Intera Deliverable D2.3

Portal Construction

Project reference number	e-content EDC-22076 INTERA / 27924
Project acronym	*Intera*
Project full title	Integrated European language data Repository Area
Project contact points	Khalid Choukri Evaluations & Language Resources Distribution Agency S.A. 55-57 Rue Brillat Savarin, 75013 Paris, France Phone: +33 1 43 13 33 33 , Fax: +33 1 43 13 33 30 Email: choukri@elda.fr
Project web site	http://www.elda.fr/intera
EC project officer	Philippe Gelin
Document title	Portal Construction
Deliverable ID	D2.3-1 First Version
Document type	Report
Dissemination level	PP
Contractual date of delivery	M22
Actual date of delivery	01.11.2004
Status & version	final version
Work package & task ID	WP2
Work package, task & deliverable responsible	MPI
Number of pages	13 except appendixes
Author(s) & affiliation(s)	Daan Broeder, Peter Wittenburg, MPI
Additional contributor(s)	
Keywords	Meta data, Language Resources
Abstract	
Additional notes & remarks	

Portal Construction

1.1 Introduction

In D2.2 we have reported about the construction of the integrated resource domain and all the tools that are necessary. All software is Open Source and well documented and the IMDI Schema is well-documented and available as well. In this deliverable the task is to describe how interested institutions can set up an IMDI metadata portal.

Metadata creation is of greatest interest for the future. It is a major part of eContent in the Semantic Web era, since it supports resource discovery in an extremely fast growing domain of electronic documents in the Internet. Due to these trends it is well-understood that general information retrieval techniques have their limitations when locating special resources. Metadata descriptions are one of the promising ways to escape from the information overflow dead end scenario. Browsable and searchable metadata may help the professional people to easily find what they need. Further, correct metadata will be a strong pillar in the Semantic Web scenario where Web Services will be applied for automatic data discovery and exploitation.

Therefore, it is important to know how to set up a metadata portal. Currently, in the domain of language resources we have two de facto standards: (1) The IMDI framework was developed mainly by European institutions to support a web-based, browsable and searchable domain for the professional to easily locate language resources. Therefore, the set is more detailed and was designed based on the needs of the discipline. (2) The OLAC framework was designed as an extension of the well-known Dublin Core set to the language resource domain. It is less detailed, but it is suitable for bringing together the resource descriptions from different metadata domains.

The two frameworks are complementary in several aspects, therefore it is agreed between the two initiatives that gateways are supported. Within ISO TC37/SC4 there is a working item about metadata that has the task to derive a unified set of descriptors and integrate them into an open data category registry. One of the authors is convenor of this activity. The ISO involvement will guarantee that the metadata work that will become increasingly important is embedded in long-term standardization efforts. We see an opening to a more flexible and modular way of metadata descriptions in the future, which will be based on the availability of open and therefore re-usable definitions of metadata categories.

From the INTERA project and other initiatives we know that metadata creation is a hard process. The main reasons are

- Often the resource repositories are in a bad state. Since metadata requires explicitness and correctness, people have to do quite extensive work to get their data well-organized.
- Often people don't know anymore the details of the resource production, so it is hard if not impossible to fill in all the descriptors.
- Institutions with repositories don't always have good people who can write scripts to automatically create correct metadata descriptions. Manual work often is too expensive.
- Not all creators are yet understanding the great importance of metadata, since they still work in an egocentric area where only the short-term individual or project interests count. Re-usage by others is often not a primary focus.

In the following two chapters we first briefly introduce the major components of the IMDI framework and then describe which steps are necessary to become an IMDI portal.

