 Use Cases)
- News Alert (2 Use Cases)

In order to ease readability, this document ships with the current TD27 document. Please see DM_TD27_v05_f.rar for further information.

3.2 Technical Requirements

This section discusses the technical (both, hard- and software) requirements which a server has to meet in order to run an instance of the MetaDataManager (from here on referred to as MDM).

Apache/IIS	Since the MDM is a web application, a webserver is needed. There are no special preferences for its implementation	
PHP 5.2	The MDM is written as object oriented PHP application. Therefore the use of the latest PHP version is recommended.	http://www.php.net
MySQL 5	Due to the need of a relational database as user and setting storage and the native support of MySQL5 by PHP, MySQL5 was the choice for MDM. Nevertheless, also the use of a Postgres database is possible	http://www.mysql.com

Zebra Server	The Zebra Server supplies the core functionality for the collection and object metadata. The Zebra Server is used to index and retrieve the generated xml records.	http://www.indexdata.dk
Iconv	Inconv is a unix tool to convert the encoded import files to MDMs internal “utf-8” encoding. Every common *nix distribution ships with iconv, whereas the support for MS Windows is supplied by cygwin.	

Table 1 Software requirements

The hardware requirements are taken from AIT’s current development server, which is running the prototype.

1 GB ram	Just running the MDM instance will never use more than 32MB, but during indexing and record creation, more ram is needed.
12 GB HDD	The MDM itself just needs a few MB but the Zebra Index needs gigabyte capacity. This is due to the huge search possibilities. The used space can be reduced by adapting the configuration to remove unused index entries.

Table 2 Hardware requirements

4 DISMARC's Core, the MetaDataManager

This section introduces the MDM, describes its basic architecture and shows the process of several user interactions.
MDM is a PHP based application able to index, search and retrieve information stored as xml. The access to this information is restricted by several access right management implementations.
The MDM is also capable of searching information in several languages. This is accomplished through a user defined dictionary and other controlled vocabularies like thesauri.

Figure 2 MDM instances shows the basic implementation of DISMARC at node level.
Each partner and archive will have its own MDM installed on their web server. In case an archive does not have the resources to run their own instance, a bigger archive or one of the partners can add this metadata to their repository.

