Project no. 027087

TENCompetence

Building the European Network for Lifelong Competence Development

Project acronym: Integrated Project TENCompetence

Thematic Priority: 2.4.10

**ID5.5: API definitions to be delivered to WP3**

Due date of deliverable: 31-05-2007
Actual submission date: 26-07-2007

Start date of project: 01-12-2005
Duration: 4 years

GIUNTI Interactive Labs s.r.l.

Version 1.0

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PU</strong> Public</td>
<td>X</td>
</tr>
<tr>
<td><strong>PP</strong> Restricted to other programme participants (including the Commission Services)</td>
<td></td>
</tr>
<tr>
<td><strong>RE</strong> Restricted to a group specified by the consortium (including the Commission Services)</td>
<td></td>
</tr>
<tr>
<td><strong>CO</strong> Confidential, only for members of the consortium (including the Commission Services)</td>
<td></td>
</tr>
</tbody>
</table>
# Project Internal Deliverable Report

**ID5.5: API definitions to be delivered to WP3**

<table>
<thead>
<tr>
<th>Workpackage</th>
<th>WP5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>1</td>
</tr>
<tr>
<td>Date of delivery</td>
<td><strong>Contractual:</strong> 31-05-2007  <strong>Actual:</strong> 26-07-2007</td>
</tr>
<tr>
<td>Code name</td>
<td>ID5.5  <strong>Version:</strong> 1.0  <strong>Draft:</strong> ☐  <strong>Final:</strong> ☑</td>
</tr>
<tr>
<td>Type of deliverable</td>
<td>Report</td>
</tr>
<tr>
<td>Security (distribution level)</td>
<td>Public</td>
</tr>
<tr>
<td>Contributors</td>
<td>Elena Demidova (UHANN), Philipp Kaerger (UHANN), Krassen Stefanov (Sofia University), Carlos Mendez (Altran SDB), Michele Dicerto (Giunti Labs)</td>
</tr>
<tr>
<td>Authors (Partner)</td>
<td>Giunti Labs</td>
</tr>
<tr>
<td>Contact Person</td>
<td>Michele Dicerto (Giunti Labs), A. Marco Luccini (Giunti Labs)</td>
</tr>
<tr>
<td>WP/Task responsible</td>
<td>Giunti Labs</td>
</tr>
<tr>
<td>EC Project Officer</td>
<td>Hans-Jürgen Westhof</td>
</tr>
</tbody>
</table>

**Abstract (for dissemination)**

Report on WP5 (Knowledge Resources Sharing and Management) activities.

**Keywords List**

- WP5, deliverable, knowledge management, knowledge resources, state-of-art, social networks, glossary, actors, stakeholders, scenarios, use cases, metadata, KRSM, system, interfaces, usability, sharing, standards, specifications, architecture, API, GUI, federated search, digital repositories, P2P, share, manual, client, proof-of-concept, model, method, validation, plan, roadmap.

---

TENCompetence Project Coordination at: Open University of the Netherlands
Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands
Tel: +31 45 5762624 – Fax: +31 45 5762800
# Table of Contents

TABLE OF CONTENTS .......................................................................................................................... 1
LIST OF TABLES ................................................................................................................................. 2
LIST OF FIGURES ............................................................................................................................... 2
1 INTRODUCTION ............................................................................................................................... 3
2. INTERFACES ....................................................................................................................................... 5
  2.1 SEARCH .......................................................................................................................................... 5
  2.2 PUBLISHING OF KNOWLEDGE RESOURCES ............................................................................. 5
  2.3 RESOURCE RATING ....................................................................................................................... 6
  2.4 INFORMATION SOURCE MANAGEMENT .................................................................................. 6
  2.5 USER MANAGEMENT .................................................................................................................... 6
  2.6 RESOURCE DOWNLOAD .............................................................................................................. 7
3. SERVICE ARCHITECTURE & IMPLEMENTATION ........................................................................... 8
  3.1 INFORMATION SOURCE LAYER .................................................................................................. 9
    3.1.1 Web 2.0 Tools ....................................................................................................................... 9
    3.1.2 Centralised Repositories ..................................................................................................... 10
    3.1.3 P2P Networks ..................................................................................................................... 10
  3.2 SERVICE LAYER .......................................................................................................................... 10
    3.2.1 Information Source Management Service ........................................................................ 10
    3.2.3 User Management Service ............................................................................................... 11
    3.2.4 Resource Rating Service ................................................................................................... 11
  3.3 CLIENT LAYER ................................................................................................................................ 11
4 INTEGRATED REPOSITORIES AND COMPONENTS ..................................................................... 12
  4.1 FEDERATED SEARCH ENGINE .................................................................................................. 12
  4.2 LIONSHARE AND LIMEWIRE FOR P2P ................................................................................... 12
  4.3 ARIADNE ..................................................................................................................................... 13
  4.4 TASTE .......................................................................................................................................... 13
    Taste overview ............................................................................................................................... 13
    Description .................................................................................................................................... 14
    Implementation ............................................................................................................................. 14
    Use of the Taste service ................................................................................................................ 16
  4.5 FLICKR ....................................................................................................................................... 17
  4.6 YOUTUBE .................................................................................................................................... 17
  4.7 EXACT LOBSTER ....................................................................................................................... 18
  4.8 EXACT PACKAGER SEARCH PLUGIN ..................................................................................... 19
    Configuration file .......................................................................................................................... 22
5 NEXT STEPS ...................................................................................................................................... 23
6 CONCLUSIONS ................................................................................................................................... 24
ANNEX A .............................................................................................................................................. 25
1 GUI ...................................................................................................................................................... 26
  1.1 EXPECTED FINAL GUI (FUNCTIONAL VIEW) ........................................................................... 26
2 KRSM MANUALS ............................................................................................................................... 35
List of Tables