2. Open IMDI Framework

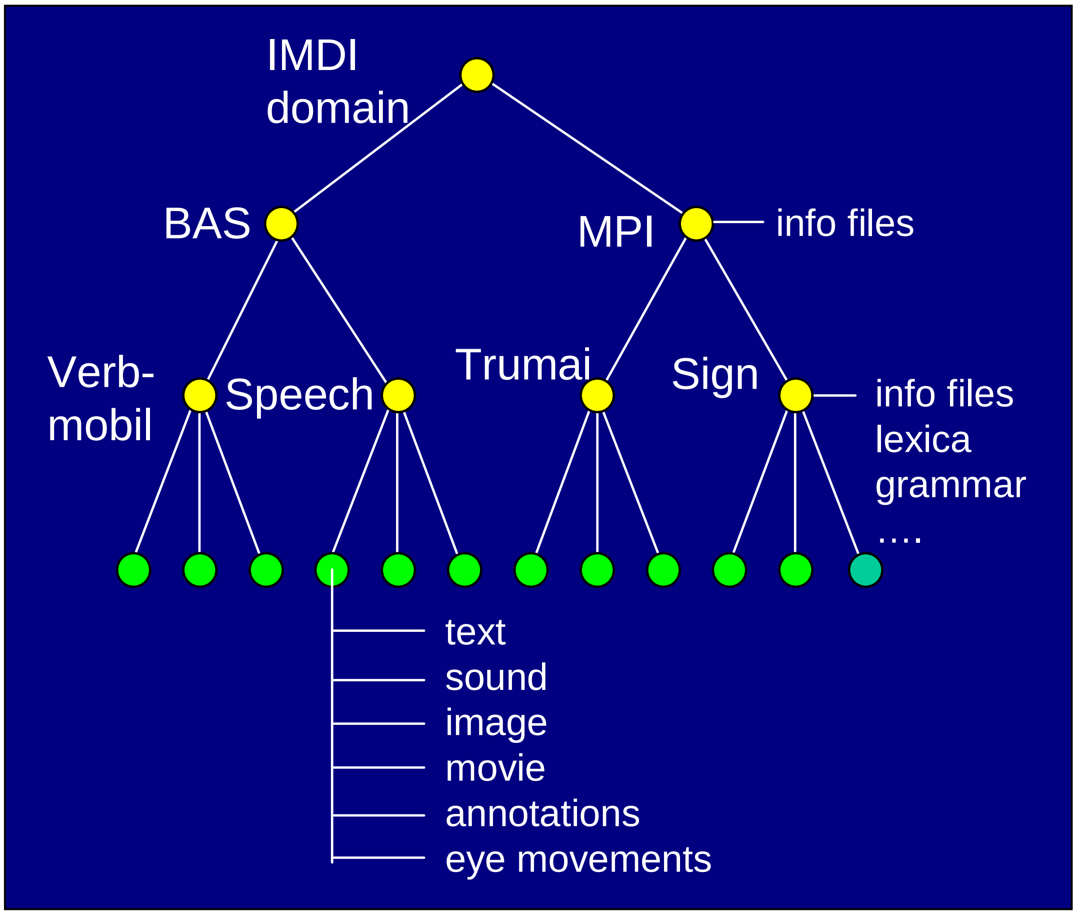
The IMDI framework is based on open metadata descriptions defined by an open and well-documented XML schema (http://www.mpi.nl/IMDI/Schema/IMDI_3.0.xsd). Also the controlled vocabularies that are used for some elements are defined by open XML schemas. This is important since everyone can create IMDI metadata descriptions, create a validating program and build his own metadata creation, navigation and exploration software.

Further, the IMDI domain is a structured one that allows to create linked metadata domains suitable for browsing. The metadata descriptions can reside on any server, they must have a URL and have to be accessible via web-server.

At the moment there are three different IMDI file types:

a) The IMDI Session file or resource bundle. This file contains metadata about one ore more language resources.
b) The IMDI Corpus file. Defines the browsable interlinked structure of one or more corpora.
c) The IMDI Catalogue metadata file. A metadata description of a corpus at the level of a catalogue. It does not describe individual resources but provides information on a corpus as a whole.

This is indicated in the following figure (corpus files are in yellow, session files are in green):