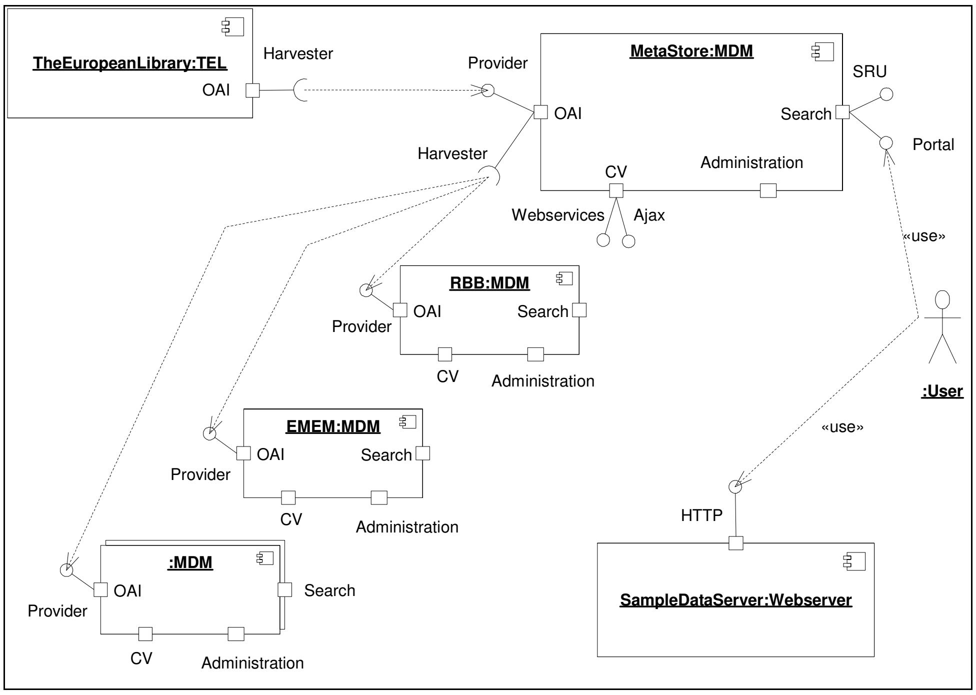

Figure 2 MDM instances

All the nodes are capable of running DISMARC on their own, but there will be a master instance at an RBB server which stores the accumulated DISMARC data and is the MDM that will communicate with OAI harvesters of an higher instance like TEL.
This is realized via an OAI provider and a harvester implementation based on the archives metadata.

The partner nodes are responsible for their own data. Therefore only checked imported metadata should be added to the network. The harvester will copy the information provided by other DISMARC nodes at predefined intervals.
Since this harvesting can be done at an infinite long chain, one can see it as a harvesting tree where smaller archives will get harvested by bigger archives, country nodes or topic nodes which will then be harvested by the central node.
This is the preferred way since the administrative costs are shared among several nodes.

The SampleDataServer provides samples or whole songs if rights have been cleared. A new version of this SampleDataServer is the audio database discussed at the Seville consortium conference in October 2007. Since this is a completely new requirement not mentioned in the technical annex, more detailed information can only be supplied in a following version of this document.

4.1 Components

As already shown the MDM provides various interaction points, which will be discussed in this section.

Figure 3 MDM components displays an abstract view of the application model.

