Project ref. no.	IST-1999-10647
Project title	ISLE MetaData Initiative

Deliverable status	Public
Contractual date of delivery	January 2002
Actual date of delivery	13.9.2002
Deliverable number	D10.3
Deliverable title	EAGLES / ISLE Metadata Showcase
Type	Report
Status & version	Final Report Version 6
Number of pages	216
WP contributing to the deliverable	WP10
WP / Task responsible	Peter Wittenburg
Author(s)	Daan Broeder, Freddy Offenga, Don Willems, Peter Wittenburg, Uli Heid, Andreas Vögele, Andrei Popescu-Belis
EC Project Officer	Brian Macklin
Keywords	Metadata, language resources, language engineering
Abstract (for dissemination)	The ISLE Metadata Initiative creates a showcase that is based on the IMDI Metadata Element set. The showcase includes an editor tool also supporting constraints and controlled vocabularies, a browser that allows the user to browse in a domain of metadata descriptions, a search tool that allows to search for specific resources and a showcase with metadata descriptions from a number of institutions working in a distributed scenario. Several search methods have been studied for further optimization.

ISLE Metadata Initiative (IMDI)

Final Report[1]

Deliverable 10.3

IMDI Showcase

September 2002

IMDI[2] Technical Report

Max-Planck-Institute for Psycholinguistics
Nijmegen, NL
Institut für Maschinelle Sprachverarbeitung
Stuttgart, D
ISSCO - University of Geneva
Geneva, CH

[1] Since the ISLE Metadata Initiative could not start before September 2000 due to missing subcontracts the final report D10.3 is delivered in September 2002.
[2] For information about the ISLE Metadata Initiative, please, look at the following web-site: www.mpi.nl/ISLE.

Contents

1. Introduction

In report D10.2 the IMDI Metadata Element sets were described including its constraints and controlled vocabularies. It was also described which procedures were used to specify the details of the IMDI sets. In this report we will present the tools that make the structures described in D10.2 work and that allow the user to create and maintain a metadata infrastructure. At this moment the tools do not support the IMDI lexicon metadata set, since it was decided by the Steering Board that with respect to lexica only a first proposal for metadata descriptions should be worked out within the limited time-span of the ISLE project. Within the follow-up INTERA project the support for lexica will be realized, since several lexica will be part of the European language resource infrastructure to be developed.

A special chapter is devoted to searching architectures and implementations, since different approaches were to be tested out for searching in these highly distributed metadata domains. Searching in a distributed metadata domain as in the IMDI case requires optimized mechanisms. Within the IMDI initiative three groups were working on different solutions for searching: (1) The MPI team worked out a search mechanism that is integrated in the IMDI Browser to create a workable environment (see chapter 2.1) and that works in a distributed environment; (2) The IMS team worked on a search architecture using a single whole IMDI corpus that was made available (see chapter 2.2); (3) The ISSCO team also received the whole IMDI corpus to work on an architecture for searching (see chapter 2.3). The advantages and disadvantages of the different solutions are briefly compared in chapter 2.4.

This report will also describe the current showcase corpus that will be developed into a European Language Resource Area in the INTERA project. To come to the showcase corpus a few European institutes were contacted and asked whether they would test out the procedures and apply them to a limited set of their resources. Due to the amount of work involved only small fragments could be described and integrated into the showcase, of course. It is reserved for the INTERA project to extend these fragments and to integrate them into a critical mass of resources.

Chapter 1 describes the IMDI tools developed by the MPI team. Together they form an integrated and operational metadata framework. Chapter 2 describes different search architectures which were tested in the IMDI initiative and where also IMS and ISSCO made contributions. Chapter 3 is devoted to present and describe the showcase corpus.

1. The IMDI Tools

1.1 Introduction

Users need good tools to create and maintain metadata descriptions, to create browsable hierarchies, and to browse and search in this domain of linked metadata descriptions. At the MPI such tools were developed to a professional level. It is intended to integrate further functionality stepwise.

The tools that support the IMDI metadata set and infrastructure are:

- The IMDI BCEditor that is used to create IMDI metadata descriptions and nodes in metadata hierarchies;
- The IMDI BCBrowser that can be used as a viewer for the IMDI metadata descriptions and that allows navigating the universe of linked IMDI metadata descriptions;

- The IMDI Search tool allows the user to specify a query to discover specific resources in the IMDI universe;
- A number of utilities that allow working efficiently with metadata;
- An implementation of the interchange of metadata between the IMDI and the OLAC domain.

All IMDI tools are programmed in Java and Perl for platform independence. The relevant programmes that are part of the IMDI showcase were made available to the user community. These tools can be downloaded from the MPI web site: www.mpi.nl/tools using Webstart technology and academic usage is free. Most of the code will be made available under an Open Source Model in December 2002. It seems to be important to protect the editor nucleus to prevent changes that can lead to incompatibilities within the IMDI domain. With Webstart[3] a simple installation mechanism was chosen to allow even naive users to install the software. A separate web site for the IMDI showcase only was created that, to not confuse users, does not mix with other corpus tool activities of the MPI.
The IMDI Tools are used in some long-term projects and will be maintained after the end of the ISLE project. New versions and new tools will be made available through the MPI web site. In future projects it will be investigated whether the different search architectures can be merged to one combining the advantages of each.

It should also be mentioned that the IMDI project is currently the metadata project with the most complete tool support as far as we know.

[3] Webstart is a product from SUN allowing the user to easily download and launch (JAVA) applications.

1.2 The IMDI BCEditor

The editor allows the user to create IMDI metadata descriptions that adhere to all details of the IMDI definitions: the element set, the constraints defined on some of the elements, the controlled vocabularies defined for other elements and the XML schema defining the generated files.

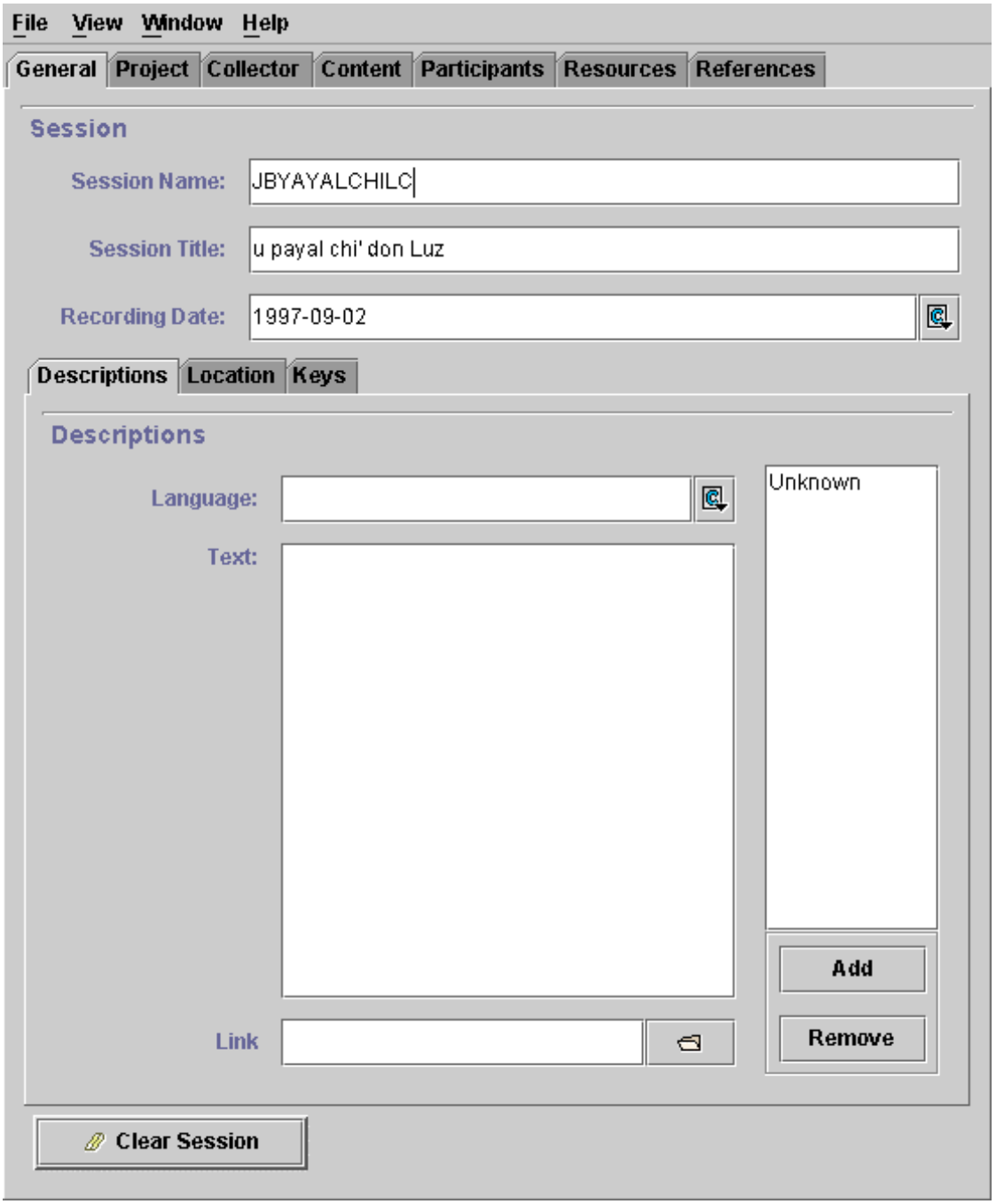


Figure 1

This editor presents all the IMDI metadata elements in a structured GUI to the user. In this document we will briefly introduce the editor. For all details we refer to the manual in appendix 1.

As can be seen in figure 1 the editor has editable fields for all IMDI elements and uses tabs for each logical sub-group of metadata elements. This way that the user can easily switch between them and the user interface remains uncluttered. It has a number of useful and user-friendly features; the most important ones are indicated in the following list.

- It fully supports the actual version of the IMDI schema definitions and creates IMDI compliant XML files. This guarantees that the created metadata descriptions show a high degree of uniformity

- It supports the use of Controlled Vocabularies and user definable keyword/value pairs that the IMDI set allows for user or project specific purposes. The controlled vocabularies are part of the tool distribution but the vocabularies will also reside in open repositories that can be contacted via the web so that updates can be downloaded. Since it must be possible to use the editor as stand-alone tool on a mobile device not connected to the Internet, even vocabularies that reside in open repositories have to be downloaded transparently to the user. Since some of the vocabularies are dynamic and will change over time, it is important that the editor includes update mechanisms that allow synchronizing with new versions when the user wants this.

- For several fields constraints are defined which help the user to easily enter the right formats.

- Some subsets of metadata descriptions such as the biographical data of informants or investigators or project administrative data are often repeated in many sessions. The editor allows the storage of such substructures in separate files for reuse. This will save much time during metadata creation.

- The editor also presents the definitions of the elements if requested to help the user in finding the correct element to be used.

- The creation of links to information files or resources is simplified by offering a directory browser.

When the development of the editor was started, there was the hope that an editor could configure itself on the basis of the XML-Schema or DTD for the IMDI set. This proved impractical due to the fact that although XML-Schemas do include information on the structure of an IMDI metadata description and the constraints on its element values, it does not contain information in how to layout these elements in an acceptable GUI. It was therefore decided to hardwire the connection between IMDI metadata elements and the relevant UI elements of the editor in the editor programme. This requires some reprogramming whenever the IMDI set is modified. An alternative would be to use a general purpose editor such as XMLSpy with a GUI not specifically tailored for IMDI, but it is much to unfriendly for the normal user. While changes in the metadata set occurred frequently at the start of the IMDI project, the set has become more stable now and work on the editor is now more concerned with the refining and perfecting of the UI.

1.3 The IMDI BCBrowser

The IMDI BCBrowser is the central tool for exploiting the IMDI metadata domain. It allows navigation of the universe of linked IMDI metadata descriptions by clicking on corpus links and descending into the corpora to the resources. The browser keeps track of its position in a browsable corpus structure and shows the metadata and human readable descriptions associated with the sub-corpus in focus. It was described elsewhere that the IMDI concept was not just to have an unstructured set of metadata descriptions, but that IMDI provides mechanisms also to manage structured corpora and complex resources. The creation of an integrated browsable IMDI domain is achieved by creating metadata nodes that contain abstractions of underlying information and linking

less abstract descriptions to this node. One of the possible hierarchical structures is called the "canonical structure". It should always be present; it is designed by the resource managers and will also be used for management purposes.

However, several such linked structures may coexist in parallel dependent on the user wishes, i.e. each user can define his own structure which is then most suitable for his purposes. One user may want to define an abstraction level where he splits his corpus on a certain level according to the languages involved. Therefore, this user would define a layer with metadata descriptions that contain the different languages included. This level would then be linked to nodes that only cover resources of the corresponding language. This is depicted in figure 2.

Figure 2

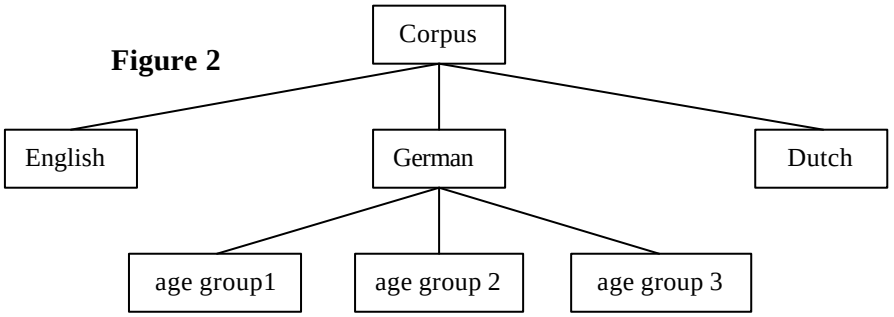


Figure 2 describes a typical example of a hierarchy of linked nodes helping to organize a corpus. At the first level it separates by languages and at the second by age groups. Another user may want to inverse this order.

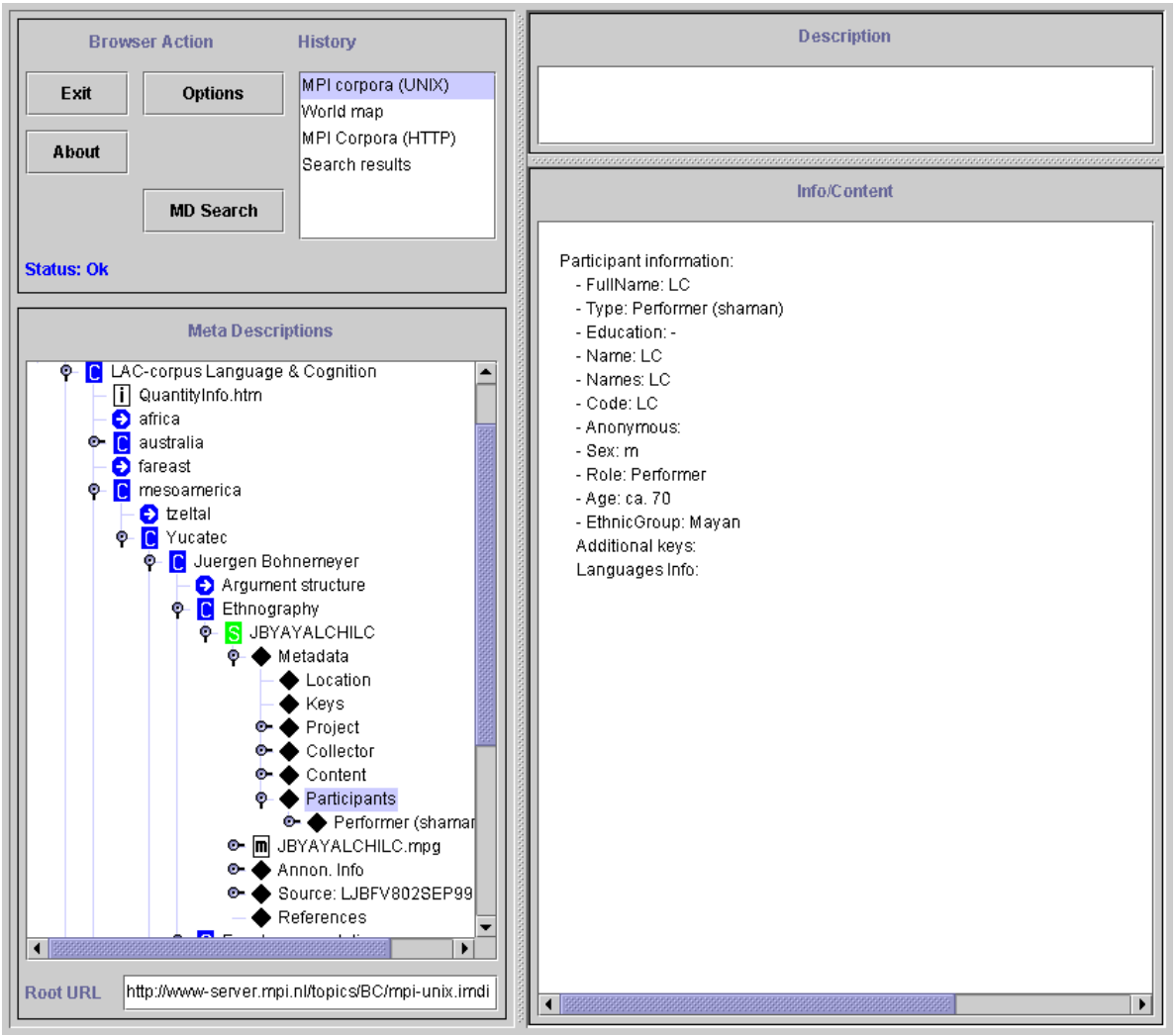


Figure 3

A picture of the browser showing part of the MPI sub-corpus for Yucatan is shown in figure 3. It shows the hierarchy of levels chosen by the corpus creator and gives information about metadata elements associated with a session.

The browser has a number of useful and user-friendly features the most important ones are indicated in the following list.

- The browser can operate in the domain of linked IMDI/XML metadata descriptions stored in the web or on a local file system and that are identified by URLs. At every moment the browser indicates which path the user has chosen so far.

- At all levels of such an organized corpus various types of information suitable for the level of the node in the corpus can be linked to the node. Such information could be lexica, language descriptions, grammar descriptions and many more. The browser is also capable of showing these types of information files that are often provided as extra documentation for corpora if they are in HTML of PDF format (using an external PDF reader). From the HTML pages there may be links back to metadata descriptions making it possible to mix classical HTML browsing with browsing the IMDI corpus universe.

- An interesting application of browsing is a world map that was created as a portal of the showcase corpus. This world map is viewable as an HTML file but has, at the appropriate places, links to metadata descriptions for corpora that correspond to those locations. Currently, corpus entries are marked as bullets in some region. But bullets do not represent projects or areas where a language is spoken etc. More intelligent methods are feasible but have to be worked out by other projects.

- The browser allows the user to select a set of sub-corpora that can be used for example to narrow down the space for a subsequent search, a copy or other useful operations.

- The user can set bookmarks on arbitrary nodes in the IMDI domain to simplify navigation.

- A very important function of the browser is that it offers the user a set of appropriate tools for further analysing resources once they have been made visible. As already mentioned, the IMDI set has some elements that identify the type and format of a resource. The browser uses this information and also the knowledge of other available resources for a session to choose appropriate tools. The mapping between tools and resources is user configurable. Researchers wishing to refine or modify the metadata descriptions may do so by starting the IMDI-BCEditor from the browser, since this is also one of the tools offered. A short manual describing this procedure can be found in Appendix 3.1

- Another tool that can be launched from the browser is “Metadata Search” making it possible to combine metadata browsing and searching.

- It is possible for the user to use prefixes and define them in a file. This allows a corpus manager to create a standard corpus directory configuration where all types off data files have a standard place. The prefixes are the directory paths for the files and the editor automatically adds them to the data file specifications.

It is not the purpose of this document to explain the functionality of the browser in detail. This is described in the manual in Appendix 2.

1.4 The IMDI BCSearch Tool

The search tool is the most recent IMDI development. Its underlying architecture is described in chapter 2.1. It allows the user to specify a query for sessions whose metadata complies with the specified constraints and vocabularies. The UI offers the user an easy way to specify a query compliant with the IMDI element set; the elements value constraints and CV's used. The IMDI BCSearch Tool offers the controlled values in the same way as the editor, i.e. also the search tool gives much help to the user to simplify working with it.

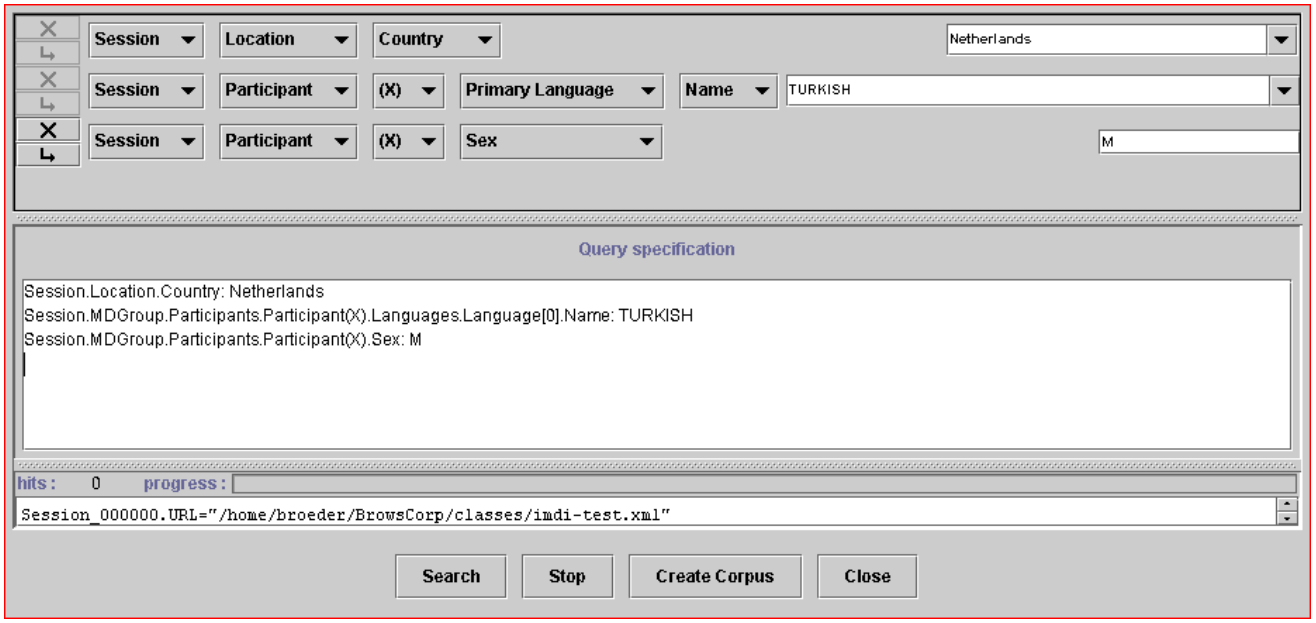


Figure 4

The browser has a number of useful and user-friendly features the more important ones are:

- The search tool operates on IMDI/XML files and it supports the actual IMDI definitions, i.e. the elements and controlled vocabularies are offered in the user interface. For offline usage the controlled vocabularies are cached just as with the other IMDI tools. Descriptions of the controlled vocabularies are available to the user through ToolTip popup menus.

- It allows the user to generate complex queries by combining any number of atomic constraints on elements as is shown in figure 4 in the query specification field.

- The search tool is not dependent on a specific metadata search infrastructure. It can work on local metadata and on distributed remote metadata. If working on local metadata it can use a simple Perl script to search through an optimized metadata index file format and does not require any special database software.

- The current MPI implementation of remote search is that special flat metadata service files have to be created. This file is then searched with help of the same Perl script that is used for local metadata search. The HTTP server that responds to requests from (remote) metadata search clients will initiate the execution of the search script on the service files. Therefore, the IMDI search concept does not depend on a central database, as is the case for example in the OAI concept.

- Results are presented in the form of URL's for the session metadata description files that comply with the query. The user may make these sessions visible in the IMDI-BCBrowser for further inspection or a special corpus can be created containing all these sessions that can be saved for future reference and processing. Together with the possibility to define bookmarks this option is a powerful tool to quickly define virtual sub-corpora for further usage.

The Search Tool is described in the Browser manual. See Appendix 2 for details.

1.5 Utilities

The IMDI team created a number of efficiency tools, i.e. tools that allow the user to easily create and extend metadata descriptions, to list the contents in a more compact form and to convert them from existing descriptions such as spreadsheets. The most important tool for the users creating metadata descriptions is one that allows specifying a whole set of metadata descriptions by simple means and apply some modification or extension to the selected set. In this way repeating information can be added economically. This tool that currently exists only as a separate Perl script has to be integrated to the IMDI tools and support all IMDI definitions such as constraints and controlled vocabularies.

Also many conversion programs were developed that helped convert "legacy" metadata files (usually the proprietary header of a transcription from a specific corpus) to the IMDI format.

An extension of the IMDI-Editor allows the user or manager to create IMDI corpus nodes and link metadata descriptions. This functionality will allow users to create their own private corpus trees next to the canonical one maintained by the corpus managers.

2. Search Solutions

In this chapter different approaches to the metadata-searching problem are discussed. In two internal workshops the partners discussed some search architectures and it was agreed to implement and compare some of them. Searching in metadata domains is not that trivial since it has to consider a number of aspects such as: (1) the number of metadata descriptions can become substantially large and changes continuously. (2) The descriptions can reside on different servers in a distributed fashion. (3) The user has to be able to access his local descriptions independent of the status of his Internet connection. (4) The definition of the metadata set can change over time, i.e. new elements can be integrated and the vocabularies may have changed. (5) In the case of IMDI the user can specify his own keyword/value pairs and wants support for those as well.