**TABLE 1 - “TASTE_PREFERENCES” table** ................................................................. 15

List of Figures

**FIGURE 1 - TENC KRMS ARCHITECTURE** ............................................................ 9
**FIGURE 2 - TASTE IMPLEMENTATION** .................................................................. 16
**FIGURE 3 - eXact Packager screenshot: search on LOBSTER and ARIADNE**
reposorites ................................................................................................................. 21
**FIGURE 4 - Functional GUI keys** ........................................................................ 27
**FIGURE 5 - Functional GUI: Create KRs** ............................................................ 28
**FIGURE 6 - Functional GUI: Open / Search KRs** ............................................... 29
**FIGURE 7 - Functional GUI: Store KRs** .............................................................. 30
**FIGURE 8 - Functional GUI: Collaborative tools** .............................................. 31
**FIGURE 9 - Functional GUI: Properties** ............................................................. 32
**FIGURE 10 - Functional GUI: Administration** .................................................. 33
**FIGURE 11 - Functional GUI: Options** ............................................................... 34
1 Introduction

Knowledge resources are spread over the web, in centralised repositories, web servers and user desktops. On the one hand, centralised and monolithic repositories (on which e.g., Learning Management Systems rely) represent the traditional approach for resource sharing. On the other hand, completely decentralised networks like P2P allow users to share content without relying on a third party repository, therefore without losing control over it. This appealing approach, which is successfully in use (e.g., Edutella\(^1\) and LionShare\(^2\)), allows learners to share e.g., their desktop resources. Lately, a successful integration of these two kinds of information sources has been demonstrated\(^3\). This integration benefited from the standard search interface SQI (S. Ternier et al., 2006). However, it does not provide other services (e.g., publishing of knowledge resources). In addition, Web 2.0 applications that provide users with a motivating approach for resource sharing have recently emerged and become extremely popular. For instance, Flickr\(^4\) and YouTube\(^5\) allow for efficient sharing of photo and video resources: they already hold millions of them.

The users creating knowledge resources have to choose among the multiple publishing platforms available. Previously, this choice was mainly restricted to centralised repositories (e.g., maintained by the user’s home institution). However, lack of openness and interoperability of these repositories led to restricted sharing across the organisational boundaries, therefore reducing the number of potential learners using such material. Web 2.0 applications provide an alternative way of sharing knowledge resources, breaking the barriers of institutional repositories and making knowledge resources publicly available.

Furthermore, when searching or publishing content on different information sources, separate manual log in for each of them should be avoided. This requires a personalised single sign on service which contacts the selected repositories with the correct log in data. Finally, repositories may easily contain millions of knowledge resources. Given

\(^1\) http://www.i3s.de/deutsch/projekte/edutella.html
\(^2\) LionShare, http://lionshare.its.psu.edu/
\(^4\) http://www.flickr.com/
\(^5\) http://www.youtube.com/
a user query, thousands of search results may be returned and need to be ordered.

Thus a flexible knowledge resource infrastructure must not only allow searching through heterogeneous information sources but must also provide services like publishing, information source and user management as well as resource rating. Unfortunately, there is no standard interface that serves this purpose in a way that all information sources could be easily targeted. This lack of interoperability represents a challenging task: building an infrastructure for sharing and management of knowledge resources, that integrates all relevant services and makes them interoperable over heterogeneous information sources.

Remarks:
Please note that in order to maintain document’s consistency with other WP5 documentation (such as D5.1), and to make it self contained, some excerpts from other suitable documents have been integrated.
Please also note that all References are here reported in footnotes and detailed in the main document D5.1.
2. Interfaces

Achieving interoperability among heterogeneous information sources requires common interfaces for all provided services. The core services we consider include discovery, publishing and rating of knowledge resources as well as information source and user management.

2.1 Search

Efficient search for knowledge resources contained in repositories, P2P networks and Web 2.0 applications, requires a common search interface. There already exist a number of different standards for knowledge resource discovery with varying complexity. Among them, the Simple Query Interface (SQI)\(^6\) is an outstanding solution which allows for session based synchronous and asynchronous querying of information sources, stateful and stateless communications and it is query language and result format independent. SQI has been our choice for knowledge resource discovery due to its simplicity and flexibility for different scenarios. The API specification of SQI can be found at http://nm.wu-wien.ac.at/e-learning/interoperability/SQI_V1.0beta_2005_04_13.pdf. Its copy is accompanying this document.

2.2 Publishing of Knowledge Resources

A publishing service interface should enable a client to store knowledge resources and/or their metadata in an information source, be it locally, in an institutional repository or in a Web 2.0 application. In this way, the process is the same for any knowledge resource and information source, such that the user can perform this task without having to change the client application she is familiar with. Furthermore, in this way peers in a P2P network may obtain an additional possibility to publish their resources in an external repository in a persistent and standard manner such that these resources stay available after such peer goes offline. KRSM implements Simple Publishing Interface (SPI v.0.1)\(^7\). The API specification of SPI can be found at http://ariadne.cs.kuleuven.be/lomi/index.php/SimplePublishingInterface. Its copy is accompanying this document.