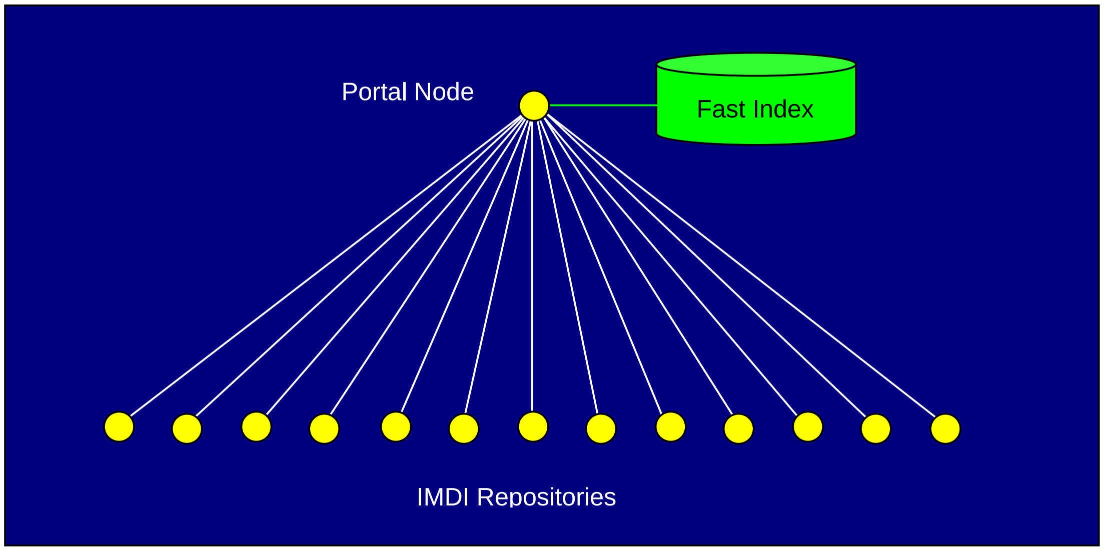
The figure shows a typical IMDI configuration. Two language resource repositories (MPI Nijmegen and BAS Munich) create a linked domain. Each of the sub-domains is responsible for its own metadata domain. A simple registration process at a top node (portal) allows to integrate the two domains to create one browsable and searchable domain. The way the individual domains organize their repository is completely up to them – it should be logical to help browsing users. Any node is an abstraction of the underlying resources, therefore at any node supplementary files can be added. If the nodes represents a language such as “TRUMAI” it makes sense to associate useful information about that language at the language rout node.

This session files contain elaborate metadata about possible multiple language resources that belong together because they are part of the same linguistic event or performance. This definition is often adapted for the specific needs of a particular corpus or research domain. The Session or resource bundle file may also link to the resource files (Transcriptions, Media files etc.) but this is not required. If these links are provided it will facilitate easy exploitation of the language resources themselves in combination with inspection of the metadata. This type of file can be created with the IMDI Editor tool [2].

The corpus files contain a short description of part of a corpus that belongs together because the resources in that part have something (common metadata features) in common. Corpus files may be linked to other corpus files or to IMDI Session files. The linked Corpus and Session file form a tree describing a particular corpus. In the same way the trees describing different corpora may be linked together presenting a description of all the corpora available at an research institution or research field. This file can also be created with the IMDI Editor.

The catalogue files are special files that describe a whole corpus on a very abstract level.

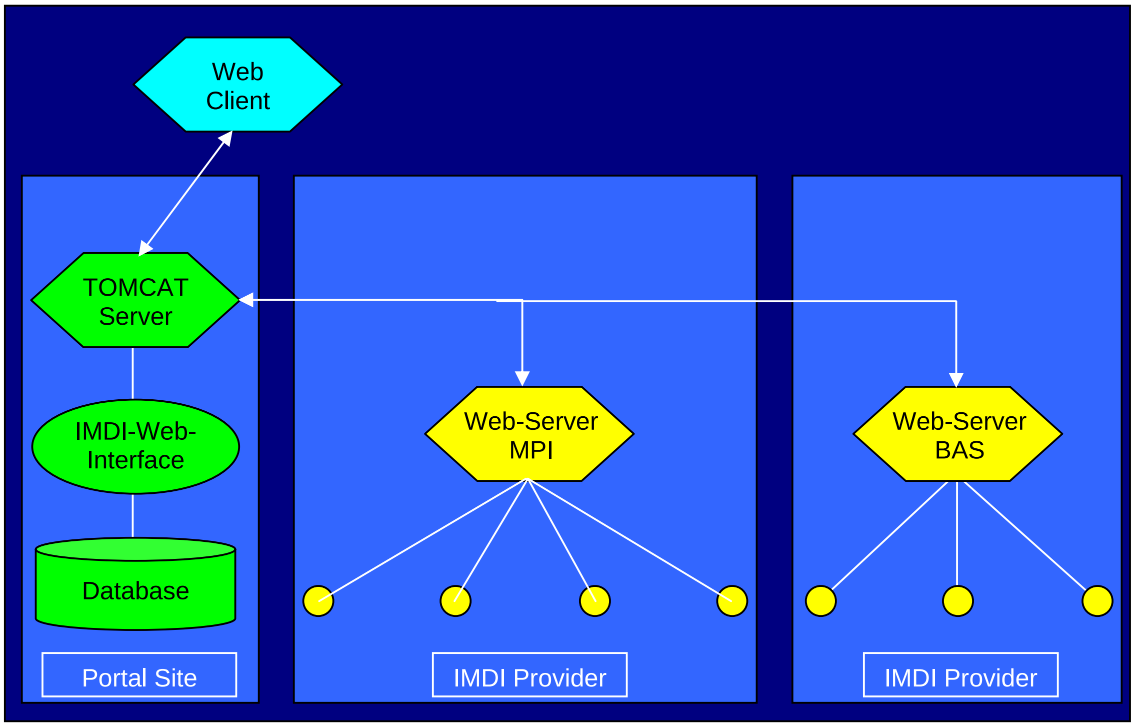
Yet there are no standard browsers that support visualizing such a domain of linked and schema-supported XML files. Therefore, the MPI built the IMDI BC Browser that allows browsing and searching in such domains. It has a search module that allows to carry out complex and detailed searches. The browser was described in D2.2. However, for searching one has to create fast indexes, processing at search time would lead to very slow response times. The institutions that run portal nodes have to harvest all IMDI metadata descriptions from the registered repositories at regular time intervals and create fast indexes. In general, these indexes are contained in a relational database. The IMDI BC Browser has such functionality as well. It can create an index for all registered sub-domains.