The task of building search architectures for the web is not new. In general a central approach is followed, ie. a central service crawls along all linked or registered URLs and builds up a central database. This database then is subject of heavy processing to create efficient and powerful search services. In the contrast to general web pages metadata is structured data and the size of the metadata descriptions is comparatively small, i.e. the interpretation of the data is simple and the total amount of data to be processed is limited. For the IMDI universe the most simple search service would be built on a crawler that inspects all metadata descriptions located under a top node specified for the search request on the fly. Via the HTTP protocol the metadata descriptions could be downloaded record for record and analyzed. The crawler would assure that the most actual descriptions would be used and at the client site nothing has to be prepared and no separate service has to run. Obviously starting such a crawler for every search requests without preprocessing the data is a slow process. Still for small amounts of data and preferably "local" data it can be a simple and workable concept.

The Open Archives Initiative proposed a centralized model. It distinguishes between data providers and service providers. These two interact with the help of a lightweight metadata harvesting protocol,

i.e. the metadata provider has to respond to a set of standardized requests and serve metadata records to the service provider. In general the service provider will ask for the complete set of metadata records (or those that have changed since the last harvesting) and build its own optimized central database including a full copy. All queries will be executed on this central database to achieve a high degree of performance. The big difference with respect to the other extreme is that some service has to run on the data provider's site. Although the OAI protocol is very simple this may form an obstacle for small institutions or even individuals who want to integrate their data into the global metadata domain.

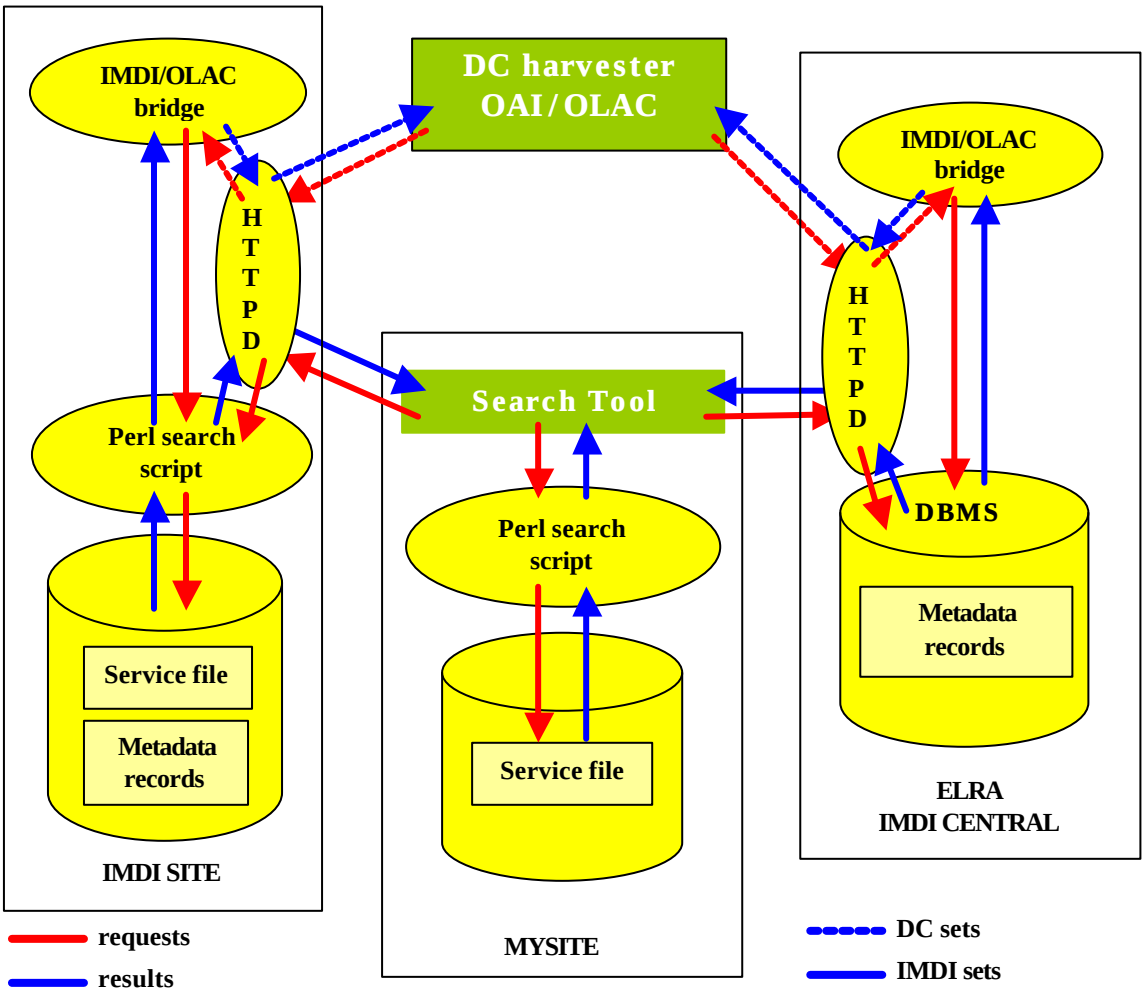
In this report we will discuss three contributions for searching: (1) The IMDI environment currently includes a solution (MPI IMDI search) that does not use a central DBMS although the architecture does not exclude the use of one. (2) Further, a proposal for searching from IMS is included in this report that focuses on aspects when using a central database. (3) Finally, a database-oriented solution for searching is included as well which was worked out at ISSCO. In the realm of follow-up projects it will be considered how IMDI search can be extended such that all criteria listed above are matched and that efficient processing can be offered even in the case of very large sets of records and distributed storage. The three investigated solutions form a good starting point for further implementation work.

2.1 Search Solution: MPI

The architecture implemented by the MPI team was optimized towards two goals assuming that for some time the IMDI metadata number of records stored locally does not become too big: (1) It should be straight-forward to be implemented and (2) even individuals should be motivated to participate. The second requirement let us to developing a system that would function without taking recourse to database management software. The need for database technology for large sites hosting a significant higher number of sessions is not excluded but the process here described has the advantage of being available on installations as small as a home PC.

We first considered a "crawler" like approach were a search programme would look through the corpus tree following the links until all the relevant sessions were found. This method, although simple is much too slow due to the repeated resolving of the sub-corpus links. So it was decided that at each metadata provider site the information of that site is gathered in two flat "service" files, one that describes the structure and the other containing all session descriptions. The Perl script that creates these service files is run regularly by corpus management but can also be launched from the IMDI browser. This is a tool, the IMDI metadata user has installed anyhow, and i.e. no extra work is required. The user should execute the script as soon as he made changes to his private corpus that warrants this update. The top nodes of the managed corpora at a site have links to the service files and when doing local metadata search the Search Tool knows where to find them. When the metadata search is performed remotely it is the site's HTTP daemon that receives a metadata search request, invokes the Perl search script and transmits the results to the Search Tool client. This procedure is mostly more efficient than downloading the service files.

A query is now translated into a Perl script searching through the files. This mechanism seems acceptable for corpora up to 100,000 sessions. Tests with the current IMDI domain at the MPI with about 13.000 session descriptions reveal that in average it costs about 120 seconds to present the results. Modifications of the metadata set do not form a big problem to this approach. New versions of the IMDI metadata sets will result in adapted IMDI tools that either are backwards compatible or come along with appropriate notes. To create the two search service files the user simply has to execute the creation script again, since the script operates independent of the structure and content of the metadata descriptions. The IMDI search concept will include the OLAC domain in so far as it will allow search queries using the OLAC service. This will be integrated later to the IMDI showcase.



The figure shows four possible search modes for the IMDI Search Tool.

1. Local Search

The search tool starts up a Perl search script that searches through an optimal "service file" that contains all the available local metadata. The result of the search is a number of URL's of sessions that comply with the query constraints. The search tool uses these URL's to access the (local) sessions. For a description of the "service file" format see Appendix 3.2

2. Remote Search

The search tool submits a query to the HTTP server of a remote IMDI site (a site storing and offering IMDI records). The remote HTTP server uses the same Perl script to search through its metadata service file and passes the results (URL's of sessions) back to the search tool. The site can only answer questions on resources available on that site. Therefore, if necessary multiple sites can be contacted by the search tool to resolve one single query. The processing can be carried out on several sites in parallel.

3. Central Search

Here we assume that one central site, for instance ELRA, harvests all known IMDI sites for IMDI metadata records. These records are stored in a DBMS. The search tool would only need to contact one site to answer queries on the whole IMDI universe. The exception is of course that local metadata that is not harvested has to be still searched with the "local search" method. The resulting session URL's can point to records stored at other sites then ELRA.

4. OLAC Harvesting

With the existence of the IMDI/OLAC bridge software it becomes possible for IMDI sites to be harvested by OAI/OLAC service providers. Future versions of the IMDI search tools will be able to query those OLAC service providers for the existence of specific (OLAC) metadata records. Since these OLAC records can also be converted versions of original IMDI ones and the original metadata format id is remembered, this can also be used to query for IMDI records. The power of the metadata query is limited though.

Summary

The IMDI metadata search implementation that is presented as part of the IMDI showcase can either be "local" or "remote". Local metadata search entailed a (optimized) text file containing all the metadata of a corpus being searched with a Perl script. Remote metadata search means that a client application (search tool) sends a search query to a remote HTTP server using the HTTP Post protocol. The HTTP server resolves the query for instance by starting a text file search with a Perl script (just as with local search) or by using a (XML) DBMS[4]. In both cases the results are returned as an answer to the original HTTP request. In the original ISLE project the use of an XML DBMS was only experimented with. The interaction between SearchTool client and Search Service (implemented by the HTTP server) is a non-standardized message exchange. The query specification format although close to XPath is non-standard but very easy to generate from the Search Tool and easy to read.

[4] This feature is not implemented yet.

2.2 Searching Solution: IMS

2.2.1 Approaches to Search and Retrieval from IMDI data

This section gives a short overview of different approaches available for the search in and retrieval from IMDI data. We first briefly recall the main properties of IMDI data, as well as a few typical queries (cf. section 2.2.1.1). Thereafter, we discuss three major types of approaches to the problem of searching through XML data, namely different kinds of direct search on XML files (cf. section 2.2.1.2), a few non-commercial XML databases (cf. section 2.2.1.3) and the most recent approach, using XQEngine. This section concludes with a short summary of the main features of each solution, from a comparative point of view.

In the next section, we will describe and illustrate our own approach.

2.2.1.1 The task: Query and Retrieval on IMDI data

In the following we briefly sketch the tasks we are faced with in the development of the IMDISearch tool. We call this the 'IMDISearch task'.

```
<?xml version="1.0"?>
<METATRANSCRIPT>
  <Session>
    <Name>liela24a.1</Name>
    <Title></Title>
    <Date>1984-10-24</Date>
    <MDGroup>
      <Participants>
        <Participant>
          <Type>Subject</Type>
          <FullName>Lavinia</FullName>
          <Languages>
            <Language>
              <Name>Italian</Name>
              <Description>first language</Description>
            </Language>
            <Language>
              <Name>English</Name>
              <Description>second language</Description>
            </Language>
          </Languages>
          <Sex>Female</Sex>
          <Age>21</Age>
          <Keys>
            <Key Name="RELIGION">roman catholic</Key>
            <Key Name="YEAR_OF_BIRTH">1963</Key>
          </Keys>
        </Participant>
      </Participants>
    </MDGroup>
  </Session>
</METATRANSCRIPT>
```

Figure 1: Sample excerpt from an IMDI metadata description in XML

IMDI Documents. IMDI data files are structured XML documents; as the collection of meta-descriptions of resources grows, typically a large number of documents need to be searched; efficiency is an issue: the current test collection of data[5] with IMDI annotations has a size of 13 MB.

Typical W3C approaches to this include:

- XSLT[6], a language for transforming XML documents;
- XPath[7], a language for addressing parts of an XML document;
- XQuery[8], an XML query language similar to SQL.

An example of a small excerpt from an IMDI document is given in figure 1, above.

The Search Problem. The search is directed to the identification of documents via a comparison of elements and attributes. The two types of partial structures, which are in principle, the most relevant ones for IMDISearch are illustrated in figure 2, below.

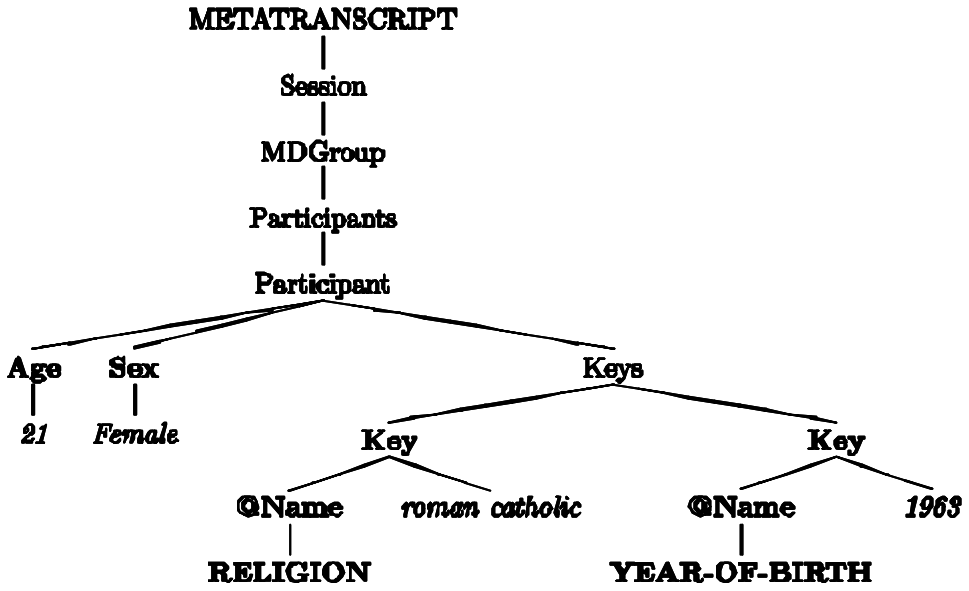


Figure 2: Element and attribute content as terminal leaves of a tree representation of the IMDI XML documents: the main search criteria in the IMDISearch task

Certain descriptors used in the IMDI metadata set, e.g. the indication of the age of a participant, are noted in the same way for different participants, i.e. the same attribute/value-structures can be found embedded at different points in a document. For example, one may wish to find all documents, where both, interviewer and interviewee, are of an age between 20 and 25 years. An example of the representation of such a situation is given below in figure 3.

Otherwise, it should be clear that, for the specific task of IMDISearch, not the full power of, e.g. XML databases are needed.

[5] Which has been made available by MPI Nijmegen: we should like to thank the MPI ISLE WP-10 group, in particular Peter Wittenburg and Daan Broeder, for the permission to use these data in our experiments, and for making the material available to us.
[6] cf. http://www.w3.org/TR/xslt
[7] cf. http://www.w3.org/TR/xpath
[8] cf. http://www.w3.org/TR/xquery

```
<Participants>
  <Participant>
    <Type>Investigator</Type>
    <FullName>Margaret Simonot</FullName>
    <Age>Unknown</Age>
    <Sex>Unknown</Sex>
  </Participant>
  <Participant>
    <Type>Investigator</Type>
    <FullName>Carlo</FullName>
    <Age>Unknown</Age>
    <Sex>Male</Sex>
  </Participant>
  <Participant>
    <Type>Subject</Type>
    <FullName>Lavinia</FullName>
    <Age>21</Age>
    <Sex>Female</Sex>
  </Participant>
</Participants>
```

Figure 3: The same type of partial structures embedded at different paths in the IMDI metadata descriptions: the example of the 'age' value

The number of query expressions to be evaluated in the IMDISearch task is rather limited; the complexity of these queries may however be considerable. Examples of typical queries can be found in section 2.2.3.

2.2.1.2 Direct Search on XML files

There are different approaches to search and retrieval on XML data that are based on a direct search through XML files. These include XSLT processors, specific search programs for XML data and the XML Toolkit, XMLTK. We will in the following discuss each of these briefly and then summarize our experimental findings.

We assume a basic familiarity with the tools: as we indicate the URLS of the tools analyzed, no specific hints about those properties and technical aspects of the tools are given which are not directly relevant for the IMDISearch task.

2.2.1.2.1 XSLT processors

The following features make XSLT processors in principle well suited for the purpose of IMDISearch:

- XSLT allows for the inclusion of XPath expressions;
- XSLT can work on multiple XML documents;
- XSLT can group data;
- XSLT is typically used anyway for output formatting; thus a solution using the same (type of) engine for all tasks would be attractive.

A major problem in the use of XSLT processors for the IMDISearch task is however the fact that typically each query requires a parse of the XML files to be searched. This procedure is too time-consuming for the type of tool we aim at.

To circumvent this problem, one could think of converting the documents to be parsed into an internal representation used by an XSLT processor, e.g. a DOM format. This would however imply that the IMDI search engine would be highly dependent on the format used by one given XSLT processor; this would considerably reduce modularity.

For these reasons, approaches based on XSLT processors have not been followed in the present study.

2.2.1.2.2 Specific XML search programs

There exist a number of XML search programs that also rely on parsing of XML documents (as XSLT processors do), for each query. None of them uses indices.

Even in small-scale tests, the same problem appeared as with XSLT processors: the amount of material to be parsed was too vast for the tools to be efficiently usable. Consequently, no full evaluation of these tools was performed on IMDI data.

Each of these programs has its own query language, although the kinds of queries that can be performed are very similar.

The following are a few examples; all tested tools are based on command line interfaces.

sggrep

This tool is part of the LT XML Toolkit (University of Edinburgh[9]): it is programmed in C, runs under UNIX and WINDOWS and can be usefully applied to querying single XML documents. Its performance is problematic, however, when it comes to the query of a large number of documents.

In example (i) we show the sggrep commands formulated to find the following:

- All sessions from 1983 and 1984,
- With a female participant between age 20 and age 25 of the Roman Catholic religion.

(i)
```
sggrep '.*/Session' '.*/Date' '198[34]-*' -- *.imdi | \
sggrep '.*/Session' '.*/Participant/Sex' Female | \
sggrep '.*/Session' '.*/Participant/Age' '2[01235]' | \
sggrep '.*/Session' '.*/Participant/Keys/Key[Name = "RELIGION"]' \
'roman catholic'
```

sgrep

This tool can also be used under UNIX and WINDOWS; the last beta-version[10] seems not to be supported by the developers any more, which made it difficult to run real tests on it.

fxgrep

This tool is announced[11] as having roughly the same functionalities as sggrep; it can be used on UNIX; as it is written in the functional programming language SML, an SML compiler is needed (which was unfortunately not available to us; thus, no tests could be run.).

2.2.1.2.3 The XML Toolkit XMLTK

The XML Toolkit has been developed at Washington University[12].

[9] URL, as of August 2002: http://www.ltg.ed.ac.uk/software/xml/
[10] URL, as of June 2002: http://www.cs.helsinki.fi/u/jjaakkol/sgrep.html
[11] URL, as of June 2002: http://www.informatik.uni-trier.de/~aberlea/Fxgrep/

General aspects: The XML Toolkit is written in C++ and has mainly been developed to allow the application of a large number of XPath expressions to XML documents. XPath allows to select parts of documents which satisfy certain criteria.

Approach: The XMLTK approach is based on a transformation of XPath expressions into a deterministic finite state automaton. As a consequence, the data throughput is independent of the number of XPath expressions. In addition, an index can be used. In this case, it is possible to leave out irrelevant parts of XML documents, when parsing these: this device increases the efficiency of processing and thus the speed, at query time.

Qualitative evaluation from the IMDISearch viewpoint: The version of XMLTK that has been tested did not support the full range of XPath expressions needed in IMDISearch. Furthermore, XMLTK seems to have been designed for tasks that are almost the opposite of those of IMDISearch:

- XMLTK is very efficient for the use of many (rather simple) XPath expressions applied to few documents. The use of an index then leads to a considerable gain in efficiency. The authors have compared their tool with the Xerces parser, as for throughput: while Xerces has a throughput of 3.9 MB/s, XMLTK has a throughput of 5.4 MB/s on the same computer and conditions. For the use of an index, the authors report a throughput of 27 MB/s.

- The IMDISearch task typically implies to apply few XPath expressions (possibly of some complexity) to a rather large collection of XML files. The use of the deterministic FSA does not really lead, in this situation, to a performance improvement, and even the index does not change the situation.

When the latest version on XMLTK has become public, a fuller evaluation may be appropriate, but due to the large amounts of data to be handled in the IMDISearch task, we still expect performance problems.

2.2.1.2.4 Summary

None of the solutions based on a direct parsing of XML documents which have been sketched above, seem to be convenient for the IMDISearch task. For that reason, we have analysed XML databases, again against the IMDSearch task definition.

2.2.1.3 Using XML databases

With all XML databases, two kinds of tasks are mainly relevant for an evaluation of the applicability of the respective candidate solutions:

- The importation of IMDI data into the database and the creation of indices. As this job needs to be done only in certain regular intervals (e.g. once a week), speed is not critical here. Since incremental updates are possible with XML databases, indeed weekly updates would not be too costly.

  A second updating-related criterion has to do with the ease of index generation: ideally one script should suffice to index all relevant elements and attributes.

- The most important criterion is speed at query time: as mentioned above, in section 2.2.1.1, we need to manage large amounts of XML documents, to be queried by means of a comparatively small number of query expressions.

[12] URL, as of July 2002: http://www.cs.washington.edu/homes/suciu/XMLTK/

Additional major advantages of XML databases, especially of upcoming developments around such databases, are the following:

- XML databases allow to include not only XPath expressions into the queries, but also XQuery. This in turn supports the processing of more complex queries and the combination of data stemming from different documents. In the IMDISearch task, we have focused on the use of 'collections'.
- For XML databases, one homogeneous programming interface is available, in the form of the XML:DB API[13]. On the basis of this API, front ends can be used along with different databases. Front ends for XML databases are only upcoming, but more such tools may be expected in the near future[14].

Two XML databases have been tested in the context of the IMDISearch task: Xindice and eXist. Both will be briefly described in the following.

2.2.1.3.1 Xindice

Xindice[15] is the XML database of the Apache project. It is written in Java and can be used under UNIX and under WINDOWS.

Xindice has a binary internal representation of documents that is not publicly documented so far. Indices can be generated for both XML elements and attributes.

The preparation of indices is however quite time consuming, as one indexing command is needed for each element or group of elements; indexing commands can be Java code or shell scripts.

Since the IMDI data contain the same element (e.g. 'date' at several levels of embedding, within different substructures (e.g. the date of recording, the date of annotation, etc., cf. above, figure 3, retrieval should be able to produce path-like expressions that allow identifying the exact position of the 'date' element in the structures of the analysed text. This seems however not possible with the indices that Xindice creates.

Xindice supports the use of XPath expressions in queries. For example, all sessions from June 1983 could be selected from the collection `/db/imdi` with the following command:

(ii)
```
xindice xpath_query -c /db/imdi \
-s 'i=http://www.mpi.nl/IMDI/Schema/IMDI' \
-q '//i:Session[contains(i:Date, "1983-06")]'
```

The Xindice version tested is much too slow to be considered for use in the IMDISearch context. The use of an index did not lead to any visible improvement in performance.

2.2.1.3.2 eXist

eXist[16] is another XML database programmed in Java. On August 5th, 2002, version 0.8 has been published. Earlier versions had to rely on an SQL database backend (e.g. MySQL), but since version

[13] For details, see http://www.xmldb.org/
[14] Around mid-august 2002, such a front-end has been announced: It is called XMLdbGUI, cf. the following URL, as of August 15th, 2002: http://titanium.dstc.edu.au/xml/xmldbgui/. According to the announcement, it supports the following functions: insert, update and delete documents interactively; add and delete collections of documents; import and export documents from/to the file system; search for data using XPath expressions.
[15] cf. http://xml.apache.org/xindice/
[16] cf. http://exist.sourceforge.net/

0.8, eXist has its own backend, which is more efficient than a relational database. Again, the internal format of representation is not published.

Other than Xindice, eXist has a function to automatically index all elements and attributes at once.

Queries are based on XPath expressions and can be evaluated in acceptable time. IMDISearch queries taken from the list by Andreij POPESCU-BELIS took between 0.5 and 5 seconds on a medium size Sun workstation. The complexity of XPath expressions seems to have a direct influence on the time needed to compute the results.

In example (iii), the session with the name `lsfpa12a.1` is retrieved from the collection `/imdi`.

(iii) `find collection('/imdi')//Session[Name = "lsfpa12a.1"]`

Had eXist 0.8 been available earlier, we might have used it as a backend for our own solution to the IMDISearch problem.

2.2.1.3.3 Other XML databases

Other products[17] (not included in this short survey) and public tools are under development. Due to time reasons, the tool could not be evaluated in the framework of the present study.

Finally, XQEngine, a Java library that needs to be integrated into a tool environment, needs to be mentioned. Developed by Howard KATZ, predecessors of the current XQEngine were commercial products. Since August 13th, 2002, XQEngine is available as free software[18].

As far as can be seen from very first tests, XQEngine would be ideally suited for the IMDISearch problem: it does not only support XPath, but also XQuery.

2.2.1.4 Comparative Summary

In this section, we have sketched the IMDISearch task.

In terms of complexity, the IMDISearch task only requires a subset of the query functions available in XML databases; typically, only elements and attributes serve as query criteria, but these may be arbitrarily embedded in complex structures. The simplest unique identifier for such query criteria is a path expression.

In quantitative terms, the IMDISearch task requires the handling of a rather small amount of query expressions, to be applied to a very large amount of different XML files.

For this latter reason, any approach based on parsing the XML files at query time, i.e. any solution working without an index runs into problems of speed and efficiency.