\(^6\) http://nm.wu-wien.ac.at/e-learning/interoperability/SQI_V1.0beta_2005_04_13.pdf
2.3 Resource Rating
Knowledge resources in an infrastructure like the one described above are not suitable for standard link analysis techniques like the ones used on the Web. Term frequency based ranking techniques are also not always applicable as knowledge resources do not necessarily contain textual information. A solution can be provided by enabling users to rate knowledge resources. Collaborative filtering techniques can then be used to rank the results. The Simple Resource Rating Interface (SRRI)\(^8\) provides methods for assigning such rating values to knowledge resources as well as for the retrieval of this information. In addition, this interface allows for the interconnection of recommendation systems. The API specification of SRRI can be found at http://www.l3s.de/~kaerger/interfaces/srri.pdf
Its copy is accompanying this document.

2.4 Information Source Management
Since users may interact with different information sources (e.g., for querying or publishing of knowledge resources), an interface is required in order to retrieve description of these information sources (e.g., name and URL together with information on the supported services, languages and metadata formats) as well as to select the ones to be used in subsequent tasks. This information can be obtained from the Simple Information Source Management Interface (SISMI)\(^9\). The API specification of SISMI can be found at http://www.l3s.de/~demidova/interfaces/sismi.pdf
Its copy is accompanying this document.

2.5 User Management
The Simple User Management Interface (SUMI)\(^10\) specifies several methods for retrieval and modification of user data. In addition, it manages user information required for accessing different information sources. This includes storage of the user login data for the connected information sources as well as its use for the authentication purposes. The API specification of SUMI can be found at http://www.l3s.de/~kaerger/interfaces/sumi.pdf

---
\(^{8}\) http://www.l3s.de/~kaerger/interfaces/srri.pdf
\(^{9}\) http://www.l3s.de/~demidova/interfaces/sismi.pdf
\(^{10}\) http://www.l3s.de/~kaerger/interfaces/sumi.pdf
Its copy is accompanying this document.

2.6 Resource Download
The Simple Content Obtain Interface (SCOI\textsuperscript{11}) provides methods for obtaining of knowledge resources stored in remote repositories. Using this interface selected resources can be transferred to the client and stored locally.

The API specification of SUMI can be found at http://www.l3s.de/~demidova/interfaces/scoi.pdf

Its copy is accompanying this document.

\textsuperscript{11} http://www.l3s.de/~demidova/interfaces/scoi.pdf
3. Service Architecture & Implementation

Integration of heterogeneous information sources within a single infrastructure raises the challenge of interoperability, both on the technical and on the semantic level. Due to the limitations (both in metadata and query capabilities) of some of the information sources we are considering (especially the web systems) we started with a basic core set of metadata (a subset of Dublin Core and therefore of LOM) and rely on keyword based search.

We implemented an open source infrastructure which supports searching and publishing (storage, update and deletion) of knowledge resources, as well as other advanced services like e.g., user management and resource rating. All these services implement the interfaces described in the previous section and allow for knowledge resource sharing and management in both the local system and remotely (repositories and Web 2.0 applications). Our Service Oriented Architecture (Figure 1) relies on a registry in which information sources can be added dynamically. The Knowledge Resource Sharing and Management infrastructure is built as a 3-layered architecture incorporating information source, service and client layer.
3.1 Information Source Layer

The information source layer consists of the set of repositories, P2P networks and Web 2.0 applications to be integrated in our infrastructure. Each of these information sources must conform to the interfaces described above for each service provided.

3.1.1 Web 2.0 Tools

There is a large list of web applications (like YouTube and Flickr) that publish their API, allowing user access not only through the web browser but also through other applications. Whereas traditional learning object repositories typically provide (partially standardised) resource descriptions (e.g., Dublin Core or LOM), Web 2.0 tools do not adhere to these standards but rather use specific metadata for the knowledge resources they deal with. Our wrappers not only perform an interface translation but also a metadata mapping. In this way, we...
connect YouTube and Flickr to a federated search engine that retrieves the results from both platforms in a uniform format.

### 3.1.2 Centralised Repositories
A number of repositories, like ARIADNE, Merlot, Lornet, etc. have been heavily in use over the last years, therefore holding now an enormous amount of knowledge resources. Some of these repositories already support integration via SQI and federated search. As an example, ARIADNE repository offers a web service based API through which learning objects and their metadata can be transparently modified. As a part of this work, the Simple Publishing Interface was integrated into the ARIADNE repository.

### 3.1.3 P2P Networks
Although repositories contain a large amount of knowledge resources, many valuable knowledge resources are stored on user desktops. P2P technologies provide a common framework for sharing these resources. Apart of the work done in order to connect Edutella P2P network with other systems via SQI we have now fully integrated the LionShare P2P network in our infrastructure. There exists an SQI to LionShare gateway (Erickson, 20016), which allows users to treat the LionShare P2P network collectively as just another large distributed learning object repository.

### 3.2 Service Layer
The service layer provides search and publishing services as well as services for information source, session and user management. This layer contains the services that will be accessed and composed by client applications. Services in this layer may also make use of other services from the same layer.

Source code for the service layer can be found at:

http://tencompetence.cvs.sourceforge.net/tencompetence/wp5/KrsmS
ervices/

### 3.2.1 Information Source Management Service
This service encapsulates selection and management of various information sources. In our implementation, we use a UDDI registry which contains a uniform description of the services that an
information source supports, metadata schemas and connection details. Based on this information, a client application can select information sources for further interaction.