There is no limitation with respect to the number of portals. Every site can act as IMDI portal by simply registering those repositories it is interested in and creating the appropriate indexes to support searching.

Since many people hesitate to download software but prefer to use standard components, the IMDI framework also supports to work with normal browsers. Browsing and searching is possible with for example Internet Explorer or Mozilla by creating on the fly HTML presentations from the XML IMDI files. To provide this functionality an IMDI metadata portal has to run a servlet engine such as TOMCAT and a Java program called “IMDI-Web-Interface” since it only operates with virtual addresses of the IMDI resources.

A web-site will point to the web-application (IMDI-Web-Interface) that is hosted at the portal site in a Tomcat application container. The application will contact a Postgres database that as a translation between all virtual and physical addresses of the supported IMDI domain. For every IMDI file referred too it will retrieve the physical address (URL), get it by contacting the appropriate web-server that can be any server in the world and transform the information. Finally, the web-browser at the client side can present the necessary IMDI information on the clients screen. The portal has to regularly scan all IMDI files that belong to the domain and create a database that contains the mapping between physical and virtual paths.



It is evident that the two solutions are different:

- using the IMDI BC Browser as exploration tool is technologically straightforward, but it requires to download a special tool that is capable to understand the IMDI files and does not require a servlet engine and DB software
- using normal web-browsers require a special setup at the portal site and a metadata harvesting step to generate the mapping database

In doing so the IMDI domain is a very flexible and autonomous one. Every site can decide whether it wants to act as an IMDI portal, which repositories it wants to connect to and which services it wants to support.

The framework we are speaking about will undergo revisions during the coming years, since new technologies and tools will be launched. Yet we hope that standard browsers will be able at a certain moment to operate in schema-based XML domains. If this occurs people do not have to either download an extra tool or service providers don’t have to install servlets that carry out additional operations. The IMDI technology provider, the MPI, will continue to improve its IMDI framework and facilitate the setup where possible.

3. Portal Creation

3.1 Current Access Points

At the moment there are two access points:

1) The MPI for psycholinguistics with the top node of the IMDI domain and includes IMDI metadata for the MPI’s own corpora, the corpora from the international DOBES project and metadata created within the INTERA project [3].

http://www.mpi.nl/world/corpora/IMDI/metadata/IMDI.imdi

2) An access point hosted by the university of Lund that points to IMDI metadata created for several corpora during the ECHO project. http://echo.sol.lu.se/BC/Corpusstructure/IMDI.imdi

The IMDI Browser tool is pointing per default to the MPI's top IMDI node, however, that is of course easily configurable.

3.2 Setting up a IMDI metadata DB server

As was explained above, there is no extra effort for setting up a portal when using the IMDI BC Browser. There has to be a web-site with some text and a link to the IMDI site allowing to download the Open Source IMDI BC Browser. Further, there could be a note extracted from the comprehensive browser manual of how to define a new root node. For supporting searching some additional work has to be carried out.

Because crawling the IMDI domain for suitable metadata is a slow process, it is advisable to harvest IMDI metadata and storing it in a relational and fast DB. The IMDI framework (the IMDI tools and metadata file formats) supplies a DB server that can be queried from a search tool that is included with the IMDI BC Browser.

When there are multiple IMDI DB servers available it is important to know which DB to query for information about a specific corpus. Therefore an attribute in the top node corpus file can point to a specific DB server, thus indicating the DB server to query for metadata of a specific corpus a specific site or the whole IMDI domain.

Setting up an IMDI DB server and how to link the corpus node file to the server is described in appendix A.