With XML databases, the (manual) effort needed to create indices is an important aspect in the evaluation (which favours eXist over Xindice). Moreover, the availability of support for XQuery is an advantage (as in XQEngine) of upcoming approaches.

[17] For details, see the following URL, as of August 2002: http://www.rpbourret.com/xml/XMLDatabaseProds.htm
[18] cf. http://www.fatdog.com/

2.2.2 A Perl-based approach to the IMDISearch task

As XQEngine and eXist have only now become available, another approach needed to be followed in our own development work. The following section will describe this approach and compare it with the other IMDI-based solutions.

2.2.2.1 Overview

We propose a search tool for IMDI data based on path-wise indices and on a query component that is geared to the types of queries that play a role in the IMDISearch task.

As indicated earlier, the IMDISearch task focuses on the identification of elements and attributes in the XML documents. Thus, only a part of the functionality of full XML query systems is needed. Due to the amounts of data to be handled, this subset should however be handled efficiently.

The system has the following main components:

- Indexer: A Perl script for creating index files of all elements and attributes contained in the IMDI data collection;
- Search: A Perl script supporting a subset of XPath for the actual query within the index files;
- GUI: a simple input/output GUI linked with the search script via CGI scripts.

Each of the components will in the following be described in more detail.

2.2.2.2. Indexing IMDI data

The indexing tool parses the IMDI data collection and creates index files[19].

During the index creation, the following steps are performed:

All documents are numbered; an index file named `documents` contains a tabulated listing displaying the document names and the respective document number, assigned by the indexing tool.

A sample excerpt from such a list is given in (iv), below.

(iv)
```
0     /esf/liela24a.1.imdi
1     /esf/ladfc16e.1.imdi
```

The tree structure of the IMDI XML documents is mapped onto the file system. An example of this organization is given in the graph in figure 4 below. This figure corresponds to the data in figure 2 above.

[19] On a Sun Enterprise 250 Sever, the tool needed less than two minutes to parse 13 MB of test data and to create the index files.

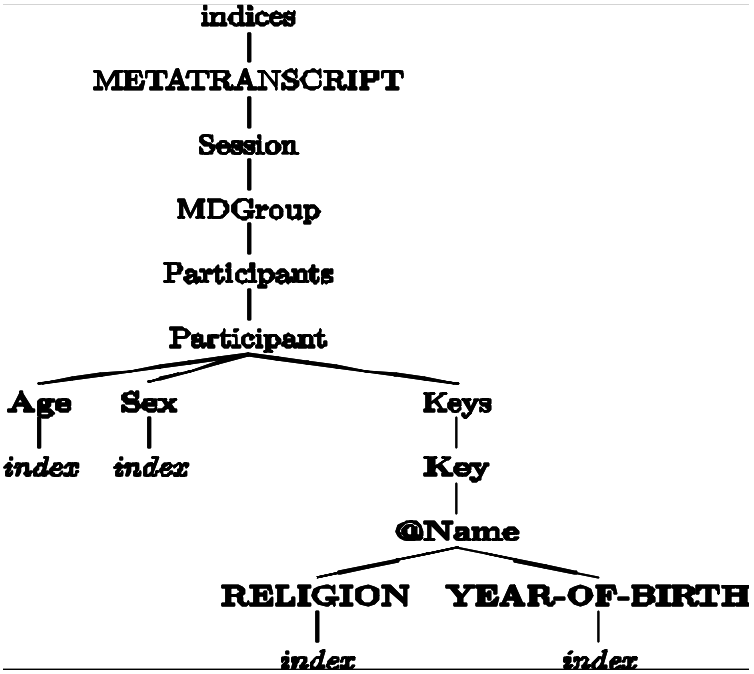


Figure 4: File structure created by the indexer, for the IMDI data files

Each element and each attribute have their own index file; an index file contains the element content, the number assigned to the document in which the element has been found, as well as a path allowing to identify the element. Thus, there is one file per element or attribute to contain the "addresses" of all instances of the corresponding element.

The notation of these data is again that of a tabulated list, see below the excerpt in (v) for the 'age' element; the excerpt covers the examples displayed in figure 3.

Paths are noted using XPointer child sequences[20]. For example, in the following tree the 'date' element can be referred to with the sequence `1/1/3`, for the third child element of the first child element of the root element:

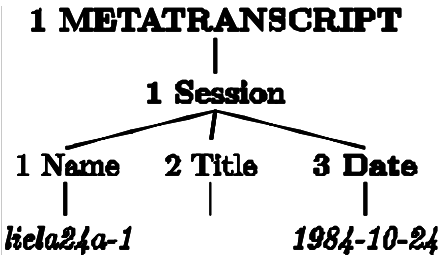


Figure 5: Child element tree

The presence of the path is important: we had noted that elements may be repeated in the IMDI documents (e.g. if several participants are being described), and the paths may be part of the search criterion: for example, the age of two or more participants may be criterial in a given search. Thus the paths are needed to identify the correct substructure and to localize the query results within a given

[20] For XPointer, see the following URL: http://www.w3.org/TR/2001/CR-xptr-20010911/#child-seqs

document.

If element content covers several lines, these lines are merged into one line (by replacing line breaks and tabulation with simple blanks), to simplify the format of the index files and to facilitate search with regular expressions.

```
(v)   Unknown          0     1/1/6/6/2/8
      Unknown          0     1/1/6/6/3/8
      21               0     1/1/6/6/4/8
      Unknown          1     1/1/6/6/2/8
      27               1     1/1/6/6/3/8
```

Index files for attributes are created in the same way: All elements with a common attribute are indexed. An example for 'key' elements with the 'name/religion' attribute is given in (vi):

```
(vi)  roman catholic   0     1/1/6/6/4/12/1
      Islam            1     1/1/6/6/3/12/1
```

2.2.2.3 Searching in IMDI indices

Query language. The search uses a subset of XPath 1.0 with additional regular expression support as its query language: for the IMDISearch task, we only need the search for bare elements, e.g. 'age' elements, and elements with a specific attribute, e.g. 'key' elements with the 'name/religion' attribute.

The result of the search is not the content of the documents, but only the document name and the position of the targeted elements in the document (their "address").

In (vii) we show the query expression[21] formulated to find the following:

- All sessions from 1983 and 1984,
- With a female participant between age 20 and age 25 of the Roman Catholic religion.

```
(vii)  Search.pl '//Session[Date =~ "198[34]-.*"]' \
       '//Participant[Sex = "Female"][Age >= 20][Age <= 25]\
       [Keys/Key[@Name = "RELIGION"] = "roman catholic"]'
```

Query Results and Ranking. Below, in figure 6, we display a few results of the query, ranked according to relevance.

The query can contain several query expressions; the sample query in (vii) contains two expressions, one constraining the date of the session, the other being a coordination of several query expressions.

The ranking takes into account how many of the query expressions contained in a query are met by the candidate documents: if all conditions are met, the ranking procedure proposes the candidate document as a 100-%-match, if one out of two is met, as a 50-%-match etc.

```
<results number="3">
 <document href="file:///esf/liela24a.1.imdi" rank="100">
  <element locn="/*[1]/*[1]/*[3]" name="Date">1984-01-24</element>
```

[21] Note that the queries need not be formulated by the user: the GUI (see below, section 2.2.2.5) produces a query expression from the filled-in form provided by the user.

```
  <element locn="/*[1]/*[1]/*[6]/*[6]/*[4]/*[8]" name="Age">21</element>
  <element locn="/*[1]/*[1]/*[6]/*[6]/*[4]/*[9]" name="Sex">
      Female</element>
  <element locn="/*[1]/*[1]/*[6]/*[6]/*[4]/*[12]/*[1]" name="RELIGION>
      roman catholic</element>
 </document>
 <document href="file:///esf/ladfc14a.1.imdi" rank="50">
  <element locn="/*[1]/*[1]/*[3]" name="Date">1983-01-11</element>
 </document>
 <document href="file:///esf/liela31a.1.imdi" rank="50">
  <element locn="/*[1]/*[1]/*[6]/*[6]/*[4]/*[8]" name="Age">22</element>
  <element locn="/*[1]/*[1]/*[6]/*[6]/*[4]/*[9]" name="Sex">
      Female</element>
  <element locn="/*[1]/*[1]/*[6]/*[6]/*[4]/*[12]/*[1]" name="RELIGION>
      roman catholic</element>
 </document>
</results>
```

Figure 6: Query results

In figure 6, the first candidate result matches all conditions, the second and third only half of them[22].

2.2.2.4 A format for the presentation of query results

We have designed our own output format for query results. It makes use, for the notation of element locations, of XPath.

W3C has come up, in 2001, with a recommendation for XML Fragment Interchange[23], which could be used to output query results. Below, in figure 7 the XML Fragment notation of one of the properties of the search results (Sex/Female) is given. The actual query result is enclosed in `<p:body>` .... `</p:body>`.

The fragment is sufficient for the validation of the XML data; in the XML Fragment Interchange proposal, path information, which is needed to locate elements in a document, is not required. Paths could be included with the attribute `sourcelocn`, but there is no standardization for this device available so far. Furthermore, we do not know of an XSLT processor supporting this syntax.

```
<p:package xmlns:p="http://www.w3.org/2001/02/xml-package"
           xmlns:f="http://www.w3.org/2001/02/xml-fragment"
           xmlns="http://www.mpi.nl/IMDI/Schema/IMDI">
 <f:fcs parentref="file:///esf/liela24a.1.imdi"
        sourcelocn="file:///esf/liela24a.1.imdi#/1/1/6/6/4/9">
  <METATRANSCRIPT>
   <Session>
    <MDGroup>
     <Participants>
      <Participant>
       <f:fragbody/>
      </<Participant>
     </Participants>
    </MDGroup>
```

[22] Results are computed by taking the cross product of each query expression's result set. If the three coordinated conditions given in (vii), i.e. sex, age and religion, were given as separate query expressions, the ranking would be different.

[23] The proposal is at the stage of Candidate Recommendation since February 2001, and we cannot see, as of August 2002, whether the recommendation will indeed be followed. We do not know of any tools supporting XML Fragments. Details about XML Fragments can be found at the following URL: http://www.w3.org/TR/xml-fragment.

```
   </Session>
  </METATRANSCRIPT>
 </f:fcs>

 <p:body>
  <Sex>Female</Sex>
 </p:body>
</p:package>
```

Figure 7: Fragment and fragment body packaged together

Given this state of affairs, we opted for the development of our own output format. An example is given in figure 6 above. This output is then processed by style-sheets to display the results in the GUI.

2.2.2.5 A GUI for IMDISearch

The search script can be used from the command line. But then, the user has to know the internal XML structure of the IMDI files. To keep this structure transparent for the user, a simple GUI has been designed. It allows the user to specify a query by filling in a form. An example of such a form is reproduced in figure 8, below. It should be noted that the GUI does not support grouping or frequency counting of data.

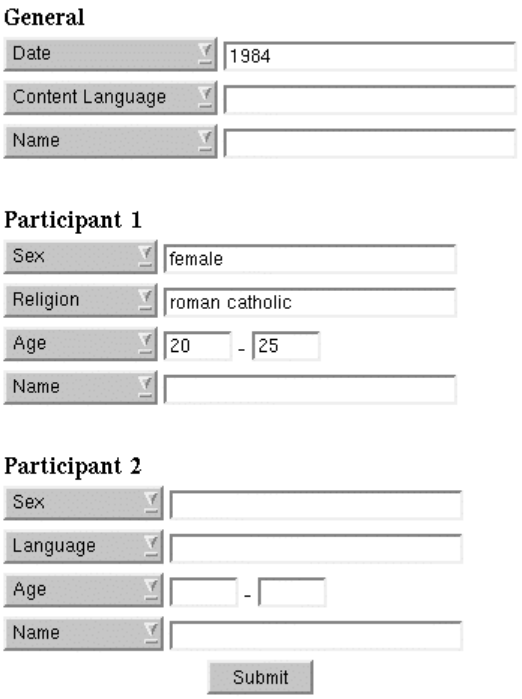


Figure 8: Search form for the entry of queries to the IMDISearch tools

When the user submits a filled-in form, it is parsed and converted to an XPath expression and evaluated by a Perl script. Any other backend could equally well be used with the GUI. The search results are delivered in the internal format discussed above and displayed in figure 9, formatted for the output to the user by means of a style-sheet and then displayed in a web browser.

The results come in a format similar to that of results of well-known search engines:

- A clickable name of the document which satisfies the query conditions;
- A short description of the corpus, taken from the `Description` field of the original IMDI document;
- A highlighted attribute/value description corresponding to the search criterion.

An example is displayed in figure 9.

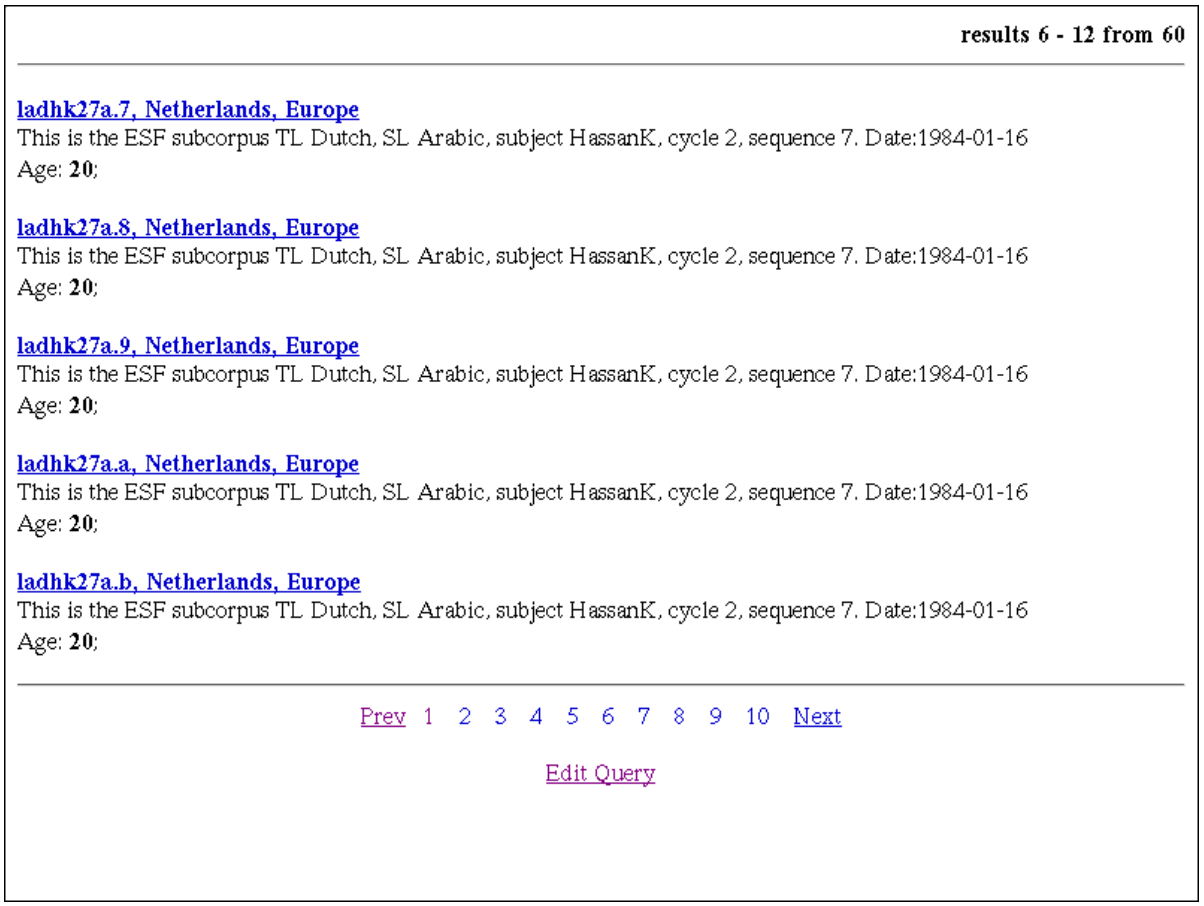
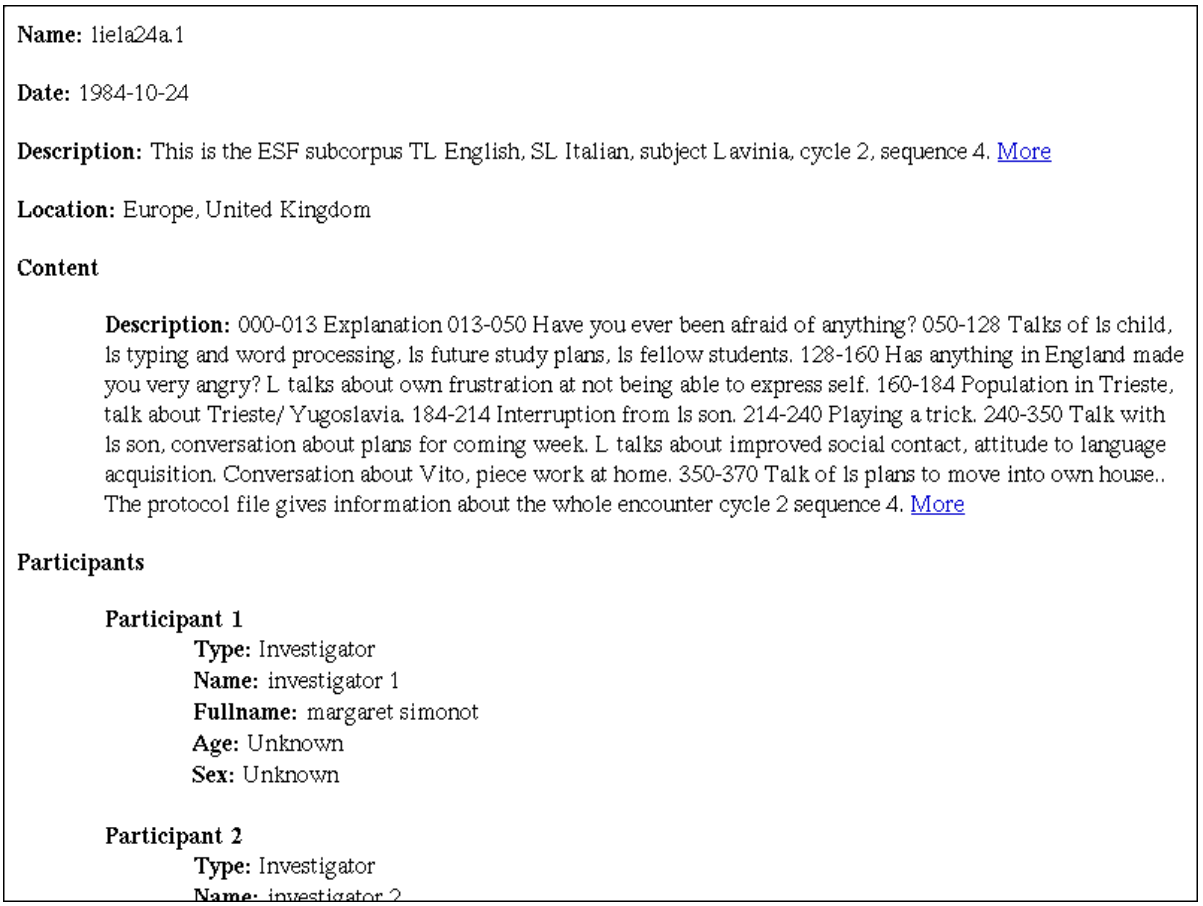


Figure 9: Summary display of results of an IMDISearch query

By clicking on the name of one of the documents listed as query results, the full IMDI description can be displayed. An example is given in figure 10.



Name: liela24a.1

Date: 1984-10-24

Description: This is the ESF subcorpus TL English, SL Italian, subject Lavinia, cycle 2, sequence 4. More

Location: Europe, United Kingdom

Content

Description: 000-013 Explanation 013-050 Have you ever been afraid of anything? 050-128 Talks of ls child, ls typing and word processing, ls future study plans, ls fellow students. 128-160 Has anything in England made you very angry? L talks about own frustration at not being able to express self. 160-184 Population in Trieste, talk about Trieste/ Yugoslavia. 184-214 Interruption from ls son. 214-240 Playing a trick. 240-350 Talk with ls son, conversation about plans for coming week. L talks about improved social contact, attitude to language acquisition. Conversation about Vito, piece work at home. 350-370 Talk of ls plans to move into own house.. The protocol file gives information about the whole encounter cycle 2 sequence 4. More

Participants

Participant 1

Type: Investigator
Name: investigator 1
Fullname: margaret simonot
Age: Unknown
Sex: Unknown

Participant 2

Type: Investigator
Name: investigator 2

Figure 10: Display of full IMDI information in Netscape, started from the query results

A traditional web browser can be used for the IMDISearch task, and users need not get acquainted with another piece of GUI software. In our tests, the IMDISearch tasks were carried out using the Netscape browser. The GUI can be run on any web server that supports CGI[24], Perl and the XSLT processor `xsltproc`[25].

The search form is generated also by means of CGI scripts: this allows for the dynamic generation of the search mask from a given data collection: in this way, always exactly those attribute values are available for query which have been used in the current data collection. This is important, because not all attribute values are subject to standardization and are neither elements of a controlled vocabulary. The user must thus be made aware, for each collection of data, which attribute values are indeed available. This function is a by-product of the index generation for attribute values.

[24] The use of Java applets or JavaScript for the user interface has been discussed, but it was decided to implement a simple prototype that utilizes HTML and CGI only.

[25] For details see the URL: http://xmlsoft.org/XSLT/. `xsltproc` is written in C and can be executed without delay, in contrast to java-based XSLT processors which take much time to load.

2.2.3 Evaluation of the available solutions

We adopted the criteria proposed by ISSCO to compare the different approaches to the IMDISearch. See section 2.3.4 for a comparison of the MPI in-house query language and ISSCO's relational database solution. The criteria are:

1. Programming effort
2. Speed of conversion to query-able format
3. Flexibility with respect to the conversion to metadata format
4. Complexity of storage solution
5. Robustness of storage
6. Access to metadata
7. Expressiveness of query language
8. Query speed
9. Requirements for local querying of metadata

Criteria	IMS Perl-based approach	Direct Search on XML files	XML databases
1.	Initial conversion script (XML to index files); High programming cost to implement the querying mechanism	No programming effort; XML documents are directly searched	XML documents are imported into the database; Moderate programming cost to implement the import scripts
2.	Conversion from XML to index files. Total under 2 minutes for 1300 sessions	No conversion	Slow; the import of 1300 sessions may take up to 3 hours
3.	The query mechanism is the only aspect to modify	The query mechanism is the only aspect to modify	The query mechanism is the only aspect to modify
4.	Text files (nothing to do)	Text files	Database (XML DBMS needed)
5.	File back-up needed	File back-up needed	Database or file back-up needed
6.	Access to local or remote files only	Access to local or remote files only	Distributed access (XML-RPC)
7.	Selections, joins	Selections, joins	Full XPath; upcoming XQuery implementations
8.	19 seconds for 17 queries on a 1300-session database	7 seconds for one-match query on a 1300-session database	2 seconds for one-match query on a 1300-session database
9.	Copy or access to the index files; Copy of the query script; Perl programming language; User-friendly interface needed or knowledge of XPath	Copy or access to the XML files; Copy of the query command; User-friendly interface needed or knowledge of query language	Access to the database; Possibility to launch a db client; User-friendly interface needed or knowledge of XPath and XQuery

2.2.3.1 Text of queries

```
# Give me the session with name "lsfpa12a.1".
# Number of results: 1
# Elapsed time: 0.82 s
perl Search.pl '//Session[Name = "lsfpa12a.1"]'
```

```
# Give me all sessions from June 1983.
# Number of results: 47
# Elapsed time: 0.95 s
perl Search.pl '//Session[Date =~ "1983-06.*"]'

# Give me all the sessions collected by "colin lector".
# Number of results: 0
# Elapsed time: 0.63 s
perl Search.pl '//Session/MDGroup/Collector[Name = "colin lector"]'

# Give me all sessions where the investigator is jorge.
# Number of results: 14
# Elapsed time: 1.93 s
perl Search.pl '//Participant[Type = "Investigator"][FullName = "jorge"]'

# More interesting: Give me all investigator names with the number of
# sessions in which each was involved.
# Number of results: 326
# Elapsed time: 13 s
perl Search.pl '//Participant[Type = "Investigator"][FullName =~ ".*"]' |
\
sed -n "s/.*name=.FullName.>\(.*\)<.*/\1/p" | sort | uniq -c | sort -nr

# Give me me all participants born in 1965.
# Number of results: 57
# Elapsed time: 0.83 s
perl Search.pl '//Participant[Keys/Key[@Name = "YEAR_OF_BIRTH"] = 1965]'

# Give me all married participants born in 1962.
# Number of results: 75
# Elapsed time: 1.11 s
perl Search.pl '//Participant[Keys/Key[@Name= "YEAR_OF_BIRTH"] = 1962]\
[Keys/Key[@Name = "CIVIL_STATUS"] = "married"]'

# Give me all sessions located in Europe.
# Number of results: 1300
# Elapsed time: 1.46 s
perl Search.pl '//Session/MDGroup/Location[Continent = "Europe"]'

# Give me all sessions located in France (without continent specified).
# Number of results: 454
# Elapsed time: 1.02 s
perl Search.pl '//Session/MDGroup/Location[Country = "France"]'

# Give me all sessions located in France (with continent specified).
# Number of results: 454
# Elapsed time: 1.50 s
perl Search.pl '//Session/MDGroup/Location[Continent = "Europe"]\
[Country = "France"]'

# Give me all participants with full name "Clive".
# Number of results: 22
# Elapsed time: 1.21 s
perl Search.pl '//Participant[FullName = "clive"]'

# Give me all male participants.
# Number of results: 989
```

```
# Elapsed time: 1.66 s
perl Search.pl '//Participant[Sex = "Male"]'

# Give me all female participants.
# Number of results: 313
# Elapsed time: 1.26 s
perl Search.pl '//Participant[Sex = "Female"]'

# Give me all combinations of one male and one female participant in
# a session.
# Number of results: 1
# Elapsed time: 2.45 s
perl Search.pl '//Participant[Sex = "Female"]' \
'//Participant[Sex = "Male"]'

# Show me all content languages with their count.
# Number of results: 6
# Elapsed time: 6.10 s
perl Search.pl '//Session/MDGroup/Content/Languages/Language\
[Name =~ ".*"]' | \
sed -n "s/.*name=.Name.>\(.*\)<.*/\1/p" | sort | uniq -c | sort -nr

# Show me the languages of the sessions in which there are two females.
# Number of results: 0
# Elapsed time: 4 s
perl Search.pl '//Participant[Sex = "Female"]' \
'//Participant[Sex = "Female"]' \
'//Session/MDGroup/Content/Languages/Language[Name =~ ".*"]'

# Give me all female participants with the age of 30 to 39.
# Number of results: 202
# Elapsed time: 2.07 s
perl Search.pl '//Participant[Age >= 30][Age <= 39][Sex = "Female"]'
```

2.3 Search Solution: ISSCO

2.3.1 Introduction

2.3.1.1 Generalities

Various tools are needed in order to take full advantage of the ISLE Metadata Set proposed by the ISLE Metadata Initiative (IMDI). For instance, in order to input metadata records, one may use the BC Editor developed by the Max-Planck Institute (MPI, Nijmegen). In order to browse a hierarchy of records, one may use the Browsable Corpus tool of the MPI. However, other useful operations are also required. One of the most important issues is the possibility to search through a set of metadata records for an entry that satisfies a certain number of formal criteria. Another issue is the possibility to network various metadata repositories and query all of them at the same time, using a standard query protocol. We will describe now the two issues (2.3.1.2-2.3.1.3 and 2.3.1.4), then outline a proposed solution to the first issue (2.3.1.5) that is evaluated later on in this report (2.4).