### 3.2.3 User Management Service

The user management is provided as a web service over a MySQL user database. Depending on the role, the user is able to modify her user profile (e.g., address, password, etc.) or to create and delete other users. This service is also used in order to enable for personalised single sign on for all the components from the information source layer.

### 3.2.4 Resource Rating Service

The resource rating service allows for recommendation of suitable knowledge resources to a particular user. Our implementation uses Taste\(^{12}\), which is a flexible collaborative filtering engine, for creation of recommendations based on the ratings assigned by the users. Taste takes users preferences for items and returns estimated preferences for other items. It provides a rich set of components from which customised recommender systems can be constructed by using a selection of different algorithms.

### 3.3 Client Layer

The client layer provides a graphical user interface that connects the services and the user. We envision the use of both stand-alone and web-based applications. The former enables managing and sharing local knowledge resources via the local LionShare peer. The latter has the advantage that it may be accessed from anywhere without the need for any software installation. We support both scenarios though our implementation currently focuses on the first one.

\(^{12}\) [http://taste.sourceforge.net](http://taste.sourceforge.net)
4 Integrated Repositories and Components

4.1 Federated Search Engine

Our infrastructure uses a federated search engine that leverages standards at two levels. At the top level, this federated search engine exposes search functionality as an SQI target. Thus, from a standards perspective, sending a query to a federation of repositories is not different than sending a query to a single repository. Moreover, this engine leverages SQI at the level of the repositories it federates to. As we want to lower the threshold of adding a new information source to the federation, adding a new information source requires no more than registering its capabilities into our information source management service. The federated search engine consults this service for the new information sources dynamically.

4.2 LionShare and Limewire for P2P

LionShare is a P2P network which primary goal is to facilitate the distribution of localized content found on the personal computers of educators and researchers not having an easy way to publish this content in popular learning object repositories or preferring to keep control over their resources. The LionShare P2P network now comes with a SQI to LionShare gateway. This SQI target allows users to treat the LionShare’s P2P network collectively as a distributed learning object repository.

Our first KRSM client implementation was based on the use of the LionShare P2P client. After performing several tests we encounter that the full functionality of the LionShare client can only be achieved after installing two additional heavy servers – one for authentication, and another one for backing up all the shared files from the peers, which made the system very heavy and difficult to be connected with the TENCompetence infrastructure. In order to significantly improve performance of the client by only using its peer-to-peer functionality we decided to replace LionShare with Limewire (an open source P2P software based on the Gnutella protocol and implemented via Limewire communication libraries), and to integrate repositories search from LionShare into the new client as Limewire does not natively support useful repository search features from LionShare network. As a result, we use Limewire P2P client (for all peer-to-peer communications and services). We implemented repositories search functionality, based on the LionShare ECL gateway on top of the Limewire client, and
supplemented the search through the SQI-based search service, described previously. In such way we still are able to provide all useful functionalities available in the LionShare network, complementing them with additional peer-to-peer functionalities available from the Limewire network.

### 4.3 ARIADNE

ARIADNE infrastructure is a distributed library of digital, reusable educational components called the Knowledge Pool System (KPS) now actively used in both academic and corporate contexts. This ARIADNE KPS is a Learning Object Repository that offers a web service based API through which learning objects and their metadata can be transparently modified. The goal of having such an API is to enable loosely coupled integrations in third party applications such as VLE's, authoring tools or federated search engines.

### 4.4 Taste

**Taste overview**

Taste is a flexible, fast collaborative filtering engine for Java. The engine takes users preferences for items and returns estimated preferences for other items. Taste provides a rich set of components from which you can construct a customized recommender system from a selection of algorithms. Taste is designed to be enterprise-ready; it’s designed for performance, scalability and flexibility. It supports a standard EJB interface for J2EE-based applications, but Taste is not just for Java; it can be run as an external server which exposes recommendation logic to your application via web services and http.

Top-level packages define the Taste interfaces to these key abstractions:

- **DataModel**: is the interface to information about user preferences, an implementation might draw from any source. Taste provides `MySQLJDBCDataModel` to access preference data from a database and `FileDataModel` to access preference data from a text file. Along with DataModel, Taste uses the `User`, `Item` and `Preference` abstractions to represent the users, items and preferences for those items in the recommendation engine.

- **PreferenceTransform**: alters preference values in some way, possibly normalizing or exaggerating them. These may be attached to a DataModel.
UserCorrelation and ItemCorrelation: a UserCorrelation defines a notion of similarity between two Users; these are attached to a Neighborhood implementation. ItemCorrelations are analogous, but find similarity between Items.

UserNeighborhood: in a user-based recommender, recommendations are produced by finding a “neighborhood” of similar users near a given user. A UserNeighborhood defines a means of determining that neighborhood, for example nearest 10 users.

Recommender: is the core abstraction in Taste; given a DataModel, it can produce recommendations. Applications will most likely use the GenericUserBasedRecommender implementation or GenericItemBasedRecommender, possibly decorated by CachingRecommender.

Subpackages of comp.planetj.taste.impl hold implementations of these interfaces. These are the pieces from which you will build your own recommendation engine.

Taste implementation requires Java/J2SE 5.0 or later and optionally:
- Apache Ant 1.5 or later, if you want to build from source or build examples.
- A servlet container, such as Jakarta Tomcat, for Taste web applications.
- An EJB 2.x container for Taste EJB.
- MySQL 4.x or later database if you use MySQLJDBCDataModel implementation.