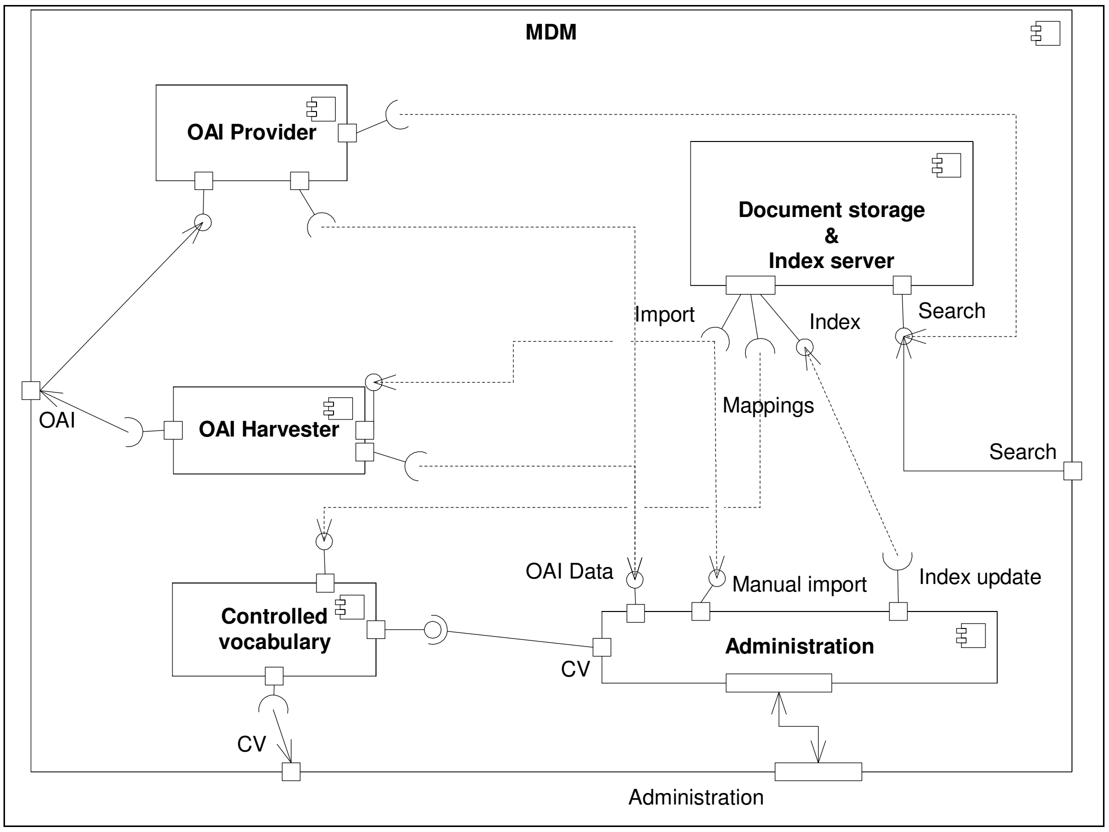

Figure 3 MDM components

Each subcomponent has its own interfaces which communicate within the MDM. The MDM acts as facade to the user for these subcomponents and implements the control and information flow in some use cases.

Both OAI components are implemented as adapters for the other MDM components and the OAI protocol. For more detailed information please see the Open Archives Initiative documentation on http://www.openarchives.org .

One of the first subcomponents discussed should be the document storage and index server. The abstract view of this components can be found in Figure 4.

Document storage provides search and indexing functionalities and is using the import and mapping interface from other components.

Import, mapping and index are the functionalities needed during the import of new or updated records and are delegated to the Zebra Server and record normalization subcomponent. More information can be found at 4.2 Procedures.

The Zebra Server holds two databases. One is the object database which stores indexes and information about each record whereas the collection database stores collection related information. Thanks to this concept, an infinite number of collections and collection levels can be supported (only restricted by available disk space) and all search functionalities can be used for both databases.

The query normalization is used to create a proper search string for the Zebra Server.

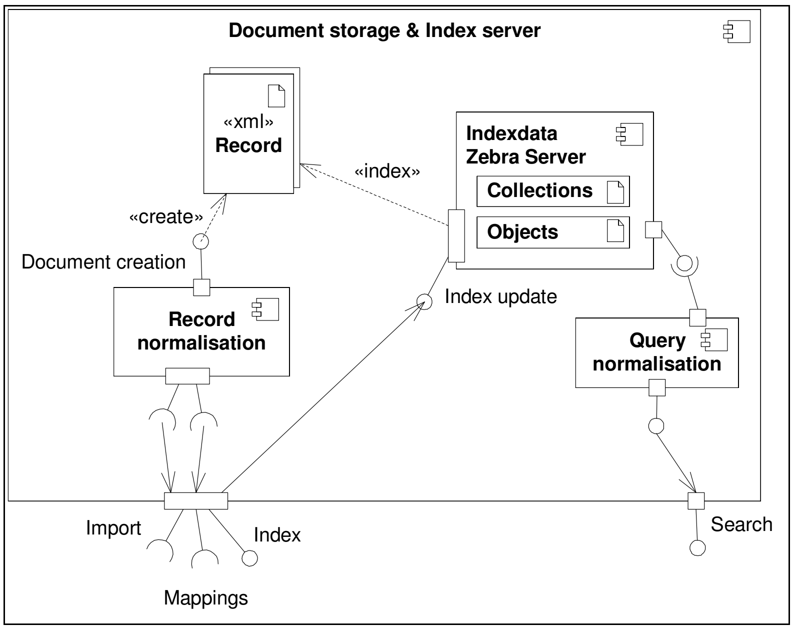

Figure 4 Document storage & index server

An additional quite complex component is the administration area. Most of the settings (settings which might break the environment of the MDM can only be set on file access level) can be accessed via this component.

Figure 5 Administration shows the subcomponents of the administration area. Settings control the appearance of the website, the access rights, the import behavior as well as the OAI namespace configuration and control.