2.3.1.2 The search problem

The metadata records have the logical organization described by the ISLE Metadata Set. Therefore, searching a metadata repository amounts at finding the records that satisfy certain conditions of the

type: “field_F has value_V”, or a combination of such conditions. There are many technological solutions to this problem, and the choice of one of them depends on the actual format of the metadata, on the available hardware and software, etc.

However, since there exists a well-known theoretical model for data storage, search, and retrieval – namely, the *relational database model* – it is worth exploring the possibility of using such a model and the numerous, efficient tools designed for it, in order to search metadata. An initial study by Ulrich Heid *et al.* (IMDI workshop, February 2001) has highlighted, from a theoretical and abstract point of view, some of the challenges of this approach. Their study, conducted at a point when the ISLE metadata format was not yet fixed (though metadata was already available at MPI), was not accompanied by any implementation or evaluation.

To frame the problem a little more, it must be noted that metadata is originally stored as XML files, with XML element names corresponding to metadata field names, and the content of these elements corresponding to the values of the fields. The specification of the XML syntax (i.e. the description of the metadata set) is given using a DTD or an XML Schema. Several search options seem thus available, e.g., use of XML Query, full-text search, etc.

The main problem in converting the XML files to tables of a relational database is that some of the fields are repeatable, and there are even whole structures that are repeatable. Indeed, if none of the fields were repeatable, the metadata structure could be flattened to a certain number of columns, with unique names for each one. Every metadata record would then occupy a line in the (unique) metadata table. But the structures are repeatable: e.g., there can be several participants to a recorded conversation, and each of them must be described in the metadata file. Hence, several tables are needed: e.g., one for participants, one for the ‘sessions’ (i.e. the main storage unit for language resources, as defined by IMDI), etc. This point was already noted by Heid *et al.* (cf. previous section) however, the complexity of the final IMDI metadata set requires the creation of a significant number of tables – and the automation of this process does not seem straightforward.

2.3.1.3 Metadata used in this experiment

The present experiment focuses on the use of existing metadata, and aims at achieving a complete conversion of a given metadata repository to a set of database tables, in order to evaluate the process and its results. The metadata that was available to us describes a series of 1640 recordings of interviews between a number of participants and one or two interviewers[26].

The logical structure of this metadata follows closely the IMDI guidelines, though it covers only a part of the total metadata set. This is quite normal since this information was input well before the IMDI metadata set was fixed.

The exact format of this metadata was not the original XML files (or more exactly the hierarchy of XML files) but a transformation of these files into a more declarative format. The XML trees were scanned by a program, and the values of the elements were extracted and written in a text file using the following format (sample):

```
Session_000000.URL="./data/L2/BCdata/D-DC1B1-RSB.imdi"
Session_000000.Name="d-dc1b1-rsb"
Session_000000.Date="1987-01-26"
Session_000000.MDGroup.Key__000={"discourse type","intention"}
Session_000000.MDGroup.Project.Contact.Name="dfg/prof. XXX"
Session_000000.MDGroup.Content.Languages.Language__000.Name="german"
```

[26] The author would like to thank Peter Wittenburg, Daan Broeder and Freddy Offenga for providing the metadata, and for helpfully answering many questions.

Every line contains thus a piece of information, preceded by an identifying label, corresponding in general to the path through the IMDI metadata structure to that piece of information. Each line contains one or more index numbers that make the line unique, such as, above, the session number (all lines) and the number of the language spoken in the recording (last line only).

An alternative format was also kindly provided to us, in which all the metadata fields are present, including the empty fields. Therefore, the text file has a much bigger volume. A sample for the same record as above, containing the same information, is given below:

```
Session_000000.URL="./data/L2/BCdata/D-DC1B1-RSB.imdi"
Session_000000.Name="d-dc1b1-rsb"
Session_000000.Title=""
Session_000000.Date="1987-01-26"
Session_000000.MDGroup.Location.Continent="unknown"
Session_000000.MDGroup.Location.Country="unknown"
Session_000000.MDGroup.Location.Address=""
Session_000000.MDGroup.Key__000={"discourse type","intention"}
Session_000000.MDGroup.Project.Name=""
Session_000000.MDGroup.Project.Title=""
Session_000000.MDGroup.Project.Id=""
Session_000000.MDGroup.Project.Contact.Name="dfg/prof. XXX"
Session_000000.MDGroup.Project.Contact.Address=""
Session_000000.MDGroup.Project.Contact.Email=""
Session_000000.MDGroup.Project.Contact.Organisation=""
Session_000000.MDGroup.Collector.Name=""
Session_000000.MDGroup.Collector.Contact.Name=""
Session_000000.MDGroup.Collector.Contact.Address=""
Session_000000.MDGroup.Collector.Contact.Email=""
Session_000000.MDGroup.Collector.Contact.Organisation=""
Session_000000.MDGroup.Content.Task=""
Session_000000.MDGroup.Content.Modalities=""
Session_000000.MDGroup.Content.CommunicationContext.Interactivity=""
Session_000000.MDGroup.Content.CommunicationContext.PlanningType=""
Session_000000.MDGroup.Content.CommunicationContext.Involvement=""
Session_000000.MDGroup.Content.Genre.Interactional=""
Session_000000.MDGroup.Content.Genre.Discursive=""
Session_000000.MDGroup.Content.Genre.Performance=""
Session_000000.MDGroup.Content.Languages.Language__000.Id=""
Session_000000.MDGroup.Content.Languages.Language__000.Name="german"
Session_000000.MDGroup.Content.Keys=""
```

The use of these initial files, and a short discussion of their comparative merits, is given in section 2. For now, let us give some *figures*: first, there are **1640 records** in both files (short one and complete one). The first file has **54306 lines**, corresponding roughly to the same amount of pieces of information – but recall that lines also have up to three index numbers, and some have also {key,value} pairs. This makes about **33 pieces of metadata information per record**. In the second file, there are about 113 information slots per record – the exact number depends on the repetitiveness of certain structures, such as the information per participant.

2.3.1.4 The interoperability problem

A problem related to metadata storage and search is the access to the metadata from a remote client computer. Apart from various proprietary solutions, an international initiative stemming mainly from the library community has proposed a common (minimal) metadata format and a standard

interrogation protocol. The format is known as Dublin Core[27], and the communication protocol between metadata archives is known as the Open Archives Initiative[28], with the associated OAI protocol[29]. The main idea behind these actions is that metadata is often free and public, and that it is a desirable goal to be able to share metadata between repositories, and enhance access to information.

The OAI protocol defines a certain number of "verbs" that a metadata server must be able to answer, through a HTTP connection with a client that wants to consult the available metadata. The format of the metadata is not fixed in advance, each server can provide metadata in a number of formats, but they have to be defined somewhere (in an accessible location). The constraints imposed on metadata providers are thus quite light: metadata must be at least provided in the Dublin Core format, and a small number of requests or 'verbs' must be answered – summarized here by order of specificity:

- Identify – retrieve a fixed set of information about a metadata repository;
- ListMetadataFormats – available metadata formats in the repository;
- ListSets – retrieve the set structure of a repository (its organization);
- ListIdentifiers – retrieve all record identifiers in a repository. Optional arguments may restrict the request to the records belonging in a certain subset, or modified/created/deleted at specific dates;
- ListRecords –same as ListIdentifiers except it retrieves whole metadata records;
- GetRecord – retrieve one metadata record, giving its identifier (typically as 'archive_name: record_number') and the desired format.

It should be noted that none of these verbs is really for metadata search. They have more to do with registering a metadata repository, declaring the metadata formats, and being able to retrieve the whole set of available metadata, i.e. *metadata harvesting*.

An important conceptual distinction has been proposed in the OAI framework: the roles of *(meta)data provider*, of *service provider* and of *client* (or user). The Digital Library Research Laboratory at Virginia Tech has designed a significant example of a client-browser solution, very useful to test the compliance of a data provider with the OAI protocol[30].

The OLAC[31] (Open Language Archives Community) is a partnership of institutions and individuals who attempt to use the OAI (by particularizing it to their needs) in order to create a network of language resources providers (metadata is generally free, but some language resources may be private or for sale). OLAC has designed one of the first metadata search engines[32], acting as a service provider through LDC[33], harvesting 18,000 records from 13 OLAC archives as of December 2001. OLAC has also defined a metadata set, an enrichment of Dublin Core targeted towards language resources, and an associated metadata harvesting protocol. More recently, the LinguistList has launched in December 2001 another OLAC service provider[34].

We will make some prospective suggestions regarding IMDI's position in this emerging framework in section 2.3.5. We will also provide some more precise suggestions regarding the use of a database, along with a web server and with modules adapted from open source repository software.

[27] http://dublincore.org/ .
[28] http://www.openarchives.org/ .
[29] http://www.openarchives.org/OAI_protocol/openarchivesprotocol.html .
[30] http://oai.dlib.vt.edu/cgi-bin/Explorer/oai1.1/testoai .
[31] http://www.language-archives.org/ .
[32] http://wave.ldc.upenn.edu:80/OLAC/sp-0.2dev/search.php4 .
[33] LDC, Linguistic Data Consortium, http://www.ldc.upenn.edu .
[34] http://saussure.linguistlist.org/olac/ .

2.3.1.5 Proposed solution for the search problem

We designed, implemented and executed the following solution to the metadata search problem. The first step, which was already implemented at MPI, was the conversion of the XML hierarchy to the type of files described above (one piece of information per line). We will outline here the conversion to tables (cf. explanations and examples in section 2.3.2) and the search process (cf. explanations and examples in section 2.3.3).

From the initial metadata file, we first designed the database tables that are needed. The idea is to avoid duplication of information, and to store unique information on each line, so that each table can be indexed. For instance, the 'session_id' table stores one line of information per session: the column values are the pieces of information that characterize a session and that cannot be duplicated, such as URL, date, and name. When such information can be duplicated, we created other tables for information that can be duplicated: for instance, there is a 'participant_id' table, with one line per participant (once again, 'name', 'age', 'sex' are unique for each participant). The 'session number' and 'participant number' constitute a unique index on this table.

With these tables in mind, we designed scripts that extract from the initial file the information corresponding to each table. Since some of the fields may be absent, we cannot extract the information directly in tabular form, but we extract on each line the index number (e.g., session_nb, or session_nb+participant_nb), the type of information (e.g., name, or date) and its value (e.g., John, or 1986-06-30).

These files are then input to a database system (here, PostgreSQL) to create temporary tables, containing only index numbers, info-types and info-values. Then we use a set of SQL queries to create the final tables, which contain the proper set of columns. These contain the same set of information, but they are coherently organized and allow for fast search.

The search process is highly facilitated by the SQL interface provided by the DBMS. This query language allows a variety of queries, and the optimization of the DBMS ensures that each query is optimized before execution. As shown in section 2.3.4, queries on the 1640-record database (with tens of thousands of pieces of data) perform very fast, typically less than 1-2 tenths of seconds per query.

2.3.2 Conversion of metadata records into database tables

The data format used as a starting point is the one described above (cf. 2.3.1.3), a text file with one piece of information per line (format: "complex.label.[index]=value"). We use the variant with no empty lines, but we have also written scripts for the format containing empty or 'unknown' lines (section 2.3.2.3). We start now by describing the pre-database scripts (2.3.2.1), then the input and conversion in the database itself (2.3.2.2). The whole protocol is finally discussed (2.3.2.4).

2.3.2.1 Primary conversion scripts

The primary conversion scripts transform the metadata from the MPI in-house (intermediary) format to a tabular form that will be used by the DBMS. Since the metadata file contains only the non empty fields, it is uneasy to build a tabular structure in the definitive form of the SQL tables, because many of the fields are missing, and the script would have to know which ones exactly are missing. This is done in an easier way using SQL commands.

The text of the Unix (Solaris 2.8) scripts is given in Appendix 4. They are all triggered by the 'mpi2db' command (see Appendix 4, 1.1), which takes as an argument the name of the initial file, and generates 10 table-files, corresponding to the table structure that will be created in the database

(see below). The main script calls the 'sed' command [35] on the initial file, with ten different extracting scripts (see examples in Appendix 4, 1.2 and 1.3). Each of them goes through the entire metadata file to extract information for each table. For instance, information for the 'SessionId' table is extracted from 'L2.dbf' using the 'sed' command and the 'SessionId.sed' command file:

```
sed -n -f SessionId.sed L2.dbf > SessionId.table
```

The result is an intermediary file that has, in the case of 'SessionId', pieces of information separated by a separation mark (here '----' for legibility), each piece of information having, on one line, the session number, the type of information and the information itself, as in this example with all the information available for the identification of the first session:

```
----
000000
url
"./data/L2/BCdata/D-DC1B1-RSB.imdi"
----
000000
name
"d-dc1b1-rsb"
----
000000
date
"1987-01-26"
----
000000
project_contact_name
"dfg/prof. XXX"
```

The following stage consists simply in rearranging these files in order to store each entry on one line, separated by vertical bars. This is already the final stage in script-based pre-processing, and its results will be given directly to the database. This operation is done by the 'oneline' script, which replaces 'newline' characters with vertical bars '|' and '----' separators with 'newline' (see text of script in A.1.4). The 'oneline' script pipes a very simple series of 'sed', 'tr' and 'tail' commands[36].

For instance, the previous file ('SessionId.table') is converted in the following format:

```
000000|url|./data/L2/BCdata/D-DC1B1-RSB.imdi
000000|name|d-dc1b1-rsb
000000|date|1987-01-26
000000|project_contact_name|dfg/prof. XXX
```

Another example is the extraction of the '{key,value}' pairs, such as the ones for the session. These are stored in 'SessionKeys.table', exemplified here:

```
000000|000|discourse type|intention
```

[35] The 'sed' command is preinstalled on Unix/Linux platforms (see http://www.gnu.org/software/sed/), and can be easily installed on Windows platforms (see http://gnuwin32.sourceforge.net/packages/sed.htm). Some useful links can be found at: http://www.math.fu-berlin.de/~guckes/sed/.

[36] The commands 'tr' (transform) and 'tail' (delete initial or ending lines) are built-in Unix/Linux shells, but they are also available for Windows, see the packages available at http://sources.redhat.com/cygwin/. The C source of these commands is also available via GNU (http://www.gnu.org/).

The extraction of all these tables is done by launching just one command: '> mpi2db metadata_file_name'. The scripts have been adapted to the present data, and a discussion of the procedure design and speed is given in section 2.3.2.4 below.

At this stage, the following 'tabular' files are thus created, ready to be input in the database. They were given explicit names, and are grouped in the following table by logical relations; the extension '.table' has been omitted everywhere.

session_id	participant_id	source
session_keys	participant_keys	mediafile
content_keys	participant_languages	annotation_unit
project_content_languages		

2.3.2.2 Creation and instantiation of the PostgreSQL tables

The definition (creation) of the tables and the transfer of the data into the tables are done in two steps, using intermediary tables. The goal at this stage is to obtain coherent tables, conformant to sound database design guidelines. One of the constraints is that each of the tables must be indexable using one or two of its columns, so that we avoid repetition of the information.

For instance, since a session can have many participants, it is correct to design two tables: one storing (non repetitive) information associated with sessions, and another one storing (non repetitive) information associated with each participant. The link between the elements is embodied in the 'session_nb' column associated to each participant, stating to which session he or she participates. Of course, at this point, one may point out that a given participant may take part in several sessions, so there is still some duplication of the information. This is true, but stems from the initial format of the metadata, which does not attempt to make unique records for each participant – rather, their name, age, sex, etc. are duplicated (copied) for each session. We do not attempt to unify here information about each participant ("no reference resolution"). If this were possible, we would remove the 'session_nb' column from the 'participant_id' table, and create a new table called 'participation_information' with two columns 'session_nb' and 'participant_nb'. Then, the two 'session_id' and 'participant_id' tables would be indexable on, respectively, 'session_nb' and 'participant_nb', but 'participation_information' would not be indexable at all.

Now, back to our conversion process. We use the PostgreSQL database engine, available for Unix/Linux platforms as well as for Windows[37]. We first create a database to store the tables, for instance launching the PostgreSQL interface, '> psql', then 'create database imdi' (this has to be done just once). Then, we create temporary tables that match quite closely the structure of the tables described above (indexing column(s), information type, information value). It is wise to delete all previous tables before starting this process, to avoid confusion or duplication.

For instance, for 'session_id', we use the following commands to create and fill a temporary table called 'session_id_temp':

```
create table session_id_temp(
   session_nb smallint,
   infotype text,
   infovalue text);
```

[37] See http://www.postgresql.org for information. Regarding Windows, PostgreSQL is pre-installed on the cygwin package (http://cygwin.com).

```
copy session_id_temp
from '/home/andrei/ISLE/IMDI/database_tests/SessionId.table'
using delimiters '|' ;

create index index_session_id_temp on
session_id_temp(session_nb);
```

The index that is created with the last command is of course far from being unique, since there are about six 'infotypes' per session – but for each 'infotype' there is just one 'infovalue', as described by the metadata specification: one name, one URL, etc. However, it is still useful to create an index to accelerate the access operations to the table.

The result is the following table structure, containing all the information relevant to sessions:

```
Table "session_id_temp"
 Attribute  |   Type   | Modifier
------------+----------+----------
 session_nb | smallint |
 infotype   | text     |
 infovalue  | text     |
Index: index_session_id_temp
```

This first stage is triggered by a PostgreSQL script called 'aliment-database.pg'. In fact, there is one main script called 'main.pg', launched very simply from the PostgreSQL interface by '\i main.pg'. This main script calls first 'aliment-database.pg', then calls the scripts relevant to the final stage ('load-*.pg'), which we describe now. Excerpts of the contents of all scripts are given in Appendix 4, section 2.

Therefore, regarding the second and final stage, we must note first that *for key-value pairs, there is nothing more do*. Indeed, the type of the key is not specified, so the tables cannot be processed (transformed) any further. The first step, as described below, is sufficient:

```
create table session_keys(
   session_nb smallint,
   key_nb smallint,
   key_name text,
   key_value text);

copy session_keys
from '/home/andrei/ISLE/IMDI/database_tests/SessionKeys.table'
using delimiters '|' ;

create unique index index_session_keys
on session_keys(session_nb,key_nb);
```

This generates the following table template, and stores into the table all the information that was in the 'SessionKeys.table' file.

```
Table "session_keys"
 Attribute  |   Type   | Modifier
------------+----------+----------
 session_nb | smallint |
 key_nb     | smallint |
```

```
 key_name   | text     |
 key_value  | text     |
Index: index_session_keys
```

The index is now unique (note the keyword ‘unique’ in its definition), because a ‘session_nb’ and a ‘key_nb’ uniquely identify a line of the table. The creation of a unique index highly accelerates the operations on the table. This is the case for all the three tables with keys: ‘session_keys’, ‘content_keys’, ‘participant_keys’.

For the other tables, the temporary “index-infotype-infovalue” tables must be converted to more detailed tables, in which the “infotype” correspond to columns, and one record occupies one line of the table. Let us follow here the example of ‘session_id’ by taking a look at the ‘load-session-id.pg’ script (see the Annex for the other ‘load-*.pg’ scripts).

First, the script creates the final ‘session_id’ table, with columns for all the allowed ‘infotypes’:

```
create table session_id(
   session_nb smallint,
   url text,
   name text,
   date date,
   project_contact_name text,
   continent text,
   country text);
```

Then, the following commands fill the final table with the data from the temporary one, column by column. The first ‘insert into – select – from’ below sets the index numbers, i.e. fills the ‘session_nb’ column with all the ‘session_nb’ from ‘session_id_temp’, taken only once (hence the ‘select distinct’). Then, for each possible ‘infotype’, the corresponding column is filled, here for ‘url’.

```
insert into session_id (session_nb)
select distinct session_nb
from session_id_temp;

update session_id
set url = (select infovalue from session_id_temp
            where
session_id_temp.session_nb=session_id.session_nb
              and session_id_temp.infotype='url');
```

The commands for filling the ‘name’, ‘date’, ‘project_contact_name’, ‘continent’, ‘country’ are similar to that for ‘url’ (two replacements must be made). It is of course somewhat awkward to be obliged to write a new command for each ‘infotype’: unfortunately, we were unable to write a unified command, which stores each ‘infotype’ in the corresponding column, since we were unable to translate the ‘text’ type of the ‘infotype’ column from ‘session_id_temp’ into a “column name type” for use in the filling query. So we had to duplicate quite a lot of code, with very few replacements.

Finally, a unique index is created for the new table. It is of course much faster to insert the data first, then to create the index. Updating of the table is of course possible at any time.

```
create unique index index_session_id
on session_id(session_nb);
```

The names of the ten resulting tables are those given in the table below – and we hope that these names are explicit enough. Below are the columns names and types (integer or text) for each of the tables, together with the number of lines (rows) with the test metadata (1640 sessions). At this final point, the metadata is converted and ready to be queried.

```
----------------------------------------
         Table "annotation_unit"
     Attribute      |   Type   | Modifier
--------------------+----------+----------
 session_nb         | smallint |
 annotation_unit_nb | smallint |
 resource_link      | text     |
 type               | text     |
 format             | text     |
 anonymous          | text     |
Index: index_annotation_unit

Rows = 1626

----------------------------------------
          Table "content_keys"
     Attribute     |   Type   | Modifier
-------------------+----------+----------
 session_nb        | smallint |
 content_key_nb    | smallint |
 content_key_name  | text     |
 content_key_value | text     |
Index: index_content_keys

Rows =  3311

----------------------------------------
          Table "mediafile"
   Attribute   |   Type   | Modifier
---------------+----------+----------
 session_nb    | smallint |
 mediafile_nb  | smallint |
 resource_link | text     |
 type          | text     |
 format        | text     |
 quality       | text     |
Index: index_mediafile

Rows = 1626

----------------------------------------
        Table "participant_id"
   Attribute    |   Type   | Modifier
----------------+----------+----------
 session_nb     | smallint |
 participant_nb | smallint |
 type           | text     |
 name           | text     |
 fullname       | text     |
 code           | text     |
 role           | text     |
 age            | text     |
 sex            | text     |
```

```
 anonymous      | text     |
Index: index_participant_id

Rows =  3461

----------------------------------------
          Table "participant_keys"
       Attribute       |   Type   | Modifier
-----------------------+----------+----------
 session_nb            | smallint |
 participant_nb        | smallint |
 participant_key_nb    | smallint |
 participant_key_name  | text     |
 participant_key_value | text     |
Index: index_participant_keys

Rows = 2446

----------------------------------------
    Table "participant_languages"
   Attribute    |   Type   | Modifier
----------------+----------+----------
 session_nb     | smallint |
 participant_nb | smallint |
 language_nb    | smallint |
 language_name  | text     |
Index: index_participant_languages

Rows = 3249

----------------------------------------
  Table "project_content_languages"
   Attribute   |   Type   | Modifier
---------------+----------+----------
 session_nb    | smallint |
 language_nb   | smallint |
 language_name | text     |
Index: index_project_content_languages

Rows = 2984

----------------------------------------
             Table "session_id"
      Attribute       |   Type   | Modifier
----------------------+----------+----------
 session_nb           | smallint |
 url                  | text     |
 name                 | text     |
 date                 | date     |
 project_contact_name | text     |
Index: index_session_id

Rows = 1640

----------------------------------------
       Table "session_keys"
 Attribute  |   Type   | Modifier
------------+----------+----------
```

```
 session_nb | smallint |
 key_nb     | smallint |
 key_name   | text     |
 key_value  | text     |
Index: index_session_keys

Rows = 3367

---------------------------------------
              Table "source"
      Attribute      |   Type   | Modifier
---------------------+----------+----------
 session_nb          | smallint |
 source_nb           | smallint |
 source_id           | text     |
 format              | text     |
 time_position_start | text     |
 time_position_end   | text     |
Index: index_source

Rows = 1836
```

2.3.2.3 An alternate solution

In case the metadata is provided in "complete" form, that is, with all fields present in the text file, even if some of them are empty (cf. 2.3.1.3), the procedure can be simplified a little. Of course, *the procedure described above remains perfectly valid*, but in the present case there is no need for intermediary tables, since the low-level Unix scripts can generate pre-database files in a tabular format close to the final tables.