Description
Our implementation is exposed as a web application through which it is possible access recommendations via simple http requests. It has been used the servlet contained on Taste, RecommenderServlet, modified for our requirements. This servlet, based on preferences expressed by users, return which items might be interested to an user specified.

Implementation
The servlet use MySQLJDBCDataModel as datamodel and the recommendations are based on user correlation.

In order to use the MySQLJDBCDataModel class is necessary supply access to a database with a “taste_preferences” table with the following schema:
Table 1 - “Taste_preferences” table

<table>
<thead>
<tr>
<th>user_id</th>
<th>item_id</th>
<th>preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxx</td>
<td>123</td>
<td>0.9</td>
</tr>
<tr>
<td>yyy</td>
<td>456</td>
<td>1.1</td>
</tr>
<tr>
<td>zzz</td>
<td>123</td>
<td>0.3</td>
</tr>
<tr>
<td>xxx</td>
<td>789</td>
<td>0.5</td>
</tr>
</tbody>
</table>

User_id and item_id must have a type compatible with the Java String type, and preference must have a type compatible with the Java double type.

The names of table and columns can be changed, in this case, when create the datamodel we must specify the names used as parameters of constructor:

```java
MySQLJDBCDataModel model = new
MySQLJDBCDataModel(datasource, nametable, useridColumn, resourceidColumn, preferenceColumn);
```

As servlet container has been used Tomcat 5.5.20 and for define the datasource, that is tell to Tomcat which is the source of our data, is necessary:

- install JDBC driver fot MySQL.
- modify the web application deployment descriptor (/WEB_INF/web.xml) to declare the JNDI name under which you will look up preconfigured data source.
- configure Tomcat’s resource factory modifying the /META-INF/context.xml file of our application web.

Finally we can create datasource in the application:

```java
Context initCtx = new InitialContext();
Context envCtx = (Context) initCtx.lookup("java:comp/env");
DataSource ds = (DataSource)envCtx.lookup("jdbc/nameJNDI");
```

and upload it in the datamodel for calculate recommendations.
Use of the Taste service

The servlet is available to http://taste.giuntilabs.com/taste/servlet. For use it, you need specify two parameters: userID and howMany. userID is identifier of the user we want know items that might be interested; howMany is the number of many item you want on result. Is also possible specificare a third boolean parameter, debug, it’s optional and it’s used for obtain an useful human-readable output. An example of request to the servlet is below:

http://taste.giuntilabs.com/taste/servlet?userID=123&howMany=5[?debug=true]

The servlet response is a simple preference-item id list which could be consumed by a client application.

<table>
<thead>
<tr>
<th>Input parameters</th>
<th>String userID, String howMany</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output parameters</td>
<td>String</td>
</tr>
<tr>
<td>Dataformat(s)</td>
<td></td>
</tr>
<tr>
<td>Preconditions</td>
<td>user Id must exists</td>
</tr>
<tr>
<td>Postconditions</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Errors</th>
<th>Description</th>
<th>Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Not Found</td>
<td>Is userId doesn’t exists or parameter is missing or empty the servlet generate an error</td>
<td>The servlet thrown a ServletException(&quot;User Not Found&quot;)</td>
</tr>
<tr>
<td>Parse Error</td>
<td>howMany attribute is missing or not a string representing an intenger</td>
<td>The servlet thrown a NumberFormatException</td>
</tr>
</tbody>
</table>
4.5 Flickr

Flickr Server implements a Proxy between the federation search in SQI and the native API of Flickr. The interface between the two API was made using a free implementation of the Flickr API (flickrj), by the other hand, to access to the flickr services a developer ID is needed. To request one in necessary to create a flick account and a key in http://www.flickr.com/services/api/keys /A full list of the features of the API is available in http://www.flickr.com/services/api/.

At this moment a running version of Flickr server is running in http://flickr.altransdb.com/sqiFlickrServer/services where are available all the methods implemented in the Server side.

For the description of the methods/parameters etc. refer to SQI/SPI/SCOI specification documentation as previously linked in section 2.

4.6 YouTube

Youtube Server implements a Proxy between the federation search in SQI and the native API of Youtube (http://www.youtube.com/dev).

For Youtube services there does not exists a java implementation of the Youtube API, At this moment an small implementation of the API in java was developed for this project implementing the methods relative to the 'Video Viewing'(see below) to implement the SQI methods. To access the API a developer ID is required. To request one is necessary to have a Youtube account and request a key in http://www.youtube.com/my_profile_dev However, the methods provided by Youtube are only seven:

User Information
"youtube.users.get_profile
"youtube.users.list_favorite_videos
"youtube.users.list_friends
Video Viewing
"youtube.videos.get_details
Video Viewing
"youtube.videos.list_by_tag (with paging)"
"youtube.videos.list_by_user"
"youtube.videos.list_featured"

At this moment a running version of Youtube server is running in http://youtube.altransdb.com/sqiYoutubeServer/services where are available all the methods implemented in the Server side.

For the description of the methods refer to SQI/SPI/SCOI specification documentation as previously linked in section 2.

### 4.7 eXact Lobster

eXact Lobster is the online LCMS environment for distributed content authoring, sharing and reviewing activities. eXact Lobster embodies all collaborative workflow features such as content management, version control, sharing and peer reviewing and is the Digital Repository where authors can store, share and retrieve resources, Learning Objects and courses. Authors can access eXact Lobster via web browser to edit or assemble content through a user-friendly interface. No client application or additional plug-in is required for online content authoring.