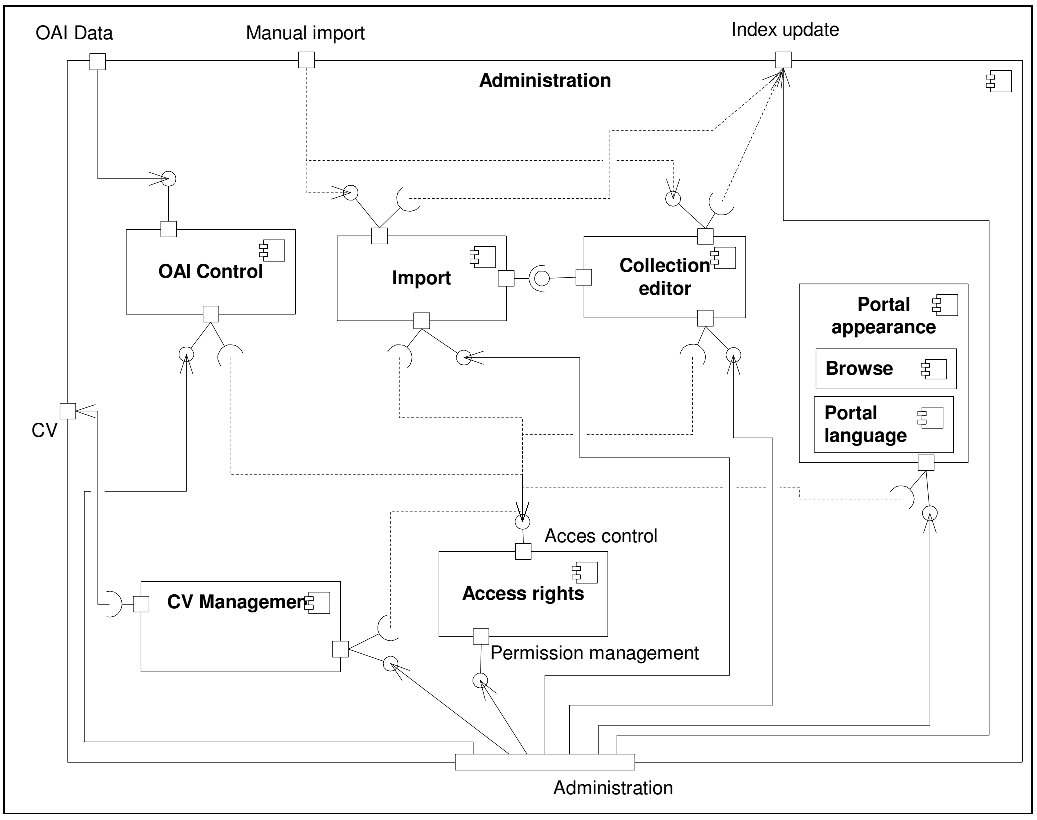

Figure 5 Administration

4.2 Procedures

The core user interactions are search and import, whereas the import will be done automatically in future. The basic setup of the import can be seen in Figure 6 Import sequence diagram.