For instance, here is the initial information for a session:

```
Session_000000.URL="../data/ladfc11a.1.imdi"
Session_000000.Name="ladfc11a.1"
Session_000000.Date="1982-11-09"
Session_000000.MDGroup.Location.Continent="Unknown"
Session_000000.MDGroup.Location.Country="Unknown"
```

The Unix scripts derive from this information directly the following line:

```
000000|../data/ladfc11a.1.imdi|ladfc11a.1|1982-11-09|Unknown|Unknown
```

Since all the fields are present, this line can be automatically entered in the database, using the following PostgreSQL command:

```
create table session_id(
   session_nb smallint,
   url text,
   name text,
   date text,
   location_continent text,
   location_country text);

copy session_id
from '/home/andrei/ISLE/IMDI/database_tests/SessionId.table'
using delimiters '|' ;
```

```
create unique index index_session_id on session_id(session_nb);
```

The final result is in both cases the same, but in the present case it is a little easier to obtain, and the coherence of the final result is also better ensured.

2.3.2.4 Discussion

At this point, we would like to provide some initial speed considerations. The first transformation stage (Unix scripts), applied to the 1640-record set, takes about 40 seconds on a SunBlade 100 workstation. The second transformation stage, executed under PostgreSQL, takes about 30 seconds on the same workstation. These operations must be performed only once, when the database is created and alimented. Then, the tables can be enriched either directly through the PostgreSQL interface, or by converting the new metadata using the same process, and inserting it into the new tables. Of course, if the metadata format changes, the scripts must be modified accordingly, and the whole conversion process must be run again.

Testing is important in order to ensure that *all* the information given in the metadata file has been entered into the tables. Here, we performed for each table a count of the lines of the initial file going into that table (e.g., 'annotation_unit'), then a count of the lines in the corresponding '*.table' file (e.g., 'annotation_unit.table'). The numbers were the same for all ten tables (e.g., 6504 for 'annotation_unit'), and their total equaled the total number of lines of the initial metadata file (54306). Moreover, we checked the number of pieces of information for each table. E.g., for 'annotation_unit', there are 6 pieces of information per line for 1626 lines, and no such information for 14 lines – more exactly for sessions 838-851 – hence a total of (6 – 2 index) x 1626 = 6504.

2.3.3 Querying metadata tables

The IMDI showcase solution to search in a metadata file is based on in-house Perl scripts that process the initial metadata files, using mainly text-based matching, and defining a special query input interface. We believe however that it is simpler and more efficient to take advantage of the database model in order to write flexible queries in SQL, that are optimized by PostgreSQL and run much faster than an ad-hoc implementation.

The experience of the Max-Planck Institute with metadata querying was kindly shared with us: the MPI team provided *a set of test queries*, formulated in natural language and also using the in-house search mechanism. We translated and tested these queries, first confirming that they were easy to express in SQL, and that SQL added powerful possibilities such as sorting, counting, multiple joins, etc. (to say nothing about speed).

The entire list of queries that were tested is given in Appendix 4,3.1: we provide first the natural language formulation of the query, then the MPI formulation, then the SQL formulation. It is quite visible that the SQL formulation is a little longer, but a little more explicit too. In Appendix 4, 3.2, we give also the results of the queries, just after each query. These files were produced using PostgreSQL scripts, so that we can test the entire series of queries automatically. The time needed for the entire set of 17 queries, including the storage of the output in one file, is less than 5 (five) seconds using the previous tables.

Here are some significant examples of the possibilities given by SQL. The full results are given in an appendix, here we give the results for last query only:

```
-- Give me all female participants aged 30 to 39

select distinct name, age
from participant_id
where sex = 'female'
  and age >= 30
  and age <= 39 ;
```

```
-- Give me all sessions from June 1986 ordered by date

select session_nb, name, date
from session_id
where (date >= '06-01-1986'
  and  date <= '06-30-1986')
order by date;
```

```
-- Give me all combinations of one male and one female
-- participant in a session

select p1.session_nb, p1.sex, p1.participant_nb,
       p2.sex, p2.participant_nb
from participant_id p1, participant_id p2
where p1.session_nb = p2.session_nb
  and p1.sex = 'male'
  and p2.sex = 'female' ;
```

```
-- Give me the languages spoken in sessions with two female
-- participants, and the number of sessions for each language

select  pcl.language_name, count(p1.session_nb)
from participant_id p1, participant_id p2,
project_content_languages pcl
where p1.session_nb = p2.session_nb
  and p1.sex = 'female'
  and p2.sex = 'female'
  and p1.participant_nb < p2.participant_nb
  and pcl.session_nb = p2.session_nb
group by pcl.language_name ;

    language_name    | count
---------------------+-------
 czech               |   347
 czech sign language |    33
 english             |    64
 german              |   103
 vietnamese          |    12
(5 rows)
```

2.3.4 Evaluation of the Database Solution

The table below provides a series of evaluation criteria, based on standard needs for metadata management and search. For each criterion, the behavior of the two approaches to search is described. We do not, however, attach a given weight to each criterion, in order to provide a final "score":

Criteria	**Original metadata format and showcase implementation (IMDI query language)**	**Database conversion scripts (Unix) and SQL queries**
Programming effort	Initial conversion script (XML to 'field=value' text file) High programming cost to implement the querying mechanism	Same initial conversion effort Conversion to database scripts Use of existing database / SQL implementation. High programming costs to develop a suitable query interface.
Speed of conversion to query-able format	No conversion apart from XML.	Several conversion steps. Total time: under 2 minutes for 1640 sessions.
Flexibility with respect to the conversion to metadata format	The query mechanism is the only aspect to modify	The conversion scripts and the table creation scripts must be changed if the metadata format changes
Complexity of storage solution	Text files (nothing to do)	Database (DBMS needed)
Robustness of storage	File back-up needed	Database back-up
Access to metadata	Access to local and distributed remote files.	Distributed access (DBMS server plus clients), secure access (user-based control)
Expressiveness of query language	Selections, joins	Full SQL at the backend, the UI defines the power offered to the user.
Query speed (samples)	7 seconds for one-match query on a 4000-session database (Sun workstation)	5 seconds for 17 queries (including one simple join and four double joins) on a 1640-session database (SunBlade 100)
Requirements for local querying of metadata	Copy or access to the file Copy of the query script User-friendly interface needed or knowledge of in-house query language	Access to the database Possibility to launch a db client User-friendly interface needed or knowledge of SQL

A summarizing discussion of the three approaches is done in chapter 2.4.

2.3.5 Towards a metadata server for OAI/OLAC interoperability

The interest of using a database system lies also in the possibility to interface it efficiently with a web server, in order to provide access to the stored metadata (at least to the public records) through the OAI protocol (see introduction, section 2.3.1.4). The main challenge to attain this goal has been answered: a mapping was defined between the Dublin Core metadata set (DCMS) and the IMDI set (necessary since all OAI compliant archives must be able to return metadata in DCMS format). This

mapping is available on the IMDI web site. Consequently, a mapping between IMDI and OLAC, deriving straightforwardly from the DCMS/IMDI one (OLAC only slightly enriches DCMS) is also available.

However, several other issues must be studied in the case of a DBMS solution:

- provide scripts that record creation/modification dates for metadata, and convert metadata from database format to XML;
- provide a web server, together with the software that converts HTTP requests into calls to the database, then waits for the results and returns them to the web server.

Regarding this last point, we have explored the possibility of using the E-Prints open source software[38] in order to answer OAI/OLAC requests *via* a web server. E-Prints is primarily a system for self-archiving scientific publications, with an interface answering OAI protocol. The database engine that this system uses is mySQL, coupled *via* Perl scripts to an Apache web server. Of course, most of the E-Prints scripts are dedicated to publications storage and retrieval, but some of them deal specifically with metadata retrieval upon OAI requests.

More specifically, we have explored the changes to be made to E-Prints version 1.1 modules in order to adapt them to our problem. Section 4.5 (p. 22) of the *E-Prints Installation Manual*, and section 7 of the *E-Prints System Documentation* (p. 16) explain how to configure various site variables. E-Prints consists of Perl modules that are executed by the Apache web server on demand, using *mod_perl*. When using E-Prints only for answering OAI request, many of the modules in '/EPrints/perl_lib/EPrints' and '/EPrints/perl_lib/EPrintSite' can be dropped. Basically, the options in '/EPrints/openarchives' must be changed, and the following modules revised: '/EPrints/perl_lib/EPrints/OpenArchives.pm', '/EPrints/perl_lib/EPrints/Database.pm', '/EPrints/perl_lib/EPrintSite/SiteInfo.pm'. It must be noted that one has to use mySQL, but the whole conversion and search process described above (sections 2.2.2 and 2.2.3) is applicable as is to the mySQL DBMS, since all the commands belong to standard SQL language. Adapting E-Prints to our problem is thus a tractable task, which seems significantly easier than writing from scratch the modules that answer OAI requests.

The OLAC team has realised the process of harvesting OLAC/DC metadata records and storing them as database entries as well. Given the different structure of the richer IMDI metadata set this solution cannot be chosen for a central IMDI database.

2.4 Comparative evaluation of the search strategies

The aim of the present sub-section is to compare the three approaches: (1) the IMDI metadata search architecture that is shown in chapter 2.1 and that was partly implemented in the IMDI showcase, (2) the solutions discussed in chapter 2.2, and (3) database-oriented approach described in section 2.3. Of course, a thorough evaluation should first define a user profile, a set of criteria for success, etc., and then compare the different approaches. Here, we will only summarize the main points that seem relevant to us, from the viewpoint of researchers in language engineering and field linguistics acquainted with corpus manipulation problems. Neither will we discuss the required level of detail in a metadata set - please refer to D10.1 and D10.2.

2.4.1 Requirements

In previous chapters a number of requirements for an optimal search environment were mentioned. The major ones are summarized here again:

[38] http://www.eprints.org . Version 2 of the software has just been released.

1. The search interface has to give access to all useful elements of the IMDI set and support the controlled variables and constraints in an easy way. Therefore, it has also to be able to support the user definable keyword /value pairs to achieve flexibility for the researchers.
2. The search solution has to deal with the fact that at certain moments these metadata definitions will be changed. Therefore, it must be possible to adapt the solution in an easy way to adaptations of the underlying schema.
3. The search solution must also support a continuously changing metadata domain, i.e. continuously new metadata descriptions will be added, some old will be deleted and some will be upgraded.
4. The query mechanism must offer the usage of variables to express questions such as "give me all resources where consultants are female speakers and have an age of 6".
5. The tool implementing metadata search should allow the user to do interactive browse and search for optimal navigation purposes and easy management. It should also be possible to use the search result (a number of sub-corpus) as a new personalized corpus for efficient operation.
6. The search solution has to support searching in local and distributed networked metadata repositories. Local work often means that users work also off-line, i.e. not being connected to the Internet.
7. Since support for local operation is a must, a simple mechanism has to be available which is not dependent on large and expensive software components.
8. The performance of the search has to be as fast as possible, especially since it can be expected that the number of metadata descriptions included in the IMDI domain will increase constantly.
9. The search client solution should offer platform independence.
10. Where possible existing tools and concepts should be re-used.
11. The IMDI and OLAC/DC metadata domains should form an interoperable domain, of course, knowing that IMDI to OLAC transformations are not without loss.

2.4.2 Discussion

It should be noted here that the MPI solution was meant to meet a number of criteria, to achieve a high level of integration with the other tools developed for the IMDI metadata framework and to aid corpus management. It is described in chapters 1.3, 1.4 and 2.1 (as well as the corresponding appendixes) in more detail in how far these criteria were met. It was agreed that the other two approaches would focus on principle search solutions especially for large data sets and central services. Therefore, some of the requirements are not seen as relevant for this comparative discussion.

Short Characterization of the Three Solutions

As said the MPI approach was mainly directed to meet all criteria mentioned except to achieve the best speed performance possible. This approach was necessary to build a fully operational metadata infrastructure that can immediately be used within the limited timeframe of the ISLE project. The IMS work describes how searching could be done remaining in an XML domain, i.e. using the possibilities of emerging XML tools. The ISSCO approach was based on looking to possibilities to include relational database management systems in a metadata infrastructure. Here of course conversion, interfacing, query language power and performance aspects are of highest relevance. While simplicity of the setup is relevant when thinking of solutions operating both locally and in networked environments, for centralized solutions speed is a key aspect. Therefore, the three approaches have to be seen as complementary for the purposes of this report.

XML-oriented Solutions

As was explained in chapter 2.2 there are many arguments in favor of a complete XML-based solution. A particular reason for this is the fact that the metadata descriptions themselves are created and stored as XML formatted files. However, the analysis carried out by IMS shows that XML-based

technology is partly not suitable and only software versions that recently appeared and where we don't have enough information about robustness (and other features) is attractive from functional perspectives. Indeed parsing XML at query-time is too slow and specific search programs available have great drawbacks. With the new version of eXist and XQEngine there may be software which can be applied in a productive environment. At MPI experiments with eXist were also carried out and Oracle XML components are tested right now, but it is too early to make statements. However, the emerging XML technologies could play a role locally as well as for a central solution.

Perl-based Solutions
Two Perl-based solutions were used and tested. The MPI version, which uses special service files and a non-optimised search script, shows on a large set of metadata records (10.000 sessions with many extra keyword/value pairs which were not part of the corpus provided to the partners) that the seek times may not be acceptable for all usage scenario's. 120 seconds to search through a single corpus may be too much for a user to be accepted. The numbers for smaller corpora (7 seconds for a one-match query with 4000 sessions for the MPI case and 19 seconds for 17 queries on 1300 sessions in the IMS case) indicate that optimization can improve the speed. The IMS solution is based on optimized representations of the XML files. However, they require more preprocessing, administration overhead and the storage of several files. For every attribute a separate index file is generated.

Database Approach
To establish a central metadata querying service the approach with relational databases seems to be the most promising at this moment when enough resources for preparing the data are available. The opportunity to use a relational database to store metadata, operate search on it, and distribute it through a 'metadata provider' server has only been studied at the *feasibility level* (cf. section 2.3). The proposed conversion process and its evaluation show that the database solution is an efficient and reliable one, provided of course all metadata users can access the database through a computer network. The generation of suitable indexes as described in chapter 2.3 leads to enormous speed gains. In addition, one can take profit from the robustness and reliability of current database software. A major disadvantage is the conversion aspect, but this can be solved by suitable scripts that (1) gather the available service files (as depicted in 2.1) at regular times (this can be done with the help of the OAI protocol that is already implemented), (2) convert them and (3) make them available. Such an approach will require more effort when the IMDI schema will be changed, since adaptations could even affect the database and tables structure as was analyzed carefully in 2.3. Interfacing the IMDI Search Tool with a central database is not a problem since the tool has been designed to function with different search modes (see 2.1). However, the code would also be subject of modifications in the case that the IMDI Schema will be changed.

Given the constraint of installing and storing local IMDI records in a database system on every computer that needs (also) local metadata search it still seems to be advisable for the future to operate with a dual system where simple Perl-based and powerful database solutions co-exist. The interaction between those two worlds has to be checked more carefully. New developments in the area of XML databases have to be studied since they could facilitate the conversion and database building process.

3. The IMDI Showcase

3.1 Corpus Data in the IMDI showcase

As described the IMDI concept is based on a domain of metadata descriptions linked by URLs. This simple mechanism allows users to easily construct a fully distributed repository of metadata descriptions where the nodes and session descriptions can be on any server that is connected to the

web and talks HTTP. This feature was used to create a first version of the IMDI showcase that was demonstrated at the official opening event of the European Year of the Languages in Lund in 2001. In this first showcase metadata from six European institutions (ELRA, Lund U, Helsinki U, MPI Leipzig, MPI Nijmegen, Lacito Paris) were combined to one distributed metadata domain.

In the meantime the available IMDI metadata universe covers the following IMDI tagged corpora:

1. The MPI corpora of the "Acquisition" and "Language and Cognition" groups. Included in these corpora are the metadata descriptions and trees the researchers at the MPI use to organize their vast number of resources. The total corpus currently covers more than 7000 metadata descriptions.

2. The ESF second language acquisition study corpus. This corpus does not only offer metadata descriptions but also the speech resources themselves allowing users to test the use of transcription/media viewers directly launched from the Browser.

3. The DOBES corpus covering endangered languages. This corpus covers many languages from almost all continents. Due to its variety and its richness in additional information it is an ideal corpus to test the usefulness of the IMDI definitions.

4. The Lund demonstration data as a corpus with examples of (parts of) corpora of six different European institutes (see above).

5. The Dutch Spoken Corpus data. This corpus of 10000 sessions that all have connected audio recordings of Dutch and Flemish speakers and is intended to be a reference corpus for the Dutch language. We have permission to only make the first audio CD available on-line.

6. Some test sets of corpus data that were used to gauge the suitability of the IMDI set and tools as exploitation environments for those corpora. These tests range from the well-known "Childes" corpora to language engineering corpora as "TIMIT". The testing of IMDI-Tools for "childes corpus" exploitation is done together with the group at CMU, the creators of childes. The CMU's "CLAN" tool, a transcription and media viewer for childes format transcriptions, was incorporated in the IMDI-BCBrowser as an executable local tool.

Within the ISLE project there was no money to support institutions in generating metadata descriptions. The institutions were interested to participate and test the methods, but were not able to spend many resources on the ISLE project. Therefore, the contributions of some institutions have to be seen as examples to prove the concept and show a general interest.

The experience gathered within ISLE shows that creating rich metadata is a time-consuming task, since normally the creation of metadata goes along with an organization of a corpus that beforehand was in a more or less chaotic state. The other major experience is that the new services are especially interesting for the domain experts when the descriptions are rich. Poor descriptions do not animate people to use it. Within the ISLE project this omission could not be solved since there was no budget for extensive on-site training and help. This problem will be tackled in two follow-up projects.

This "show-case" corpus data is available through a standard bookmark in the IMDI-BCBrowser.

4. Future Developments

4.1 Future Developments of the Tools

There is a need for further tool development. Such as a tool offering the users a graphical interface for creating alternative "personal" corpus trees, a tool that allows the user different views on collections of metadata descriptions and allows the user to modify specific metadata elements of all descriptions all at once. There is also further need of maintenance programs and scripts that will allow users to copy parts of corpus trees to other portable media such as CDROM and DVD. In this way they can work under field conditions or make personal archive copies that are of interest for language engineers who want to easily select corpus parts to train their recognizers.

With respect to the existing search infrastructure it makes sense to formalize the exchange between SearchTool and the IMDI record repository by using emerging standards such as WDSL and SOAP. Implementing this will be part of the future INTERA project. Also the query specification format should preferably move still closer to Xpath.

Since as stated above IMDI has a place in several long-term projects we expect do some of this development work within these projects. Therefore a long-term support seems to be assured. Currently, the IMDI tools are free to be used by academic institutions. It seems to be wise to change to an open source model for distributing the tools. Possibilities will be evaluated until the end of this year when a suitable model will be chosen.

4.2 Future Developments of the IMDI domain

IMDI is now already accepted by projects in Europe and the US. Participation in coming conferences and meetings will broaden the basis. Presentations at the recent LREC conference raised much interest in the IMDI work [1-10]. Since the most difficult part is to convince researchers to invest time to create such metadata descriptions, it is important to have a critical mass of metadata descriptions in the domain and to distribute the experience that applying the IMDI framework is "data and knowledge management" which pays off in the long run. At the Max-Planck-Institute the break-even point where people recognized the added value was already reached. In many other institutes the awareness is spreading at this moment.

The MPI team will continue to offer training courses and organize workshops to advance the IMDI environment. The launch of follow-up projects will be used to

- generate a critical mass of metadata descriptions
- establish hopefully a central European repository site
- extend the functionality of the IMDI domain by connecting resource and tool repositories
- add resources of other disciplines into the IMDI domain

More arguments about future perspectives can be found in deliverable 10.2.

5. References

[1] P. Wittenburg, U. Mosel, A. Dwyer: *Methods of Language Documentation in the DOBES Program.* Proceedings of the LREC 2002 Conference. Las Palmas, May 2002

[2] R. Skiba, H. Brugman, D. Broeder, P. Wittenburg: *Corpus Organization and Access in Field Linguistics at the MPI.* Proceedings of the LREC 2002 Conference. Las Palmas, May 2002

[3] D. Broeder, F. Offenga, D. Willems: *Metadata Tools Supporting Controlled Vocabulary Services.* Proceedings of the LREC 2002 Conference. Las Palmas, May 2002
[4] D. Broeder, P. Wittenburg, T. Declerck: *LREP: A Language Repository Exchange Protocol.* Proceedings of the LREC 2002 Conference. Las Palmas, May 2002
[5] P. Wittenburg, W. Peters, D. Broeder: *Metadata Proposals for Corpora and Lexica*. Proceedings of the LREC 2002 Conference. Las Palmas, May 2002
[6] P. Wittenburg, D. Broeder: *Metadata Overview and the Semantic Web.* Proceedings of the International Workshop on Resources and Tools in Field Linguistics. Las Palmas, May 2002.
[7] R. Guirardello-Damian, R. Skiba*: Trumai Corpus: an Example of Presenting Multi-Media Data in the IMDI-Browser*. Proceedings of the International Workshop on Resources and Tools in Field Linguistics. Las Palmas, May 2002.
[8] P. Wittenburg: *Metadata - Future Perspectives and ISO Tasks.* ISO TC37/SC4 Foundation Meeting. Las Palmas, May 2002
[9] P. Wittenburg, D. Broeder: *Management of Language Resources with Metadata*. Workshop on International Standards of Terminology and Language Resources Management. Las Palmas, May 2002
[10] P. Wittenburg, D. Broeder, F. Offenga, D. Willems: *Metadata Set and Tools for Multimedia/Multimodal Language Resources.* Workshop on Multimodal Resources and Multimodal Systems Evaluation. Las Palmas, May 2002

Appendix 1

IMDI Editor, version 0.9

Manual

This manual was last updated: 28 Feb 2002
The latest version of the manual can be downloaded from the following webpage: www.mpi.nl/tools
Author: Birgit.Hellwig@mpi.nl

Introduction

The IMDI (ISLE Metadata Initiative) Editor was developed at the Max Planck Institut für Psycholinguistik, Nijmegen, The Netherlands, with the purpose of helping you to organize the data that you work with in a standardized way. Much of your data will fall into three broad categories: (a) media tapes and files (that contain audio or video recordings), (b) annotation files (that contain transcripts and analyses), and (c) metadata files (that contain descriptive and administrative information about the data). The IMDI Editor targets this last type of data, the metadata files, in that it provides a format for entering information such as, e.g., the date and location where the data was collected, the larger project to which the data belongs, the participants involved, a description of the content, the labels of the original audio and video tapes, links to the corresponding annotation and digitized media files, etc.

The IMDI Editor does not introduce any new categories in this respect, but rather integrates the various types of information that you would have to store in one way or another anyway. This is done through providing a standardized format, which makes it possible to support features that allow you to enter information with a minimum of effort. Particularly, the following features are supported:

- templates that allow you to supply recurring information only once, and which can be re-loaded every time they are needed again;
- pull-down menus of controlled vocabulary that allow you to choose among items.

While these features are intended to make the creation of metadata files easier, the IMDI Editor serves a second purpose: preparing the display of the entered information in a standardized way in order to facilitate both the search process and the access to the corresponding media and annotation files. This is the function of the IMDI Browser, which links all of your data (media files, annotation files, metadata files) and thereby makes it possible for you to simultaneously access all of them through a single interface, viz. your desktop computer. The prerequisite for being able to use the IMDI Browser is the existence of metadata files of the IMDI Editor format.

We therefore ask you not to see the IMDI Editor as an end in itself, but rather as a tool that helps you to organize your data. This manual will help you to understand the logic of the IMDI Editor, and it will provide you with explanations and examples of how to fill in the different fields. It is organized around four parts:

I. The IMDI Browser and the IMDI Editor

This part introduces the IMDI Browser, illustrating how the files that you create with the IMDI Editor are displayed and used in the Browser.

II. Getting started

This part introduces (a) the key structure, concepts and terminology of the IMDI Editor, and (b) those features of the Editor that allow you to enter recurring information.

III. The User's Guide to the IMDI Editor

This part provides detailed information and examples of how to fill in all fields of the IMDI Editor. It is organized on a screen by screen basis.

IV. Appendices

This part provides brief summaries of the key concepts of the IMDI Editor and of its internal structure. In addition, it gives lists of all the recommended vocabulary items that are to be used in certain fields of the Editor.

Before you can use the IMDI Editor efficiently for your individual purposes, you have to have a sense of its overall structure, its possibilities and current limitations. We therefore advise you to read especially the information contained in part II (Getting started) before you embark on using it for describing your own data.