The main features of eXact Lobster are:

- Online Learning Objects editing, packaging and indexing
- Collaborative workflow management
- Shared workspaces with task and milestones scheduling
- Personal workspace for each author
- Version control with update notification and version propagation
- Centralized and distributed Application
- Profile for multiple portal configuration
- Localization Engine for rapid export translation-re-import of Learning Objects
- Map and Taxonomy based content classification
- Integration with 3rd party Open Source or Vendor Solutions

The goal of this task is to integrate eXact Lobster to the TENCompetence Federation. To achieve this goal we need to develop the requested web services. At the end of the integration what we have to expose as web service is:
• Target Service (SQI)
• Session Service (SQI)
• Publishing Service (SPI)
• Download Service (SCOI)

Actually we have implemented Target Service and Session Service.

The other services are under development.

The Session Service supports anonymous and authenticate services.

The Target Service supports:

• VSQL as query language
• LOM as response type
• Synchronous query mode.

For the description of the methods refer to SQI/SPI/SCOI specification documentation as previously reported in section §2.

4.8 eXact Packager search plugin

eXact Packager is the module for creating, indexing and packaging content.

By using eXact Packager it is possible to design interactive instructional content and develop the learning content elements (Learning Objects) forming the structure of a course.

Thanks to the separation of the design phase from the delivery phase, Learning Objects can be readily adapted to a variety of learning requirements (PC, print, mobile device) by simply versioning the same Learning Objects in different ways.

When integrated with eXact Lobster digital repository it gives support for collaborative authoring workflow including functionalities such as content management, publishing and retrieving.

The main features of eXact Packager are:

• SCORM (1.2 and 2004) content authoring
• Templates, Wizards and WYSIWYG for rapid interactive content development
- Learning Objects and Assessments templating technology
- Open Learning Object Modeling framework (easy-to-create templates)
- Standard and proprietary course packaging formats
- Indexing and classification of resources, Learning Objects and courses
- Plug-ins for legacy content repurposing
- Availability in online and/or offline authoring solutions.

eXact Packager allows to develop plugin for external search and retrieving. This gives the author the possibilities to search resources, potentially, from anywhere in the web. For this reason we have developed a plugin to access TENCompetence federation the we are implementing in the WP5.

eXact Packager is based on Microsoft technologies, for this reason the plugins was developed using Microsoft .NET 2.0 framework. This is a good example of interoperability between different platforms.

The integration between a commercial product like eXact Packager and Open Initiative like TENCompetence or ARIADNE is another important target we have reached with this component.

The development process was divided in two steps.

In the first phase we have developed a plugin able to interact with Repositories exposing SQI interfaces.
Figure 3 - eXact Packager screenshot: search on Lobster and ARIADNE repositories

Actually the plugin is able to easily integrate, modifying a simple configuration file, any repository exposing SQI interface with VSQL language, and supporting LOM as result type. At the moment it is possible to integrate only repositories that support synchronous queries. From the interface it is possible to select one or more repositories and make parallel queries.

In the second step we have integrated the possibilities to access a federation based on RMI interfaces and SQI. Through this section of the interface is possible to query RMI to find available repositories in the federation. After this, it is possible to send a query to all the federation or only to some of the repositories. Even for the Federations it is possible to add others Federations that are using the same interfaces SQI and RMI simply by adding their information in the configuration file.
Configuration file

The configuration file have root tag named repositories and it can contains repositories description and federation description.

The tag “repository” is used to describe a data source, it contains information like:
- Name. The name of the repository
- Description. The description of the repository.
- SqiSessionService, SqiTargetService. The entry point of the SQI web services
- Querylanguage. the query language used by the repository
- responseType. The type of the response
- anonymous, logdata. Login information
- querymode. State if synchronous or asynchronous queries are supported.

The tag “federation” is very similar to repository but have some additional information
- sqiRepositoryService. RMI web service for federated repositories
- sqiPublishingService. Entry point for publishing web service,
- sqiDownloadService. Entry point for download web service.
5 Next steps

During the mid-March 2007 WP5 focus meeting held in Sestri Levante, and whose main findings will be reported in part V, the joint work and analysis with WP3 representatives brought to the definition of the collaboration between the WPs in terms of consistent design and development.

In particular, it was decided that WP5 would have followed the TENC client communication architecture, where all current communications are client-server based, and all clients are supposed to store and use all the needed information from the databases available from the TENC server(s). As a result, the KRSM system will be significantly modified, since all pure P2P features will be removed.

As major impact from this change, all the information representing the links (i.e. connections) between the KRs will be stored in the central DB. Such information will be then used for ‘discovering’ KRs (rather than simply searching for), and for semantic analysis purposes. WP3 will provide other WPs with the APIs for accessing this central DB.

Moreover, publishing and sharing resources will become one common function supported by saving the resources on a server, and providing more sophisticated access and filtering mechanisms.
6 Conclusions

Nowadays, learners require a more flexible approach for lifelong learning, making use of the large amount of knowledge resources available as well as collaborating with other users. We identified the requirements a knowledge resource sharing and management infrastructure should address in order to integrate heterogeneous information sources and provide homogeneous search and publishing services. KRSM middleware provides the basic backbone for a flexible and powerful environment in which learners can find, publish and use knowledge resources wherever they are stored. We make use of simple but flexible and powerful interfaces to provide services of our open source service oriented architecture.