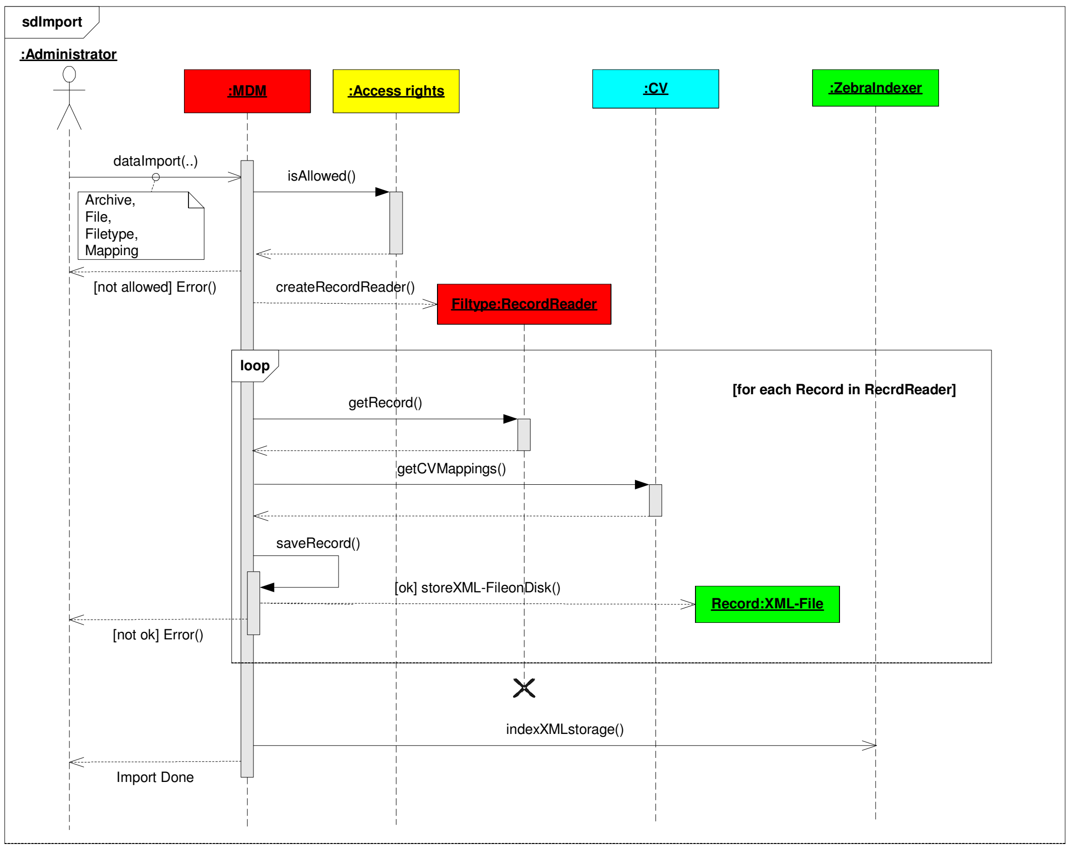

Figure 6 Import sequence diagram

The administrator tells the MDM to import a new file for an archive. MDM checks in the access right module if the user is allowed to do so. With the file, its encoding and type definition, a record reader is generated. MDM then loops through the records, checks the CV for mapping information and creates a new xml file which will then be indexed by the Zebra Server.

This quite easy process becomes considerably comprehensive during the mapping. Thesauri terms have to be mapped and native field information be ripped apart to match the DISMARC application profile to represent the native data without the loss of any information at the same time.

The search process as shown in Figure 7 Search sequence diagram looks even simpler than the import process. With the easy process, the result gets broad. Due to the user preferences MDM selects additional words to search with from the dictionary based on the user's known languages and current language settings.
This results in a broader result set even in non-multilingual searches because the dictionary also contains synonyms for search terms the user might not have thought about.