Notation Conventions

The following notation conventions are used:

- Major screens of the Editor start with a caption on a gray background.
- Screens, schemata and fields of the Editor are written in the font MS Sans Serif.
- (SHORTCUT) KEYS ARE WRITTEN IN SMALL CAPS.
- "Text in double quotes refers to items in a pull-down menu, and to all menu items."
- Information on troubleshooting starts as follows: **!**

Table of Contents

PART I: THE IMDI BROWSER AND THE IMDI EDITOR

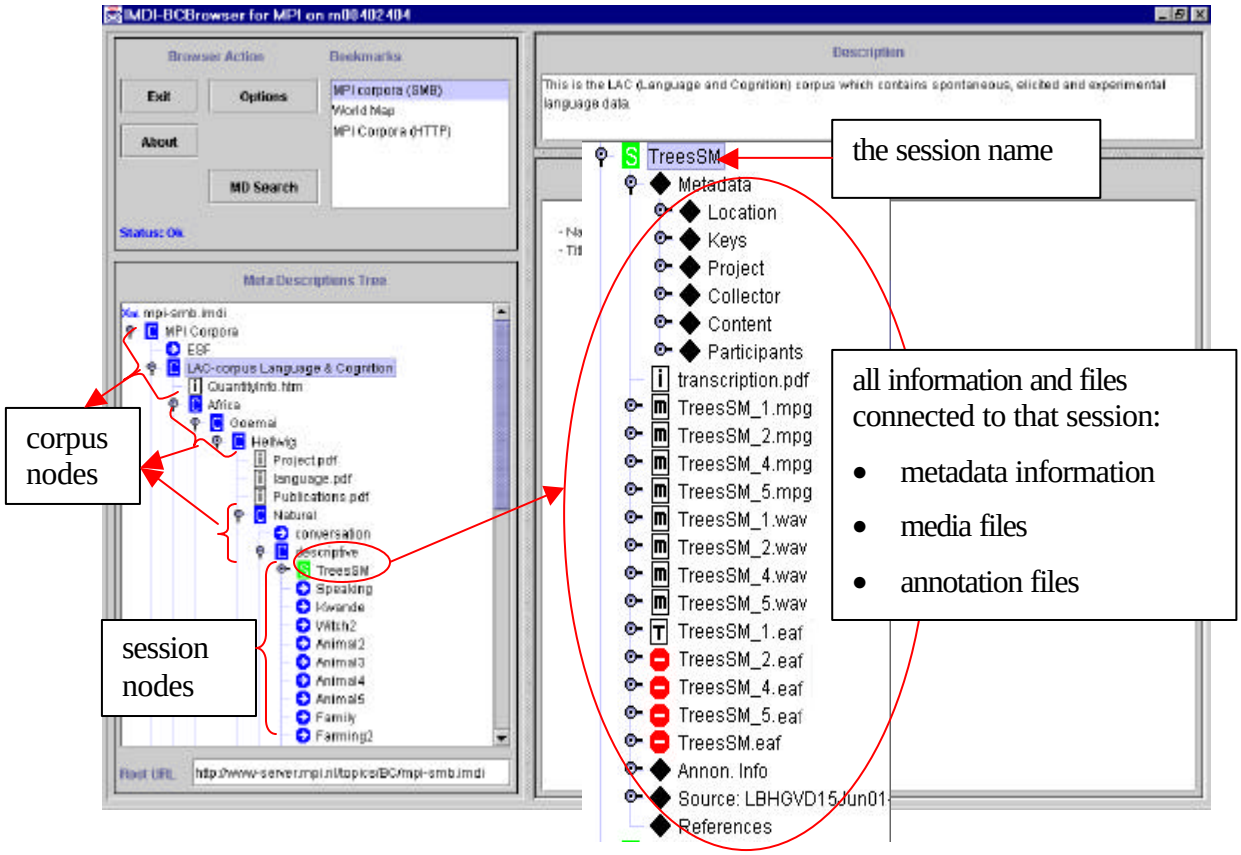
This manual describes the IMDI Editor. However, entering information into the Editor is not a goal in itself, but rather the first step in a more general process of organizing, archiving, and displaying the annotation and media data that you work with. The present part introduces this larger framework in order to give you some idea as to how the IMDI Editor is integrated into it. It illustrates the following three points:

- how the final organization of your data may look like;
- how the information that you enter into the IMDI Editor will be displayed;
- and how you can work with the information that you enter in the IMDI Editor.

Note that this part is just meant to give an illustration of how the final product may look like. A separate manual will explain these points in much more detail.

1 The organization of your data: The IMDI Browser

Your data will contain files of the following three types: (a) audio and video files, (b) transcription and annotation files, and (c) metadata files (i.e., IMDI files created with the IMDI Editor). A specific tool developed at the Max Planck Institut für Psycholinguistik, the so-called IMDI Browser, links these three types to each other, integrates them into a hierarchy and displays this hierarchy on your computer screen, e.g.:



The screenshot above illustrates the organization of the corpus collected for the language Goemai. It is organized in form of a hierarchy that contains nodes and subnodes, which allow you to navigate through to the actual data. For example, the node 'Natural' contains all types of natural texts. It is made up of a number of subnodes, which specify genres of natural texts, e.g. 'descriptive' (texts). This node, in turn, is made up of a number of subnodes, which contain the individual sessions.

A 'session' is a coherent unit of your data. It is the unit that is described by one IMDI file. Such a unit can contain, e.g., a particular descriptive text told by a particular speaker at a particular date and time (like the text named 'TreesSM' in the screenshot above). The session node displays all the information relevant to this particular text. It contains the following types of information:


link to the metadata file of the session (i.e., the IMDI file).


link to the digitized media file(s), containing the audio/video data of the session.


link to the transcription and annotation file(s) of the session.


link to the info file(s) providing general background information.

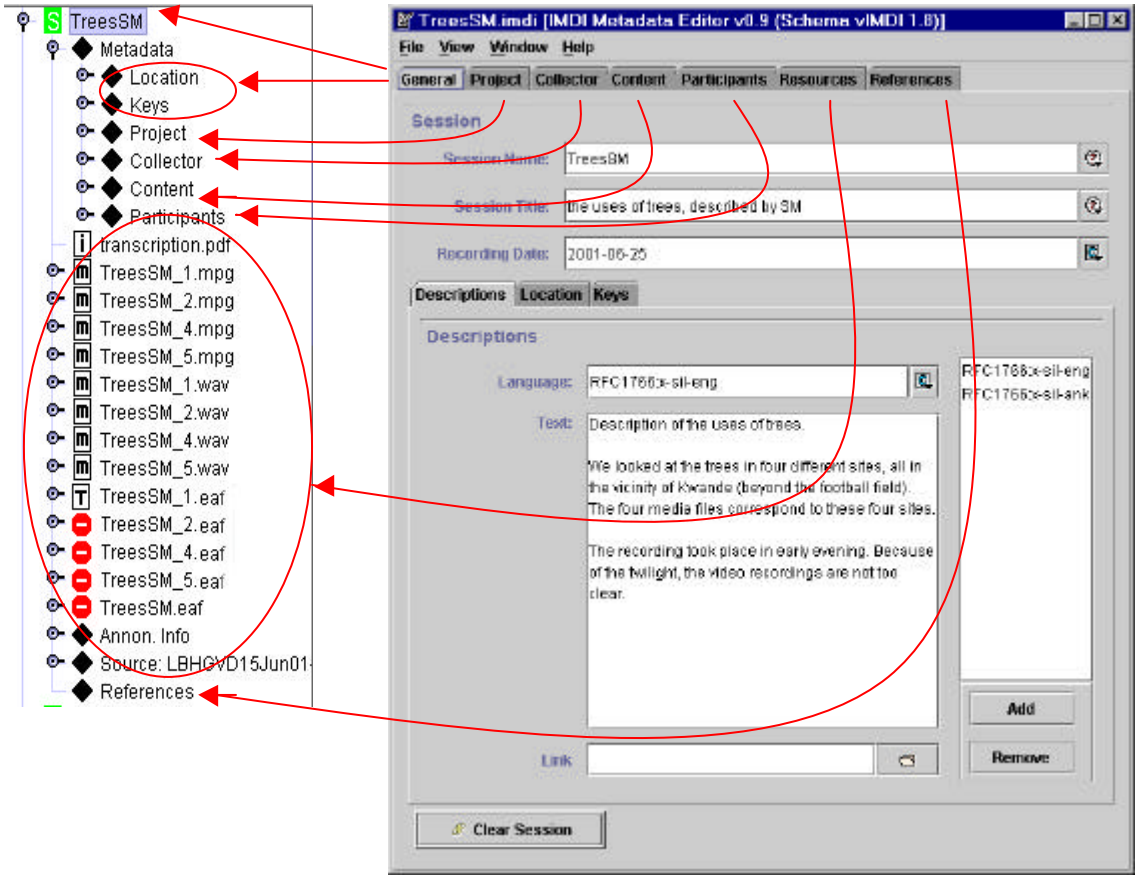

link not available (either because the file does not exist yet, or because access is denied).

2 The relation between the IMDI Editor and the IMDI Browser

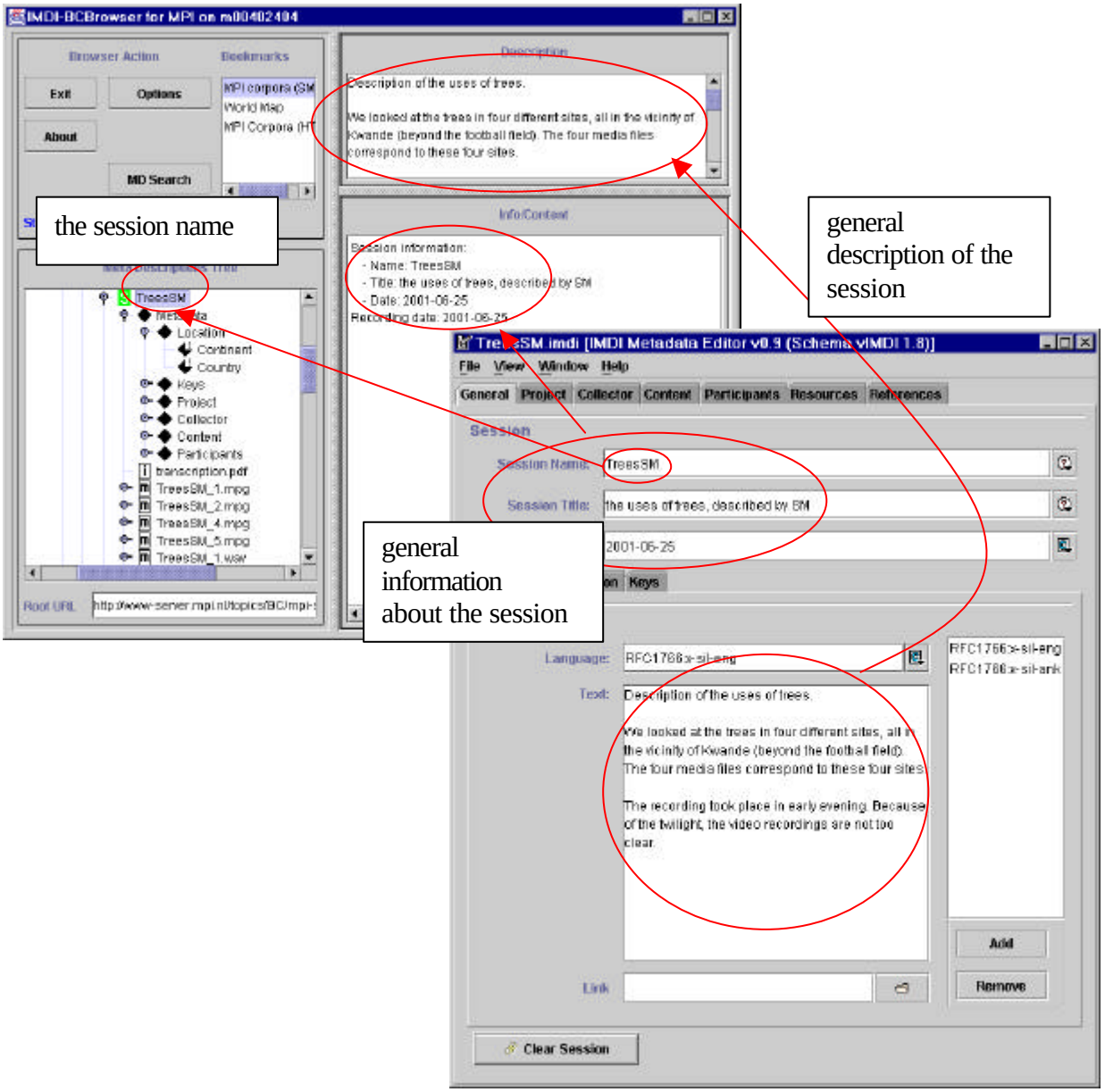
At the session level, the IMDI Browser displays metadata information. This metadata is all the information that you enter into the IMDI Editor: the participants in a session, the session's location and date, its content, its corresponding audio and/or video tapes, etc. In the following screenshot, you see how the structure of the Editor corresponds to the nodes displayed in the Browser.

IMDI **Browser**:
display of session information

IMDI **Editor**:
entering session information



More specifically, the following screenshot shows one example of how the actual information (entered in the Editor) is displayed in the Browser.



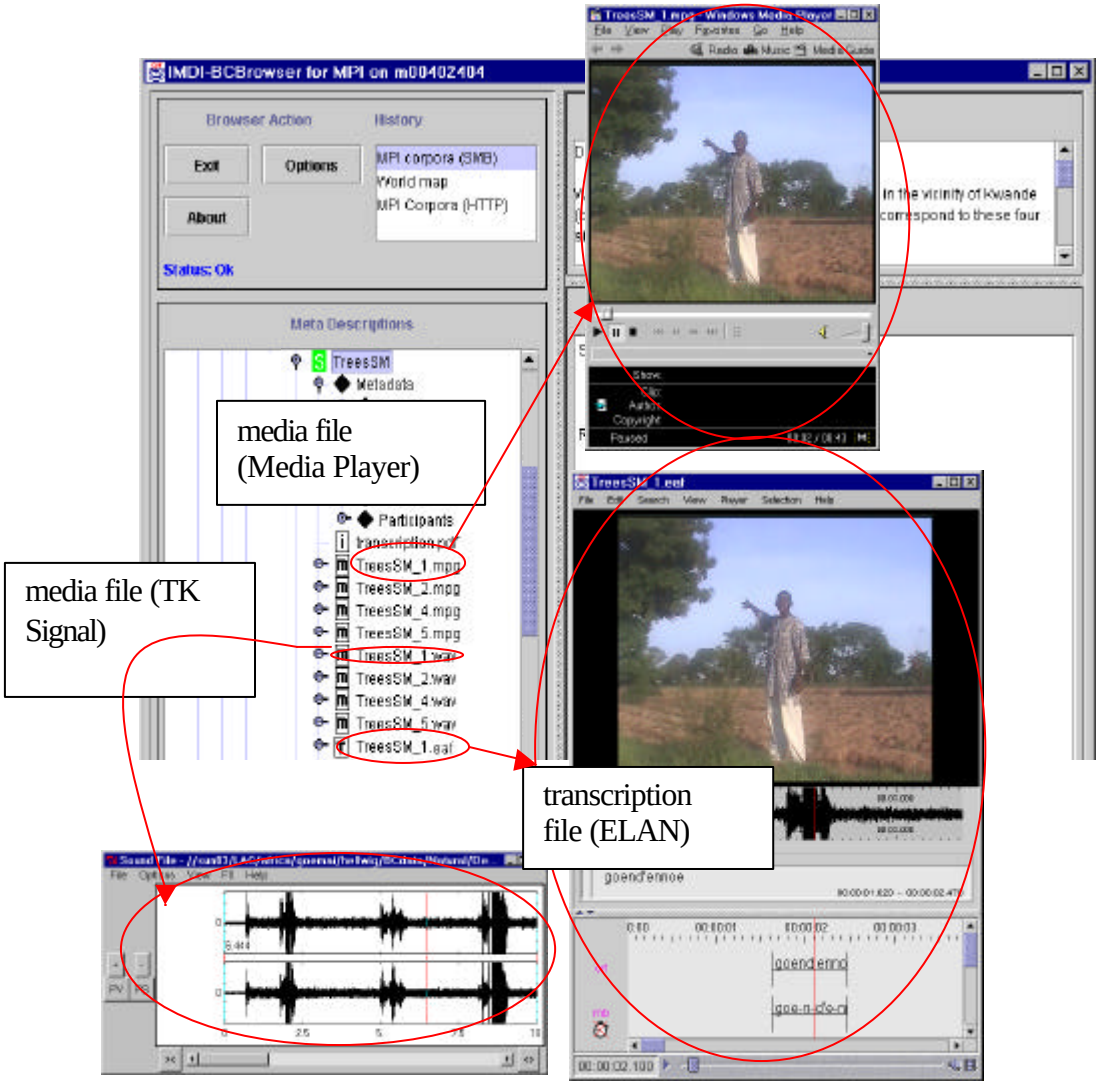
3 Using the IMDI Browser

The purpose of the IMDI Browser is to ensure easy access to the data. It supports the following features:

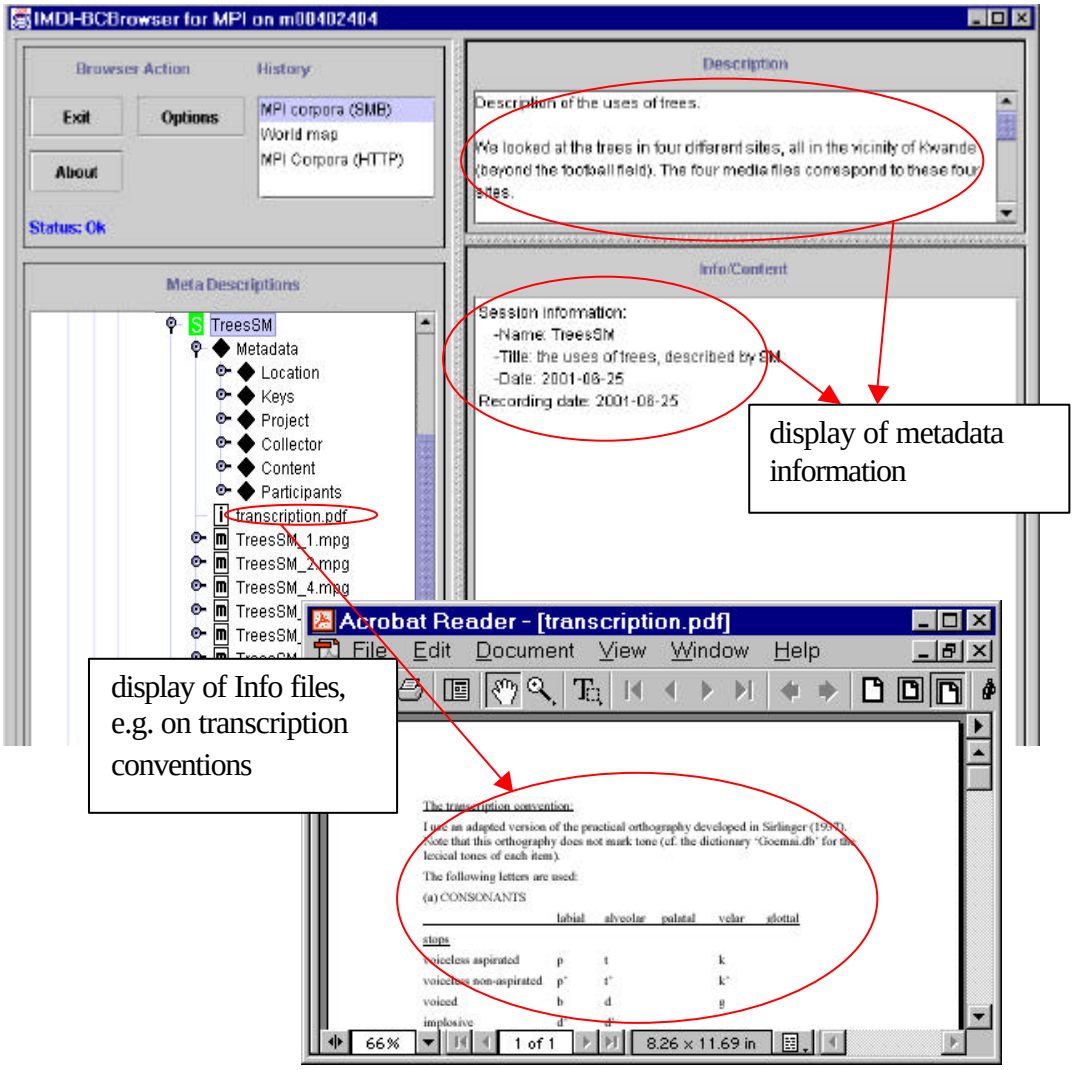
- The hierarchy of nodes and subnodes allows you to organize your data in a systematic way. For example, you can navigate through to a subnode, e.g., to the node ‘descriptive’, which will immediately display all sessions that contain such a text.

  Note that the Browser can display the same session in various nodes (e.g., a session can be simultaneously displayed under the nodes ‘descriptive’, ‘male speaker’, ‘age-group 20 to 30 years’ and ‘dialect Kwo’).

- The media and annotation files displayed under the session node can be accessed immediately. Just click on the relevant icon and the corresponding file is opened.



- The metadata information displayed under the session node allows you to quickly read up on the circumstances under which the data was collected, e.g.:



- The Metadata information can be searched. Note that this feature is not implemented yet, but will be available in the next version.
- The IMDI Browser is a very flexible tool. A hierarchy of nodes will be created to suit your individual needs - you will even be able to create your own temporary hierarchies (with the help of the IMDI BCTreeBuilder). There are no restrictions on the format of your annotation files (ELAN, Media Tagger, Shoebox, CHAT, etc.), and only very few on the format of the digitized media files. Moreover, you can constantly update your files (provided you do not change their file names). This allows you to change annotations so as to reflect changes in your analysis, or to change IMDI files so as to enter, e.g., additional information that you did not judge to be relevant at first.

The IMDI Browser thus facilitates the access to and the work with your data. It is the responsibility of the corpus manager team (corpus.manager@mpi.nl) to help you integrate your data into the Browser: they develop a hierarchy in consultation with you, they link the

metadata information to the media and annotation files, and they digitize the media files. Your responsibility is it to provide them with the means to do their work efficiently. Essentially, this means to provide them with metadata information in form of the IMDI files.

PART II: GETTING STARTED

This part introduces you to the underlying structure of the IMDI Editor. It addresses the following points:

- the key structure, concepts and terminology of the IMDI Editor;
- features of the Editor that allow you to enter recurring information.

Note that part II is not yet concerned with the actual content of the various screens of the IMDI Editor. This is the topic of part III.

4 Basic Information

4.1The access policy

The metadata information that you enter into the IMDI Editor is meant, in principle, to be visible to the outside world via the Internet. Note that this does **not** concern the audio, video and annotation files as these will only be visible to you. And note also that the IMDI Editor itself allows for the possibility to hide certain types of information from the outside world. In part III, we show you how to suppress such information.

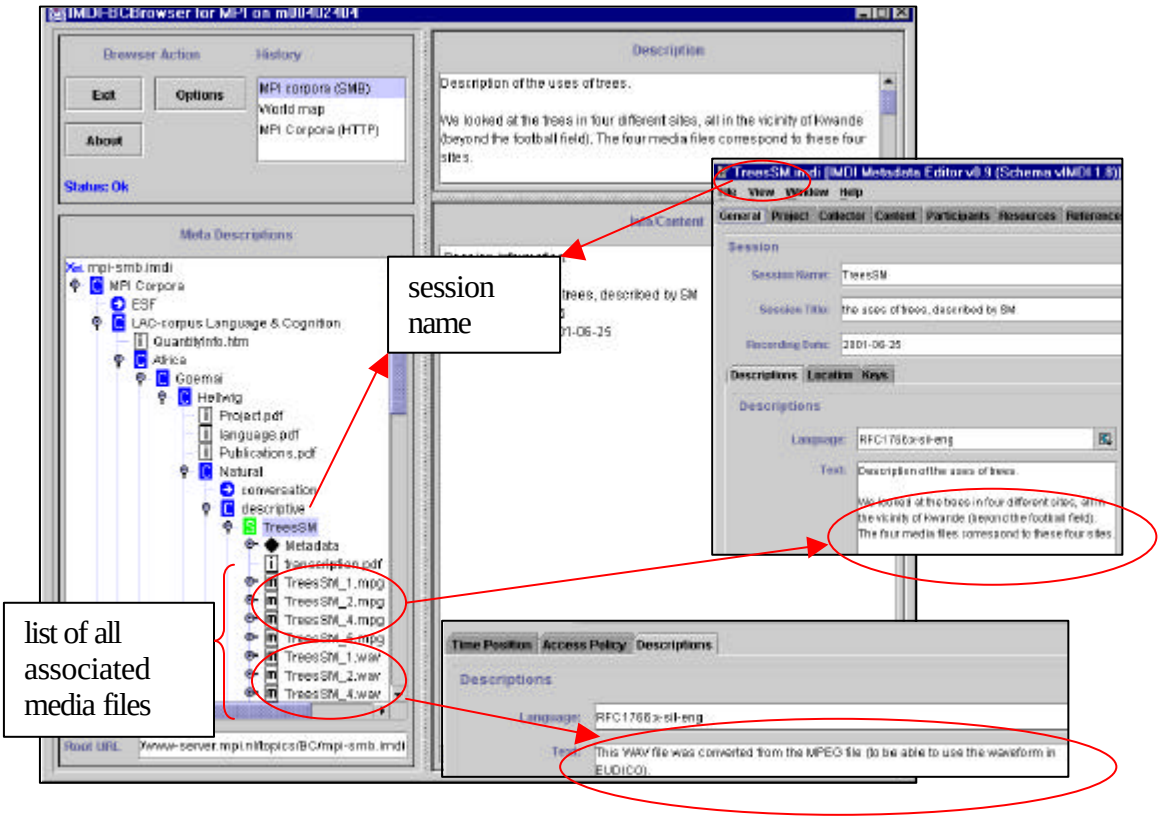
Two considerations lie behind the decision for making IMDI files available to the public: (a) the documentation of little known languages, and (b) a general policy (of the Max Planck society and of several scientific journals) relating to the transparency of the database on which you base your analysis. So please be aware that other people will have access to the IMDI files. Keep this in mind when you enter information into the Editor, and make sure that you exclude all sensitive information.

> **!** Note: We are aware that there exist old IMDI files that still contain sensitive information. If this is the case, please contact corpus.manager@mpi.nl, who will propose a solution.