3.3 Setting up an HTML-based IMDI service

As was explained above also some extra work has to be carried out to support the capability to browse with the help of normal web-browsers in an IMDI domain. The steps are described in appendix B.

References

[1], [2] All IMDI tools are available from http://www.mpi.nl/IMDI/tools

[3] INTERA reference at ELDA

Appendix A

How to setup an IMDI metadata DB server

The IMDI metadata DB server is based on the HSQLDB software [http://hsqldb.sourceforge.net/]. The code for running the daemon is contained in "hsqldb.jar" and it is started from a script file "startIMDIdb.sh". In this startup script information is specified where to find the java installation and the DB files. The DB server uses a number of DB files that hold the RDB tables with metadata. One of these files also contains configuration data that can be edited. The configuration information consists amongst others of the different corpus root URL's that should be changed when the corpus changes (physical) location.

The DB files themselves are created from the IMDI browser just as the special DB files for local use are created (see the manual), i.e. except the above mentioned libraries no additional software has to be installed. With this difference that first some configuration information has to be specified. This is done by selecting "Options" - "Preferences" - "DB creation". The resulting panel offers the following options:

1. SeparateKeyValueTables: set this to false if the DB files concern a single corpus that systematically uses the same set of user defined keys. Default is true.
2. Include Descriptions: if true the information in the "free text" description elements is included in the DB. Default is false.
3. Corpus Root: The part of the URL path seen by the IMDI browser that all IMDI files of the corpora have in common. This information is used to make the path names in the DB files relative.
4. Remote Corpus Root: The URL path seen by a www-browser that all IMDI files of the corpus have in common.
5. UNIX Local Corpus Root: The URL path seen by a (UNIX) NFS client that all IMDI files of the corpus have in common.
6. Windows Local Corpus Root: The URL path seen by a (Win) SMB client that all IMDI files of the corpus have in common.

This information concerning root paths is needed if the DB files are to be used by both remote as well as local clients.

Once the DB files have been created, the DB daemon can be started with the start up script., With the currently use version (1.7.1) of the HSQL DB, the daemon can only handle one DB.

Finally the corpus top node should be made to point to the DB daemon by modifying the topnode "SearchService" and "CorpusStructureService" attributes. These should both have the value “jdbc:hsqldb:hsql://hostname/” where hostname is the name of the host running the HSQL DB server.

(N.B. This reference format will change with the next version of the HSQLDB software)

The command file for starting the IMDI DB server on Linux

```
#!/bin/sh
#This is the startup script for the HSQLDB IMDI DB server
#modify DBDIR to the directory where the DB files can be found
#modify JAVA_HOME to specify the java installation dir
#The hsqldb jar should be in the DBDIR directory
#
DBDIR=/data/corpora/IMDI/DB
JAVA_HOME=/usr/lib/java/jre
cd $DBDIR
```

```
$JAVA_HOME/bin/java -mx200M -classpath hsqldb__V1.7.1.jar org.hsqldb.Server -
database imdi 1>&2 > HSQLDB.log
```

The software for setting up the search feature can be downloaded from ??

Appendix B

How to setup the HTML Option

The Software is available via http://www.mpi.nl/tools/. It depends on an running tomcat-5 installation (see http://jakarta.apache.org) and ant (http://ant.apache.org/).

Installation:

1. Unpack the sources of the IMDI web-interface.
2. Make sure your ant installation is capable to provide the advanced catalina-tasks (do so by copying the file server/lib/catalina-ant.jar of your tomcat installation in the libdir of your ant installation)
3. Edit the file build.xml in the root directory of the IMDI web-interface sources. There are four lines marked with the comment "UPDATE THIS!". These four parameters are the root-directory of your tomcat installation, the URL of the tomcat-manager-application, the username and password for a tomcat-user of the manager-role (a user, who is allowed to deploy new applications, see conf/tomcat-users.xml of the root-directory of your tomcat installation).
4. Edit the file src/virt_phys_path_Bean.java. There are two variables to set describing the root IMDI file (the topnode, parent of all your IMDI files) and its name: rootPhysicalPath and rootVirtualPath. These variables are also marked with the comment "UPDATE THIS!".
5. Start the tomcat server. Go to the root directory of the IMDI web-interface and execute the command "ant install".
6. Open a web-browser and and type in your tomcat base URL and the suffix "/iwi" (for example: http://127.0.0.1:8080/iwi). The IMDI web-interface should open.

When you prefer another j2ee-container rather than tomcat, you can execute the command "ant dist" and install the war-file in the dist directory into your j2ee-server.