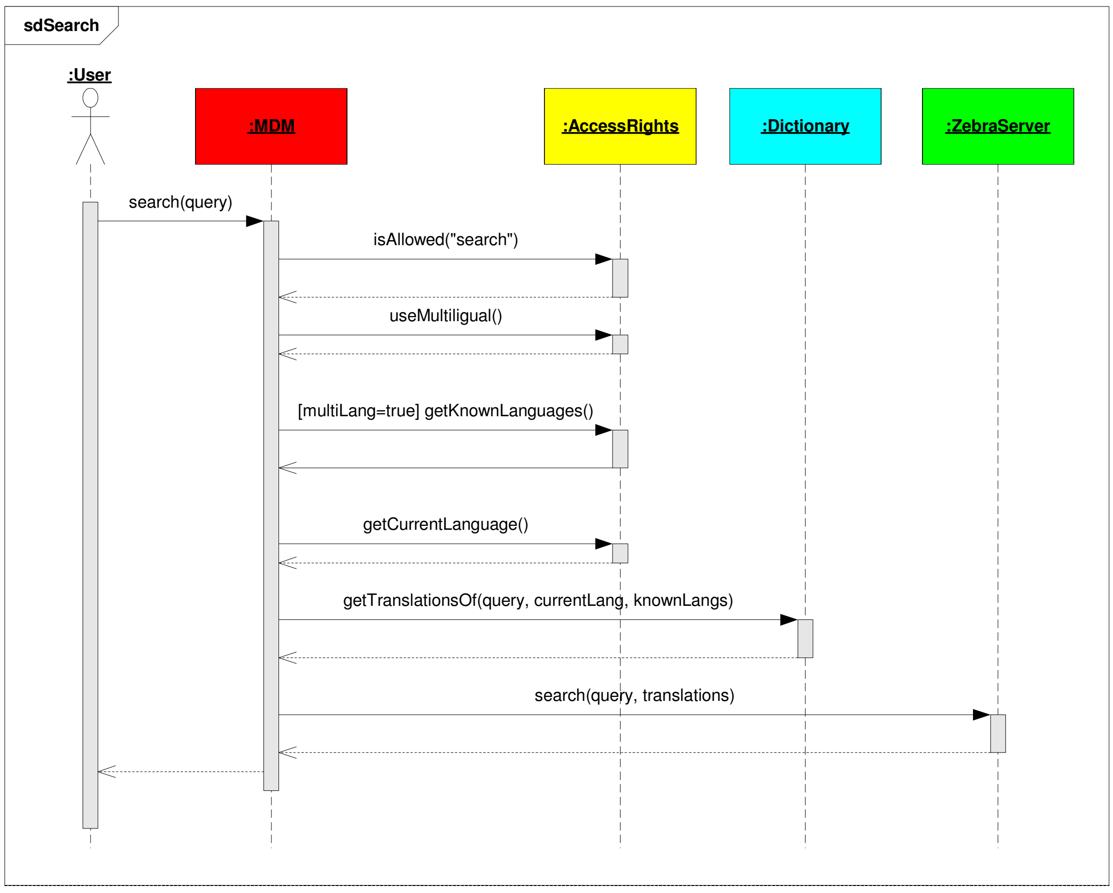

Figure 7 Search sequence diagram

I Acronyms

CV	Controlled Vocabulary	
GB	Gigabyte	
MDM	MetaDataManager, the PHP application.	
HDD	Hard Disk Drive	
HTML	Hyper Text Markup Language	
HTTP	Hyper Text Transfer Protocol	
IPTC	International Press Telecommunications Council	http://www.iptc.org/
MySql	Open Source Database	http://www.mysql.com/
LQ	Low Quality	
OAI	Open Archives Initiative	http://www.openarchives.org
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting	http://www.openarchives.org/OAI/openarchivesprotocol.html
ODBC	Open Database Connectivity	
OMG	Object Management Group	http://www.omg.org/
ONTOLOGY	An ontology is a specification of a conceptualisation.	http://www-ksl.stanford.edu/kst/what-is-an-ontology.html
OPAC	Online Public Access Catalogue	
PHP	PHP: Hypertext Preprocessor	www.php.net
RAM	Random Access Memory	
REGNET	Cultural Heritage in REGional NETworks (IST-Research Project IST-2000-26336)	http://www.regnet.org
RSS	Really Simple Syndication (RSS) is a lightweight XML format designed for sharing headlines and other Web content.	
SOAP	Simple Object Access Protocol providing a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralised, distributed environment using XML proposed under the w3c.	http://www.w3.org/TR/SOAP/
SQL	Structured Query Language: ISO, ANSI standard user front end to a relational database management system.	
SRU	SRU, the Search and Retrieve URL Service, is a companion service to SRW, the Search and Retrieve Web Service. Its primary difference is its access mechanism: SRU is a simple HTTP GET form of the service	http://www.loc.gov/z3950/agency/zing/srw/sru.html
SRW	SRW defines a web service for searching databases containing metadata and objects, both text and non-text. The SRW Initiative builds on Z39.50 along with web technologies. Building on Z39.50 semantics enables the creation of gateways to existing Z39.50 systems; web technologies reduce	http://www.loc.gov/z3950/agency/zing/srw/

	the barriers to new information providers allowing them to make their resources available via a standard search and retrieve service.	
UC	Use Case	
UID	Unique Identifier	
UML	Unified Modelling Language	http://www.uml.org
URL	Uniform Resource Locator	
XML	EXtensible Markup Language	
Z39.50	National Information Standards Organization Z39.50 Information Retrieval Protocol (Z39.50/ISO 23950), a computer protocol that can be implemented on any platform, defines a standard way for two computers to communicate for the purpose of information retrieval. standard (= ISO 23950)	http://www.niso.org/z39.50/z3950.html

II Figures

III Tables