4.2The session

The 'session' is the basic unit that is described through one single IMDI file (i.e., the file created with the IMDI Editor). It refers to one linguistic unit having the same overall content, the same set of participants, and the same location and time. For example, one elicitation session on the topic 'demonstratives', or one folktale, or one 'matching game'.

Note that the session does not need to correspond to a single media file and/or tape and/or annotation file. For example, in the following screenshot the session 'TreesSM' contains a number of separate media files.



The different media files in the screenshot above correspond to different subparts of the one session - they are included in one session because they have the same topic, the same speaker, the same location and the same date. And as such the one IMDI file **TreesSM.imdi** contains information about all subparts.

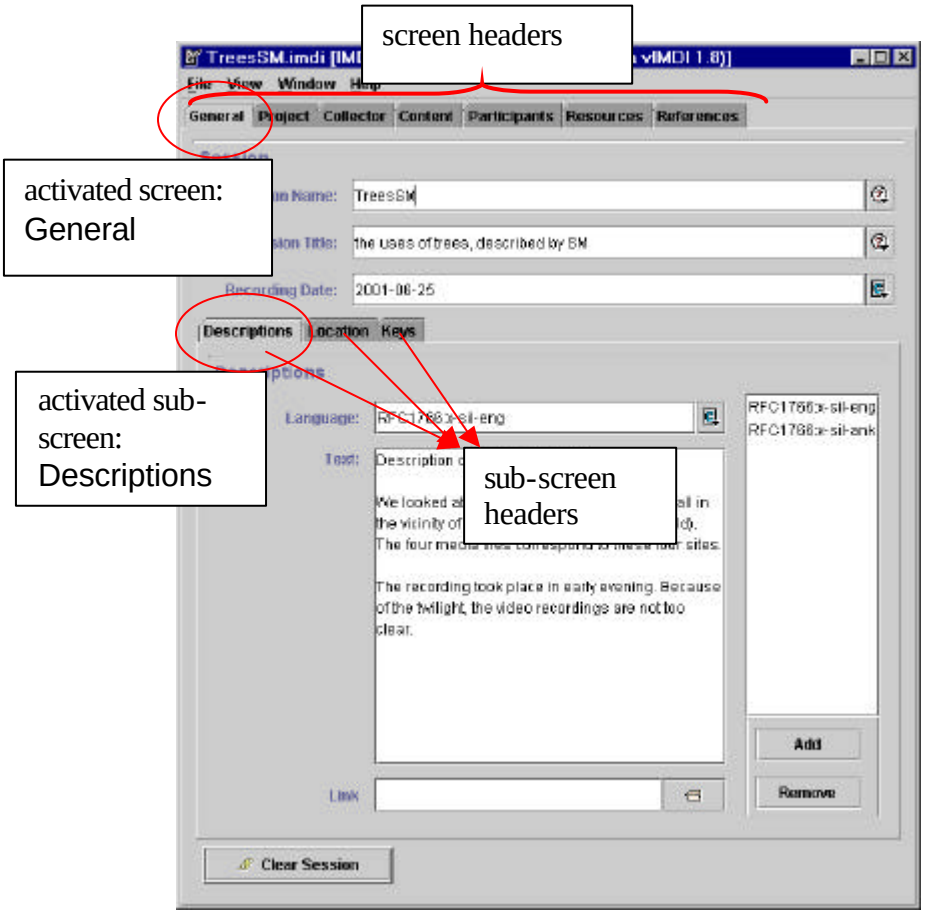
Conversely, there may be different sessions on one media file and/or tape. For example, one speaker first explained the uses of trees, and after that you recorded another speaker talking about politics. In this case, you would have to divide the tape into two sessions (one containing the 'trees' session, and the other one containing the 'politics' session), each of which corresponds to a different IMDI file.

One single IMDI file thus corresponds to one single session.

4.3The structure of the IMDI Editor

4.3.1 Screens

The IMDI Editor is organized around a number of screens, which are displayed like filing cards.

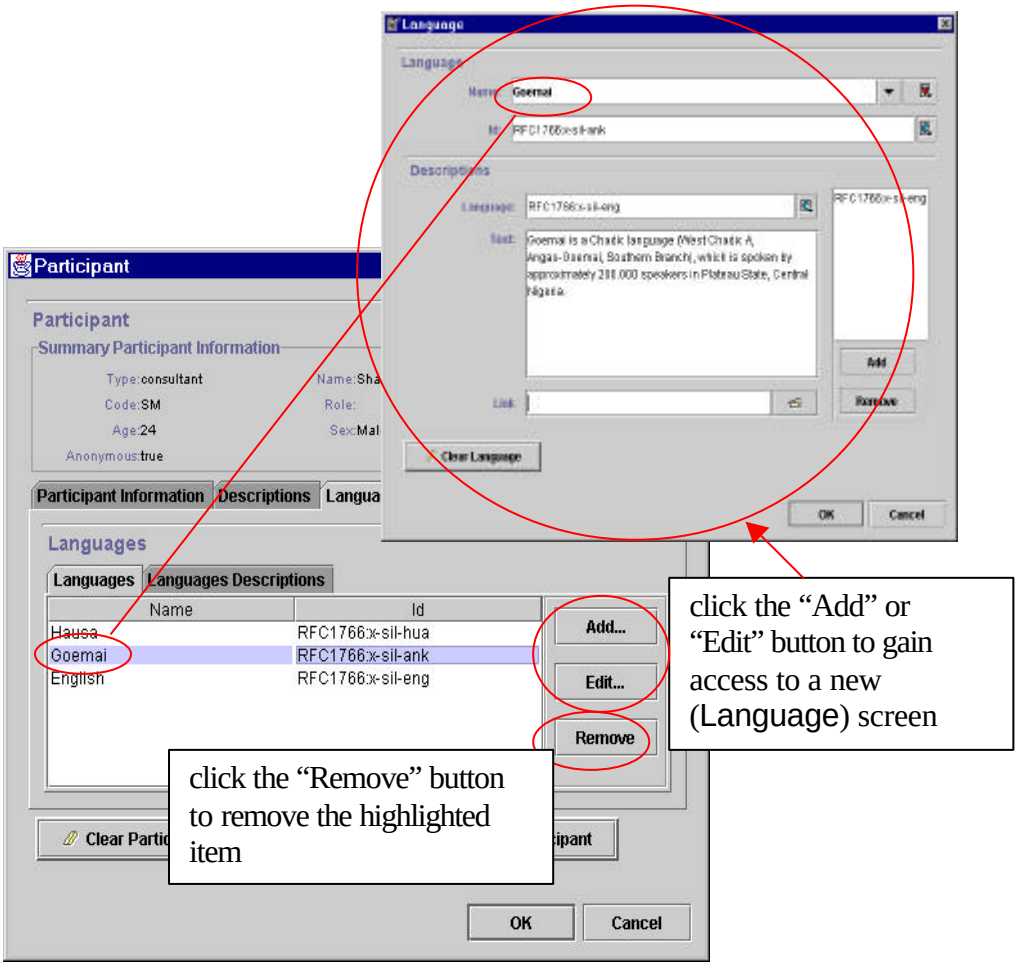


Click on a screen header to activate the corresponding screen. The following main screens are available:

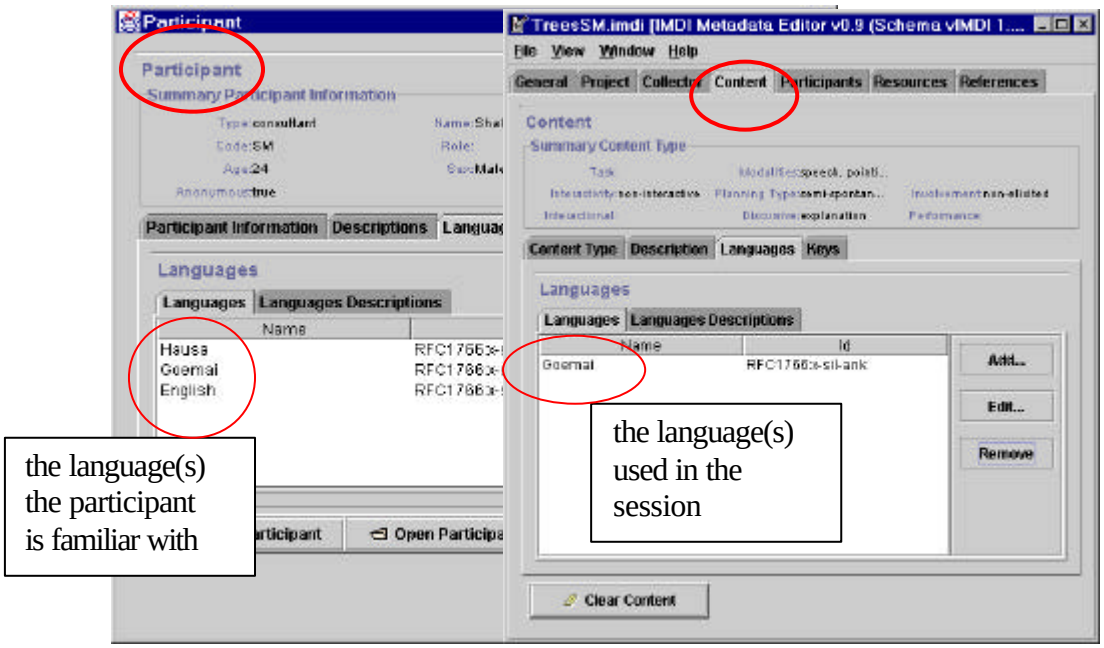
- General: general information about (a) the session (name, date, location) and (b) the IMDI file.
- Project: information about the larger project within which the data for the session was collected.
- Collector: information about the person who collected the session data, and about the person who is responsible for the collection of the session data.
- Content: information about the content of the session.
- Participants: information about (a) each participant in the session and (b) the interrelations among different participants of the session.
- Resources: information about (a) access rights, (b) the annotation and digitized media files, and (c) the original media tapes.
- References: cross-references to other sessions, fieldnotes, or publications, which are relevant to the content of the session.

These seven main screens thus bundle all the information available on an overall topic. In addition, some of the main screens give access to additional screens. The following two types are available:

- Sub-screens give access to additional fields and schemata relevant to the main screen (cf. the screenshot above).
- "Add" buttons (and their corresponding "Remove" and "Edit" buttons) allow you to enter multiple types of information of the same kind, e.g., one screen added for each language, e.g.:

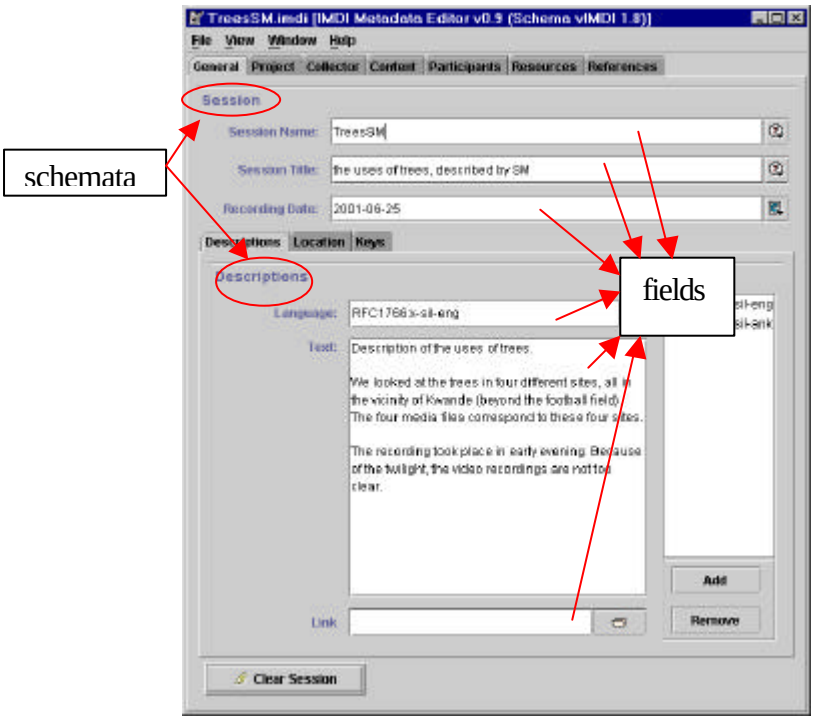


The important thing to remember from this structure is that all entered information is relevant to the main screen only. An example will illustrate this point: you will find that you are asked to provide information about 'languages' both on the Participant screen and on the Content screen. But the information you enter differs: in the former case, you enter the language(s) that participant X is familiar with, while in the latter case, you enter the language(s) used during the session.



4.3.2 Schemata and fields

Screens are made up of a number of units, which are referred to as 'schemata'. In addition, they contain a number of boxes into which the actual information is entered. These are referred to as 'fields', e.g.:

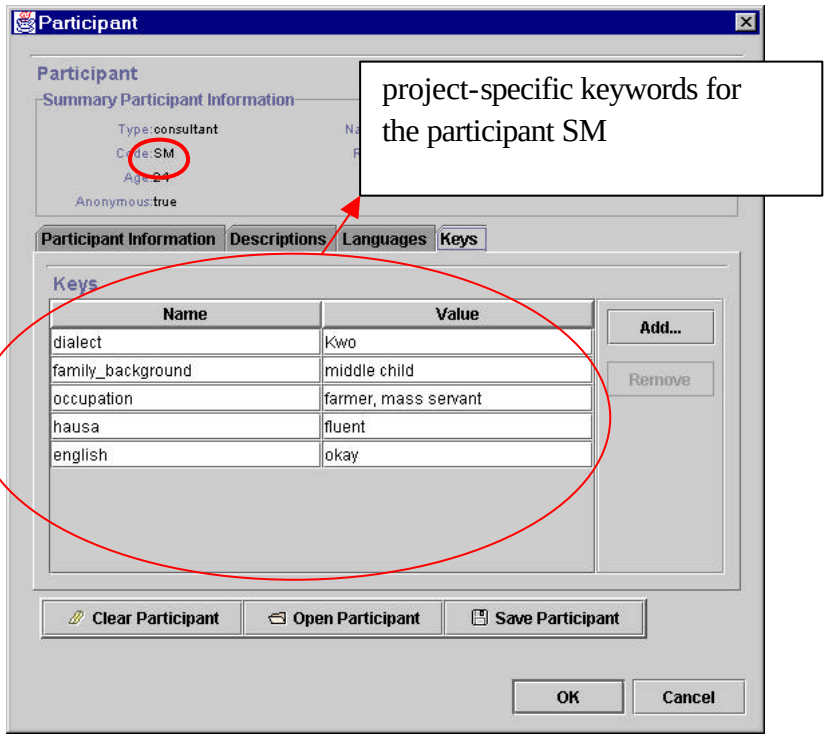


Most of the fields are more or less standardized (cf. "Appendix 3: Lists of recommended vocabularies"). The following options are available:

'open vocabulary' 	There are some recommendations, which can be selected from a pull-down menu. However, since this list is potentially endless, you will often have to type in a different content.
'open vocabulary lists' 	There are recommendations, which can be selected from a pull-down menu. This list is nearly exhaustive, and you will only occasionally have to type in a different content.
'closed controlled vocabularies' 	The content must be selected from the pull-down menu.

In all these cases, you can either choose the item directly from the pull-down menu. Or you can start typing, in which case the pull-down menu will automatically open to display the available options.

In addition to these standardized fields, the IMDI Editor allows for the possibility to enter user-defined information. Such information is entered into a Keys schema (cf. "Appendix 2: Screens and recurring schemata" on instructions of how to fill in a Keys schema), e.g.:



Like all other fields, keywords are searchable. Please keep in mind that these keywords are not standardized in any way: they are very often project-specific, and they are thus likely to be of relevance to you alone. We therefore recommend to enter the following kind of information into a Keys schema: project-specific information that is (a) not taken care of in the standardized fields (provided by the IMDI Editor), and that (b) you would want to search for. If such additional information is irrelevant for your purposes, feel free to leave a Keys schema blank.

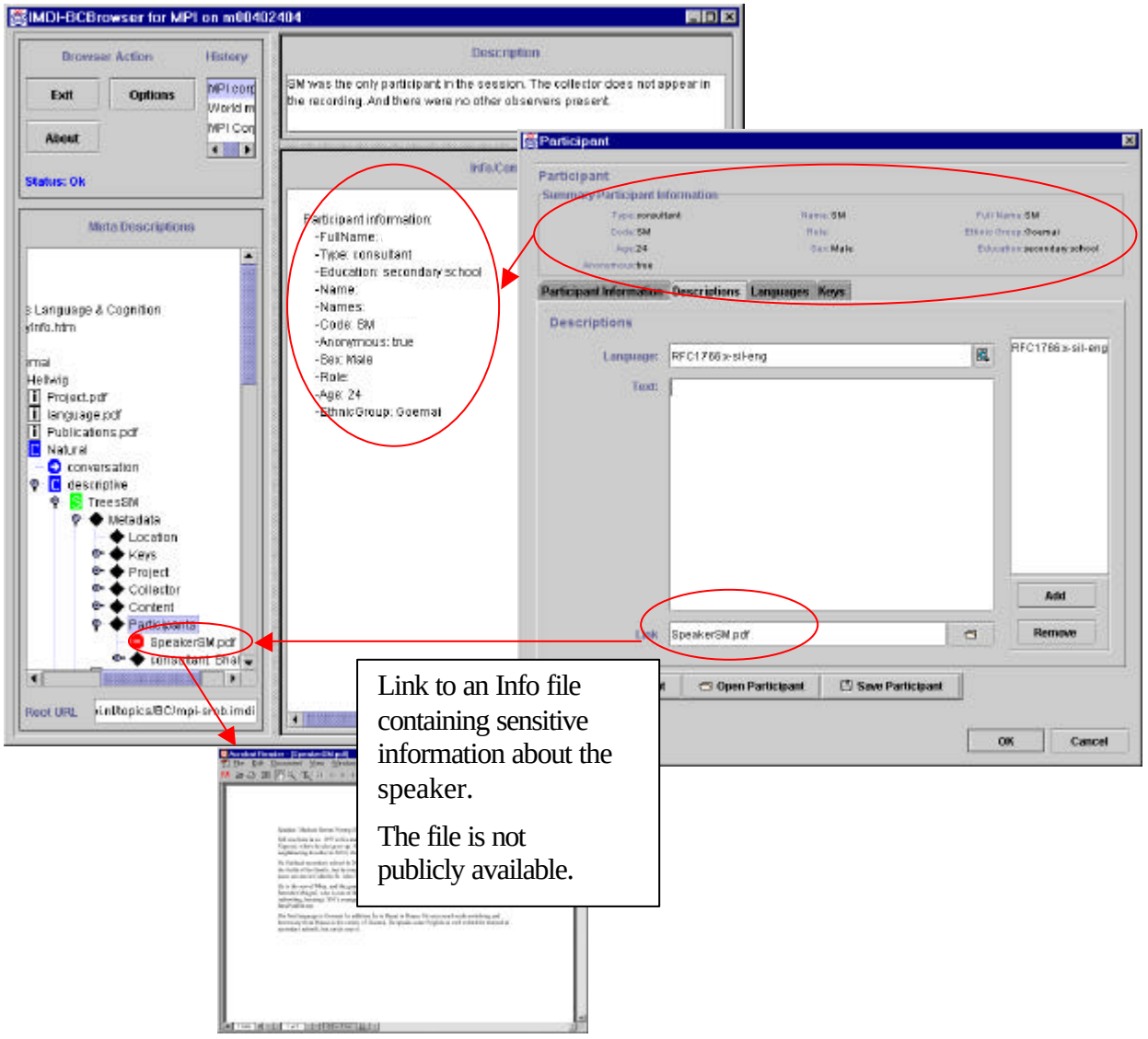
In addition to the searchable fields and keys, the IMDI Editor allows for the possibility to enter descriptive information. This is entered into the Descriptions schema, which you will find on every main screen (cf. "Appendix 2: Screens and recurring schemata" on instructions of how to fill in a Descriptions schema). A Descriptions schema should contain a prose description, which will serve as a reminder of the circumstances of data collection and which is displayed whenever you access the relevant session in the IMDI Browser, e.g.:

! Note: There are many different Descriptions schemata available, each of which is accessed at a different node in the IMDI Browser (e.g., under the session node, under the "content" node, under the "participant" node, etc.). Please familiarize yourself with the structure of the IMDI Editor, so that you will know which kind of information to enter into which Descriptions schema (cf. Part III).

! Note: When you transfer information from old metadata files (that were not written in an IMDI format) into IMDI files, it may be a useful strategy to simply copy and paste all the relevant information en bloc into the Editor's Descriptions schema instead of devising separate keywords. And although a Descriptions schema is not meant to be searched, it can be searched in an unsystematic way (e.g., search for all schemata that contain the words 'football field').

4.3.3 Info files

In addition to the information that you enter directly into the IMDI file, it is possible to create additional links to so-called Info files. Like the IMDI files, the Info files can be accessed through the IMDI Browser.

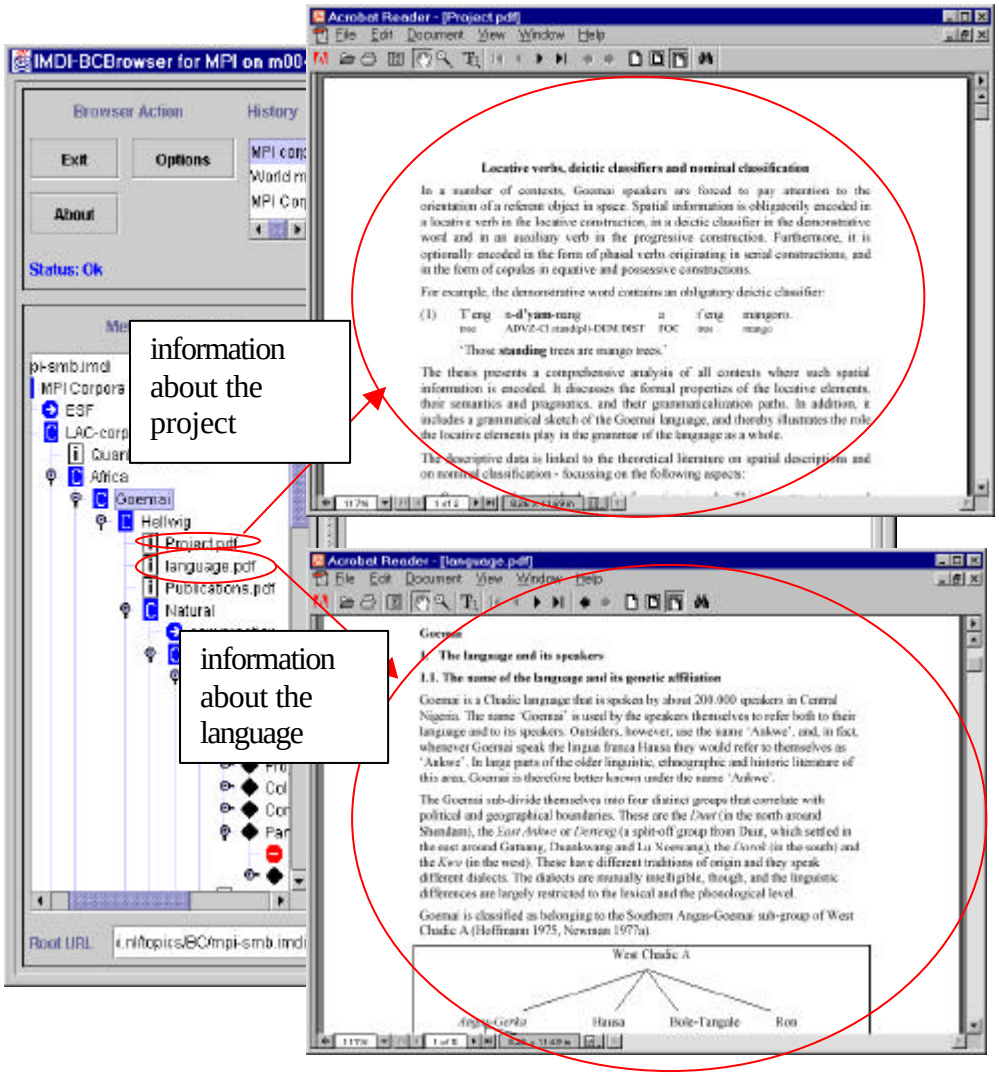


We suggest that you use Info files for storing the following kind of information:

- Any protocols that are connected to the session.
- Any digitized photos that are connected to the session (in ***.html** format). Note that it is currently not possible to integrate photos in any other way.
- Any kind of sensitive information that is connected to the session. By default, Info files are accessible to the outside world (like any IMDI file). But you can request that the content of an Info file should be hidden. In this case, please contact corpus.manager@mpi.nl.