Although the results presented here are promising, there is still a lot of work to do. Further refinement and validation of the interfaces, as well as exploration of the new services must be performed. Furthermore all our work needs to be validated as part of the overall infrastructure for the competence development and lifelong learning.
Annex A

1 GUIS ..................................................................................................................26
    1.1 EXPECTED FINAL GUI (FUNCTIONAL VIEW) .................................................26
2 KRSM MANUALS ...............................................................................................35
    2.1 INSTALLATION MANUAL .............................................................................35
    2.2 USER MANUAL ............................................................................................47
1 GUIs

1.1 Expected final GUI (functional view)

List of functionalities
- Create a Resource
- Search for / retrieve a resource
- Store a resource
- Pack / build knowledge resources
- Delete a resource
- Modify access and permission rights associated to a resource (Sharing)
- Rate a resource
- Preview a resource
- Preview resource’s metadata
- Log the KRMS system (This should be provided in the proof-of-concept system)
- Set the working mode
- Add a repository
- Set the storage capacity
- Perform back-up

Additional functionalities
- Recommending systems for users’ preferences via collaborative filtering like ‘Taste’
- Sharing images via ‘Flickr’
- Sharing videos via ‘YouTube’
- Sharing bookmarks via ‘Technorati’
- Creating / sharing text documents and spreadsheets via ‘Google Docs & Spreadsheets’

Remark: related scenarios and use cases are NOT available yet !!!
## Keys

<table>
<thead>
<tr>
<th>What is related to current use cases</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>What is missing in the use cases but is a natural addition to them</td>
<td></td>
</tr>
<tr>
<td>What is completely missing in the scenarios and use cases, but it was foreseen for the 2nd cycle system (i.e. the ‘Collaborative’ section)</td>
<td></td>
</tr>
<tr>
<td>What has been added for consistency sake with WP2 diagram</td>
<td></td>
</tr>
<tr>
<td>What is related to WP3 but has to be implemented in WP5 for the proof-of-concept (e.g. subscription system)</td>
<td></td>
</tr>
<tr>
<td>What is strictly related to WP3 (e.g. Options) but is functional to understand the overall picture</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4 - Functional GUI keys**
Figure 5 - Functional GUI: Create KRs
Open / Search

<table>
<thead>
<tr>
<th>Most recently used:</th>
<th>Open...</th>
<th>Search by...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Item 1</td>
<td>Browse all repositories</td>
<td>by Name</td>
</tr>
<tr>
<td>2. Item 2</td>
<td>Ariadne</td>
<td>by Format</td>
</tr>
<tr>
<td>3. Item 3</td>
<td>Lobster</td>
<td>by Content Pattern</td>
</tr>
<tr>
<td>Most rated:</td>
<td></td>
<td>by Search Engine</td>
</tr>
<tr>
<td>1. Item a</td>
<td>DSpace</td>
<td>SET THE MAXIMUM # OF RETURNED RESULTS</td>
</tr>
<tr>
<td>2. Item b</td>
<td></td>
<td>SET THE MAXIMUM # OF RESULTS TO BE DISPLAYED</td>
</tr>
<tr>
<td>3. Item c</td>
<td></td>
<td>PREVIEW ?</td>
</tr>
</tbody>
</table>

**Figure 6 - Functional GUI: Open / Search KR**
Store

Figure 7 - Functional GUI: Store KRs
Collaborative tools

<table>
<thead>
<tr>
<th>Resources in use:</th>
<th>Images:</th>
<th>Social Bookmarking:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Select all resources</td>
<td>• FLICKR</td>
<td>• Technorati</td>
</tr>
<tr>
<td>• Item 1</td>
<td>• ...</td>
<td>• ...</td>
</tr>
<tr>
<td>• Item 2</td>
<td>• ...</td>
<td>• ...</td>
</tr>
<tr>
<td>• Item 3</td>
<td>• ...</td>
<td>• ...</td>
</tr>
<tr>
<td>• Item 4</td>
<td>• ...</td>
<td>• ...</td>
</tr>
<tr>
<td>• Item 5</td>
<td>• ...</td>
<td>• ...</td>
</tr>
<tr>
<td>• Item 6</td>
<td>• ...</td>
<td>• ...</td>
</tr>
<tr>
<td>• Item 7</td>
<td>• Google Docs</td>
<td>• EXPORT</td>
</tr>
<tr>
<td>• Item 8</td>
<td>• ...</td>
<td>• IMPORT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Videos:</th>
<th>Others:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• YouTube</td>
<td>• ...</td>
</tr>
</tbody>
</table>

Figure 8 - Functional GUI: Collaborative tools
### Resources in use:

<table>
<thead>
<tr>
<th>Item</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item 1</td>
<td>✓</td>
</tr>
<tr>
<td>Item 2</td>
<td>✗</td>
</tr>
<tr>
<td>Item 3</td>
<td>✓</td>
</tr>
<tr>
<td>Item 4</td>
<td>✗</td>
</tr>
<tr>
<td>Item 5</td>
<td>✗</td>
</tr>
<tr>
<td>Item 6</td>
<td>✗</td>
</tr>
</tbody>
</table>

### Rating:

Enter the quality rate:

(1 - poorest, to 5 - highest):

Select the quality rate:

- Poor
- Low
- Medium
- Good
- High

### Access / Permission Rights:

Set the Permission rights:

- Read
- Write
- Execute
- Share
- None
- All

Set the Access rights:

Select the User group

Enter the User name:

### Figure 9 - Functional GUI: Properties
Administration

<table>
<thead>
<tr>
<th>Working mode:</th>
<th>Tools:</th>
<th>Settings:</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>BACK-UP</td>
<td>Storage Capacity:</td>
</tr>
<tr>
<td>Wireless</td>
<td>Add a repository</td>
<td>Select the repository</td>
</tr>
<tr>
<td>Mobile</td>
<td>Remove a repository</td>
<td>MAX_STORAGE_THRESHOLD</td>
</tr>
<tr>
<td>Dial-up</td>
<td></td>
<td>Max value available</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>Enter the desired value (GB)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>PWD</th>
<th>LOGIN</th>
<th>LOGOUT</th>
</tr>
</thead>
</table>

Figure 10 - Functional GUI: Administration
Options

<table>
<thead>
<tr>
<th>Character:</th>
<th>Menu bar:</th>
<th>Others:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font</td>
<td>Text &amp; Icons</td>
<td>Text only</td>
</tr>
<tr>
<td>Size</td>
<td>Text only</td>
<td>Icon only</td>
</tr>
<tr>
<td>Colour</td>
<td>Icon only</td>
<td>Icon size</td>
</tr>
</tbody>
</table>

**Page format:**

- Foreground colour: BLUE
- Background colour: WHITE
- Window size: 1100 × 800
- Skin: TEN
- Status bar: CHECKED

**Figure 11 - Functional GUI: Options**
2 KRSM manuals

2.1 Installation manual
Setup Eclipse IDE and Run KRSM project

1. Install Java JDK version 5.x or 6.x on the machine if there is no one. The java can be downloaded from http://java.sun.com/javase/downloads/index.jsp


3. The following screenshot may appear when Eclipse is loaded.
4. From file menu choose “New\Project”
5. The next step is to select “CVS\Project from CVS”
6. Fill up the fields:
   Host: 62.44.100.138
   Repository path: /TENC
   User name: wp5guest
   Password: guest
7. On next screen fill Use specific module name
8. Type the project name:
9. Click “Finish” button. At this moment Eclipse must start downloading the source.
10. Close the welcome screen and you see the “Java perspective” (the perspective shows the project and java classes).

11. Please pay attention on “Problems” tab (bottom of the screen). Only some warning must appear:

12. Before to run the project extract the .limewire.rar file in “C:\Documents and Settings\username” folder. The username is the user name of the currently logged windows user. This archive contains settings of P2P search. Without them the project is not capable to make this search.
13. The Project can be run as the root of the project tree is selected and “run” menu is chosen (The green button with arrow on tool bar).

14. Select “Java application” and double click on it.
15. Type the name of configuration: KRSM
16. After that choose “Search” button. Type Main and select: Main – com.limegroup.gnutella.gui:
17. On argument tab type : VM arguments: 
   -Djava.library.path=./lib/native/windows
   -Dkms.l3s.username=mechthild
   -Dkms.l3s.password=mechthild

   and Working directory: ${workspace_loc:limewire/gui}

18. Finally press “Run” button to run it.
19. When the program is run for first time it asks about share folder. Choose one and you should see the following screen.
2.2 User manual
KRSM Client main functionality

Main Functionality of each of the Main Tabs of the client:
1. Search – through the Search Tab. Use “All Types” as a search type. There are three different ways to search for:
   - P2PSearch – searching in the Gnutella network of peers (using Gnutella protocol) – this search is implemented by the Limewire client searching functionality
   - Repositories – federated search in the repositories inside the “Lionshare” network, which we implemented inside the Limewire client.
   - KRSM – federated SQI-based search, which we developed and implemented

   All these three different ways of search can be combined together, as well as both Repositories and KRSM search can be further fine-grained by including or excluding specific repositories from the search (for such
fine-graining the “instrument” icon next to “Repositories” or “KRSM” tabs has to be pushed, and desired repositories from the list appeared on the screen can be selected or de-selected). Several searches can be started simultaneously. The results from each search appeared in a new tabbed window. Results from P2P search can be downloaded, while results from federated searches can be launched in a Web browser. Each result from the search can be Rated (how useful and needed was this resource according to the user).

2. Monitor - Limewire-specific Tab, used for controlling and monitoring of the connectivity and network performance

3. Connections – Shows who is connected to the client, or who the client is trying to connect to
4. Library – it is used mainly for sharing objects and making them available to other peers.

For each selected Shared Object we can Launch it, Describe it (Making metadata description of the Object), Delete it, and Publish it.

5. Console – shows details of the Gnutella network events and other communication messages

6. Logging – for making logs of all of the KRSM activities performed

7. Collaborative – This tab will be used for using inside the KRSM client the new Web 2.0 tools and resources for social collaboration like Flickr, YouTube, Wiki’s, Blogs, Podcasting and other (still in development).

Main Functionality of each of the Menu Options of the client:

1. File – used for connection and disconnection to the P2P network, for closing the client, and for implementing the TASTE function. This function is used for the recommendation from the KRSM system of
other resources to the user, based on the current user’s preferences, calculated by the list of chosen/selected objects for searching, downloading, launching, etc. This function is performed by linking KRSM client to the TASTE system, but still it is not fully operational, as it should be linked with the User Management module, which is not available yet.

2. View -> Show/Hide is used for the customization of the GUI, by choosing which Tabs to be visible.

3. Navigation -> for providing access to the same functions as through the Main Tabs

4. Tools -> Options – it is used for choosing preferences how to work with the client, for assigning specific folders and objects for Sharing and Downloading, for assigning parameters specific to the Gnutella protocol, etc.

5. Help -> will be used for providing User Help and Documentation
At the bottom of the screen the client shows various parameters describing quality of the network connection, number and quantity of the file transfers, etc.