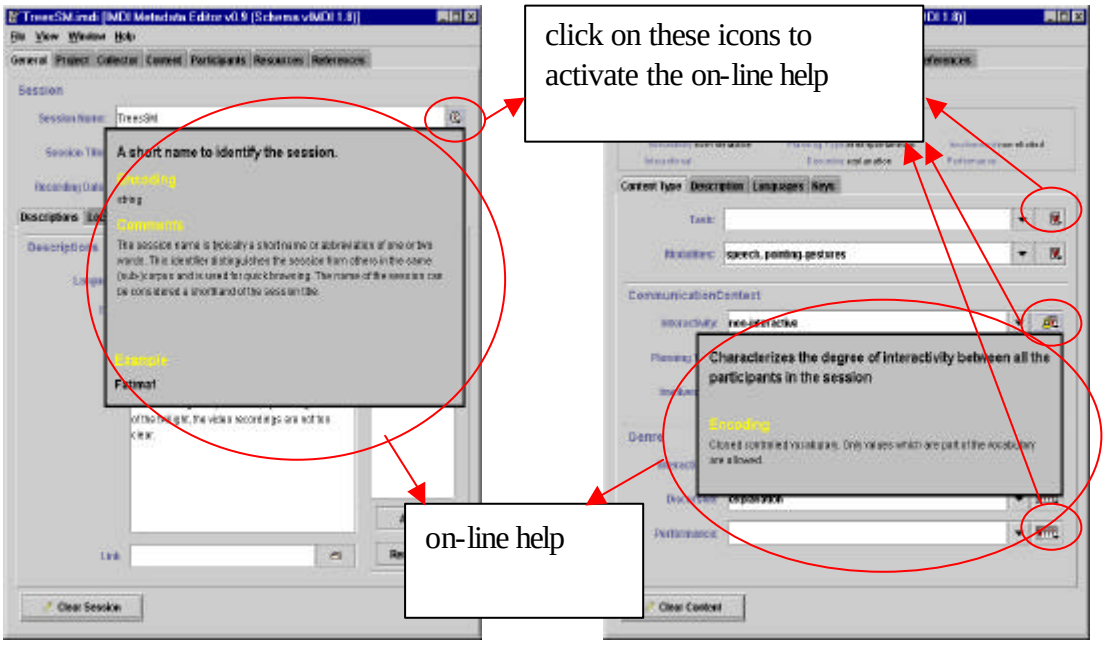
! Note: The Info file has to be saved as a ***.txt** file, as a ***.pdf** file or as an ***.html** file. Unfortunately, the IMDI Browser cannot display other kinds of files, e.g., Word files. If your need help with converting files into a ***.txt**, ***.pdf** or ***.html** format, please contact corpus.manager@mpi.nl.

! Note: Please use such Info files for session-specific information only. It is, of course, also possible to create additional Info files that are not relevant to the individual session, but to the corpus as a whole. Such information is very important, but it should not appear under the session node as this will multiply the information unnecessarily. We therefore ask you **not** to include links to such Info files in the IMDI Editor file. Instead, contact corpus.manager@mpi.nl, who will advise you about where to best store such kind of information. For example, in the following screenshot the background information about the project and the language are stored as ***.pdf** files directly under the first node in the corpus.



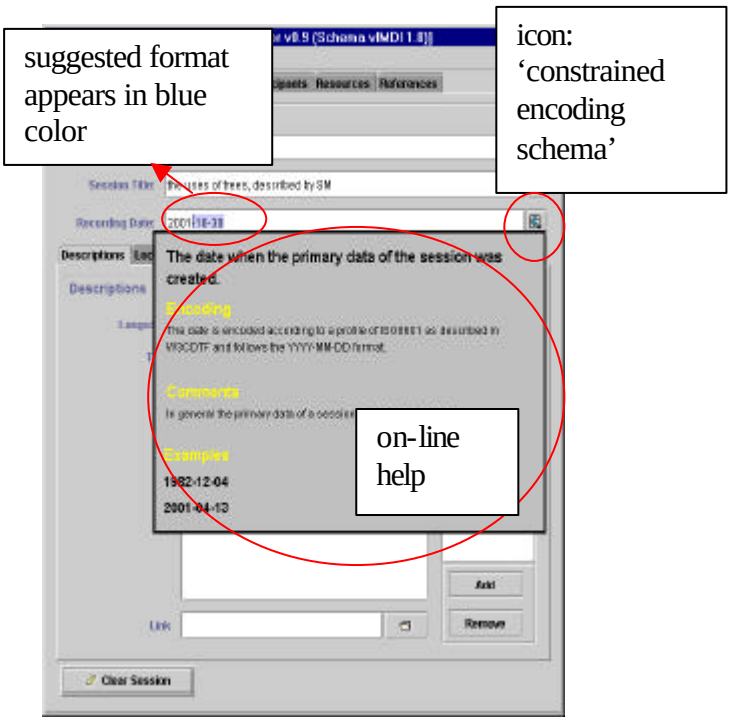
4.3.4 Online help

Most fields display an 'information' icon to their right. When you click on this icon, a textbox appears that gives information and examples about how to fill in the corresponding field.



! Note: The textbox only disappears again when you click into the corresponding field.

In addition, there are fields into which the data has to be entered in a standardized format - fields such as date, language (identifier), e-mail address, or (time) position. All these fields display the icon 'constrained encoding schema' to their right. Click on this icon to activate the on-line help. In addition, as soon as you start typing, the standardized format is displayed in the field (highlighted in blue color). You then have to type over this format.



5 Preliminaries: Preparing an IMDI file

Before you can enter information into an IMDI file, you will have to make some preparations first. We recommend the following strategies:

1. Decide on the unit that should be described by the IMDI file!
2. Think about the information that is likely to occur repeatedly!
3. Create master IMDI files and templates for recurring information!

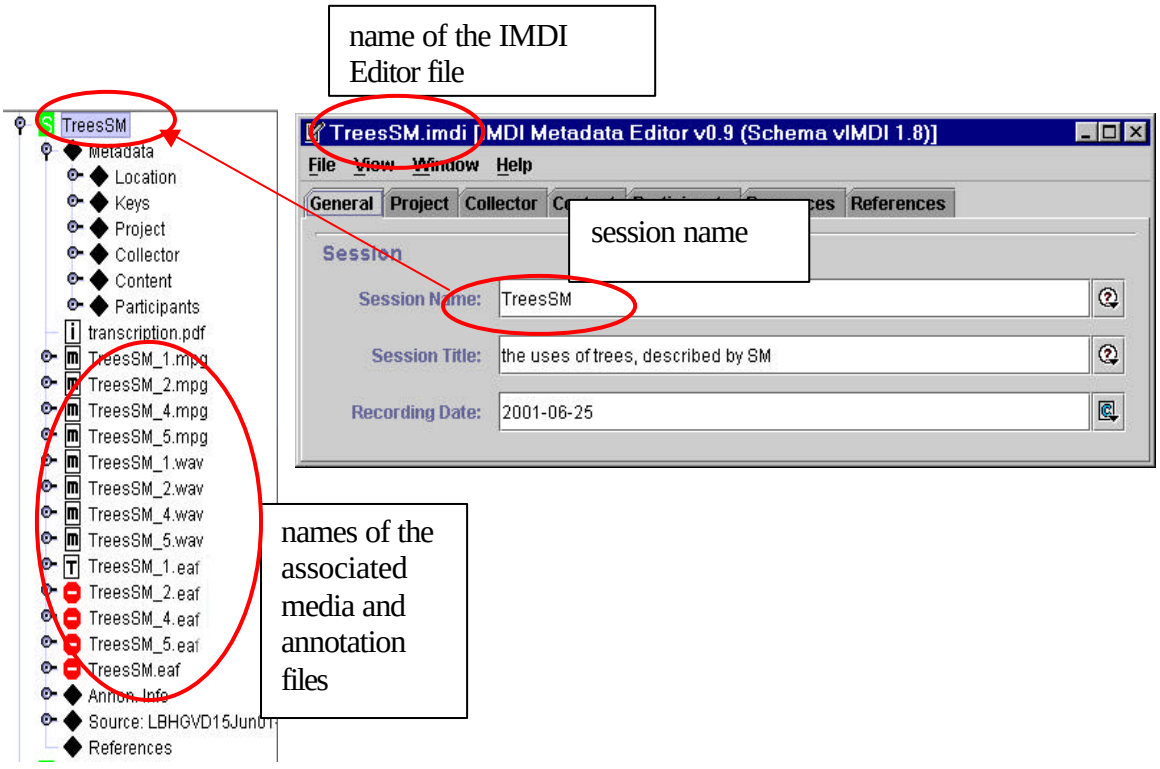
These three strategies will help you to create IMDI files with a minimum of time and effort.

5.1Decide on the unit that should be described by the IMDI file

The unit to be described is a 'session', which is a piece of data with a single overall topic (e.g., a folktale, an elicitation session on the topic 'demonstratives', a 'matching game'), collected at a specific date at a specific location and with a specific set of participants (cf. section 4.2 for further details).

If there is a corresponding audio or video tape, it needs to be digitized first (cf. "Appendix 5: Digitization Policy").

Once you have identified the session, create an IMDI file for it. Assign a name to that IMDI file. Please note that the name of the IMDI file should be the same as the name for the corresponding annotation file(s) and the corresponding media file(s) (if any). And it should be the name that is entered in the field Name on the General screen of the Editor.



! Note: If necessary, use cardinal numbers to differentiate between different annotation or media files belonging to the same session.

! Note: The file name has to be Unix compatible: do not use file names longer than 14 characters, do not use non-letter or non-number characters (except for the underscore: _), and do not use blank spaces.

5.2Think about the information that is likely to occur repeatedly

The IMDI Editor contains a large number of screens, schemata and fields. Its comprehensive nature may look threatening - however, there is much recurring information that needs to be supplied only once. Obviously, you are the only person who is able to decide, which type of information is session-specific and which is recurring, but some types are more likely to occur repeatedly than others. A list of these types is given in the following table, together with the corresponding screen(s) of the Editor.

Type of information	**Screen/ Sub-screen(s)**
The location where the data was collected	General/ Location
The larger project within which the data was collected	Project
The person who collected the data, and the person who is responsible for the collection of the data	Collector
The language(s)	Content/ Languages/ Language Participants/ Participant/ Languages/ Language
The participant(s)	Participants/ Participant
The access policy by which you grant/deny access to different parts of your data	Resources/ Anonymous Info/ Access Resources/ Media Files/ Media File/ Access Policy Resources/ Annotation Units/ Annotation Unit/ Access Policy Resources/ Sources/ Source/ Access Policy

Table (1): Recurring information

The type of information illustrated in the table above will either not change at all (possibly: the larger project) or it will only change within certain limits (e.g., the participant is likely to come from a set of recurring participants). It thus makes sense to supply this information only once and save it somewhere, so that you can re-use it from then on.

You have two options to enter recurring information:

1. A master IMDI file: This is used for all information that is relevant to all/most sessions (e.g., the project).
2. A template: This is used for information that recurs in many sessions, but that (a) does not occur in all sessions or that (b) does not occur in the same combinations in all sessions (e.g., different constellations of participants).

The following section shows you how to create and use such master files and templates.

5.3Create master IMDI files and templates

5.3.1 Master IMDI files

Master IMDI files are used for information that is relevant to all/most sessions. To create such a file, do the following:

1. Open the IMDI Editor.
2. Fill in all fields that will recur in all/most sessions.
3. Save the file under a dummy name like, e.g., 'master.imdi'

Every time you create a session IMDI file, do the following:

1. Open the IMDI Editor.
2. Open the master IMDI file, e.g. 'master.imdi'.
3. In the "File" menu, choose the option "Save As". Specify a new name for the IMDI file.

   ! Note: Please choose the session name as the name for the IMDI file (cf. section 5.1 above).

   All information that was contained in the master IMDI file is now also contained in the session IMDI file. Only the session-specific information needs to be added now.

! Note: You can always delete information from any IMDI file, either manually, or by making use of the option Clear …, which you find at the bottom of all screens and which automatically deletes all information on the relevant screen. It is thus possible to create master IMDI files even in those cases where some part of the information changes occasionally. For example, you might usually collect your data at the same location, but occasionally you move to a different village. In this case, it is still useful to enter the usual location into the master IMDI file, and to change it manually for those sessions where you collected the data somewhere else.

! Note: You can create as many master IMDI files as necessary. For example, you might always work with the same two 'collectors' - in this case, you can create two master files, one for each collector. It is also possible to create master files for the different participants. Note, however, that this is not always advisable as participants usually do

not occur in the same constellation. In this case, it might be better to only include the main consultant in the master file (as he/she is likely to occur in many sessions), and then (a) delete him/her whenever he/she does not take part in a session and (b) add the other participants by means of templates.

5.3.2 Templates

Templates are used for information that recurs in many sessions, but that (a) does not occur in all sessions or that (b) does not occur in the same combinations in all sessions.

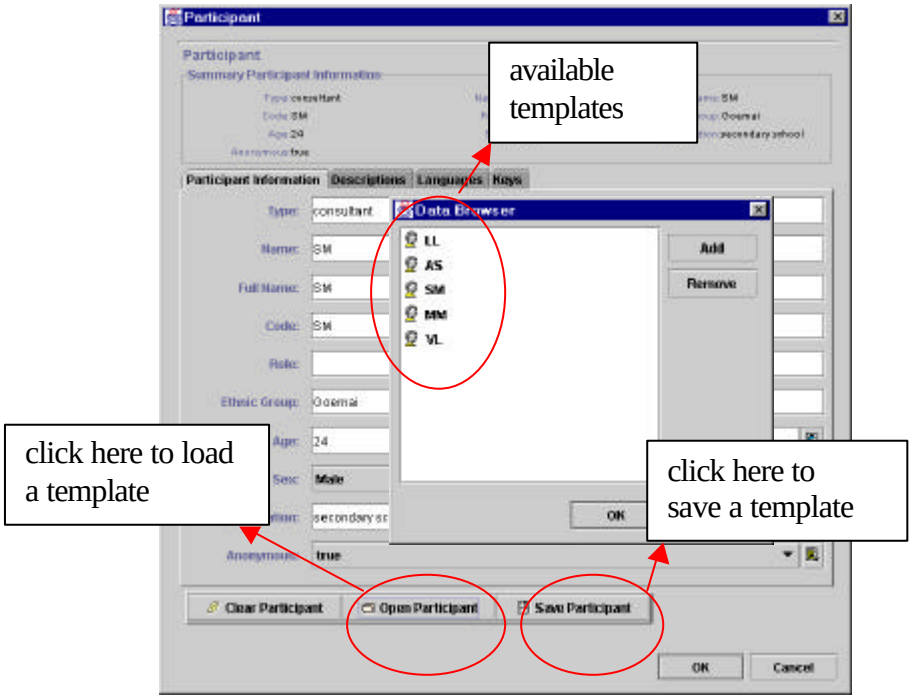
Templates are available for the following (sub-) screens:

- Project
- Collector
- Participant (accessed via the screen Participants)
- (Access) Policy (accessed via the four sub-screens under the screen Resources)

To create a template, do the following:

1. Open the IMDI Editor.
2. Fill in the relevant screen (i.e., the screen for which you want to create the template.
3. Click on the button “Save …”. The “Save” dialogue box appears. Browse to the folder where you want to save the template, and save it there under a descriptive name, e.g. ‘SpeakerSM.imdi’.

! Note: The file name has to be Unix compatible: do not use file names longer than 14 characters, do not use non-letter or non-number characters (except for the underscore: _), and do not use blank spaces.



Every time you want to re-use a template (e.g., whenever the participant appears again in a session), do the following:

1. Create/open the session IMDI file.
2. On the relevant screen, click on the button "Open …". The "Data Browser" dialogue box appears.
   (a) When the template is displayed in the dialogue box, click on it to highlight it, and then click "OK".
   (b) When the template is not displayed, click the "Add" button to browse to the folder that contains it and open it. It will then be displayed in the "Data Browser" dialogue box. Click on it to highlight it, and then click "OK".

   The IMDI Editor automatically fills in all fields.

! Note: Sometimes the "Data Browser" dialogue box erroneously displays a template that has already been deleted or moved to a different folder. If you attempt to load such a template, nothing will happen. Please remove it manually by using the button "Remove".

! Note: When you load a template, it may be necessary to change some of the information. For example, when you load the template 'speakerXY.imdi', it may be that

his/her role in the current session differs (e.g., she might occur in the role of 'mother' in one session, but in the role of 'wife' in another). In this case, you can simply change the information manually.

! Note: You can create templates in retrospect. For example, a participant who only appeared once in the beginning (so that there was no need to create a template for him/her), may become more important later on. In this case, you can open the old session file that contains the information about this participant, and save him/her as a template.

! Note: You can always change the information contained in a template. Do the following:

1. Load the template.
2. Make all the necessary changes manually.
3. Save the template (using the "Save …" button) under the same name as the old template. The old information is thereby deleted

5.3.3 Session-specific information

The two strategies outlined above will save you much time and effort when creating IMDI files. In the end, only very little session-specific information will be left that you have to enter manually. This is likely to be of the following kind:

- Information on the General screen, which contains very brief session information and IMDI file information. But even here, some information might be included in a master IMDI file (information on the location).
- Information on the Content (and on the References) screens. Both screens deal with the session-specific content; it is thus only to a very limited extent possible to create master files or templates (maybe the main language could be included in a master IMDI file).
- Information on the Participants screen - but only the information relating to the particular session-specific constellation of participants needs to be entered manually. Information on individual participants can be loaded from templates (with some manual changes relating to the type, role and age of the participant).
- Information on the Resources screen. This information needs to be entered manually for each session. But although many fields have to be filled in, much of it is semi-automatic in form of pull-down menus and lists of controlled vocabularies. The information is crucially needed for (a) linking the different files (media, annotation, and IMDI) to each other and (b) for digitizing the data from the original tape. Note that at least some of the information on the screen Annotation Units can probably be included in a master IMDI file (since you will probably make use of the same annotation units throughout).

Please keep in mind that elaborate descriptions about the whole corpus can be entered into separate Info files (background information about the participants, the language, the projects,

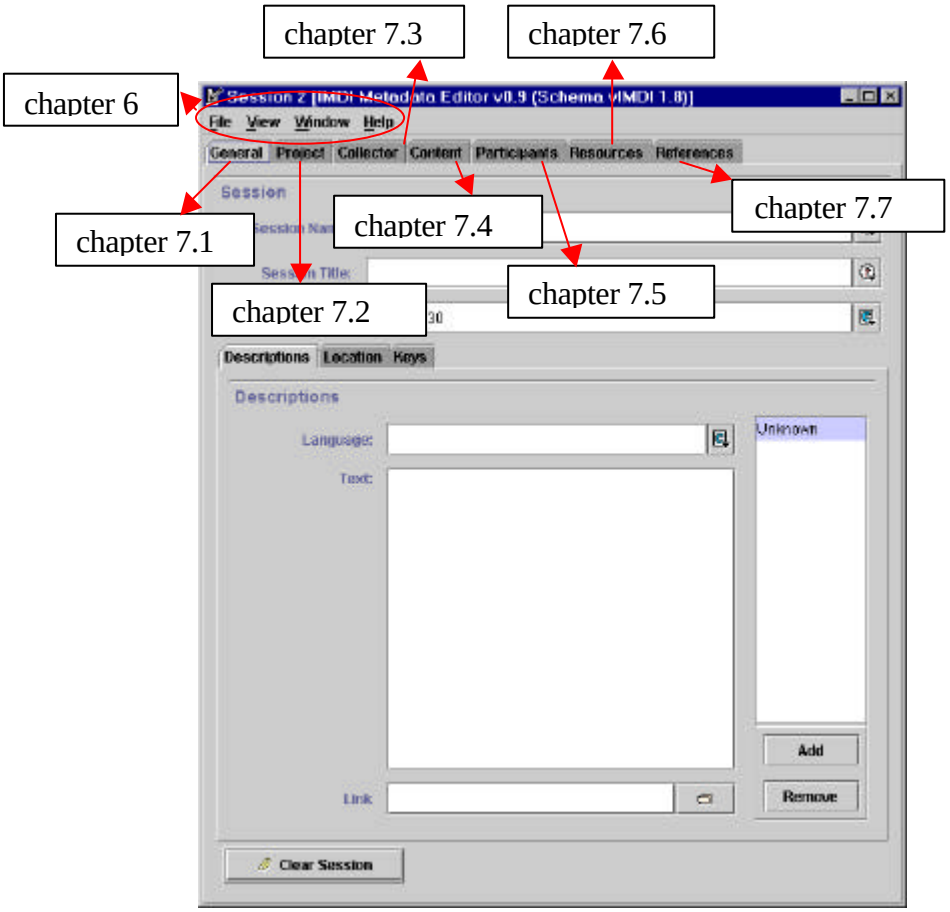
the transcription conventions, etc.). Such information need not (and should not) be repeated in every session (cf. section 4.3.3 for further information).

PART III: THE USER'S GUIDE TO THE IMDI EDITOR

This section guides you through using the IMDI Editor. It explains the purpose of each screen, schema and field, and it illustrates with the help of examples the kind of information to be entered there.

The IMDI Editor creates an IMDI file (***.imdi**). Every IMDI file contains the description of one session.

When you start the IMDI Editor, the following window appears:



! Note: After you have worked with the IMDI Editor for some time, it will become very slow, and there is the additional danger that files will not be saved anymore. It has even happened that saved files were destroyed. Currently, there is no other way around this problem than to (a) close all unused programs while working with the Editor and to (b) periodically close the Editor and restart it again.

6 Menu Items

The menu items are located at the top of the IMDI Editor window. The following items are available:

- "File": use this menu to create, open, save, or exit an IMDI file.
- "View": please ignore this menu item.
- "Window": use this menu to jump between open IMDI files.
- "Help": use this menu to read about the copyright and version information.

Some of the menu items occur with an underlined letter. Such items can be accessed through the shortcut key ALT plus the underlined letter, e.g., the "File" menu can be accessed through pressing ALT+F.

"File" Menu

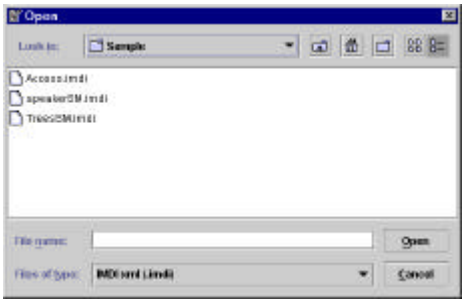
The "File" menu is used to create, open, save, or exit an IMDI file. It includes the following options:

"New Session"

Click on "New Session" to open another IMDI Editor window.

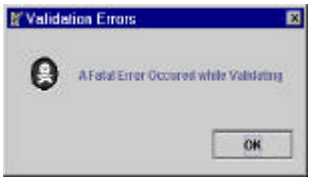
"Open"

Click on "Open" to open an existing IMDI file. The "Open" window appears:



Browse to the folder that contains the IMDI file and double-click on it to open it. A second IMDI Editor window containing this file appears.

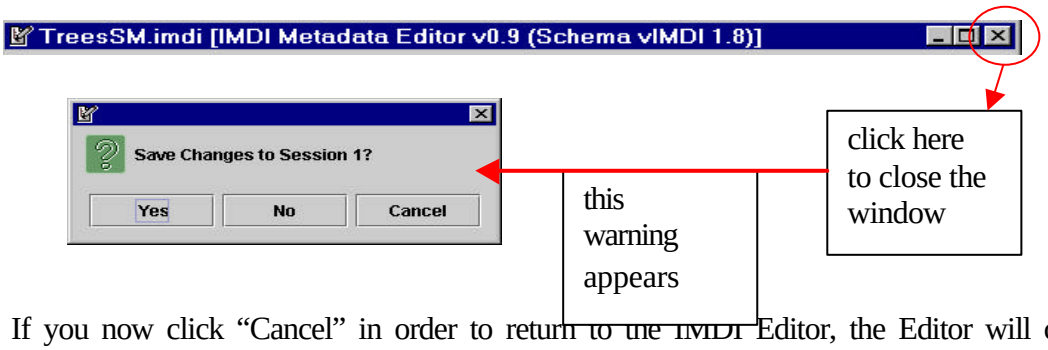
! Note: You can only open files of the IMDI Editor format. If you try to open a different file, the following two warnings appear:



"Close"

Click on "Close" to close he current IMDI file. Note that all other currently open IMDI files will remain open.

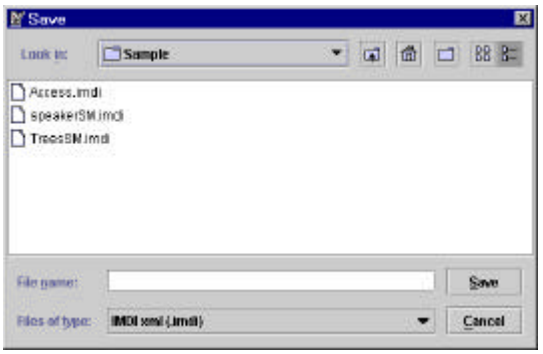
! Note: You can also close an IMDI Editor window through clicking on the cross icon in the upper right corner. In this case the following warning is displayed:



If you now click "Cancel" in order to return to the IMDI Editor, the Editor will close without a warning, i.e., you will loose all unsaved changes. Please do not make use of this option.

"Save"

Click on "Save" to save the current IMDI file. When you save a file for the first time, the "Save" window appears:



Assign a name to the IMDI file. Click on "Save". It is now saved.

! Note: Please remember that the name of the IMDI file should be the same as the name for the corresponding annotation file(s) and the corresponding media file(s) (if any). And it should be the name that is entered in the field Name on the General screen of the Editor (cf. section 4.2).

! Note: The file name has to be Unix compatible: do not use file names longer than 14 characters, do not use non-letter or non-number characters (except for the underscore: _), and do not use blank spaces.

! Note: When you save an IMDI file, the IMDI Editor will check if the corresponding folder contains a file **imdi.xsd**. If not, it will automatically create such a file.

"Save As"

Click on "Save As" to save the current file under a different file name. You are then asked to assign a new name to that file (cf. the explanation under **"Save"** above).

! Note: This option is especially relevant when you work with master IMDI files (cf. section 5.3.1).

"Save All"

Click on "Save All" to simultaneously save all open IMDI files.

"Revert to Saved"

Click on "Revert to Saved" to return to the last saved version of the current IMDI file.

! Note: This option will automatically delete all changes that you have made since last saving the file. The following warning will appear:

"Recent Files"

Go to "Recent Files" to access an IMDI file that you have recently worked with. A submenu appears showing the most recent files that were saved. Click on the file that you want to open. A second IMDI Editor window containing this file appears.

"Exit"

Click on "Exit" to exit the IMDI Editor window. All remaining IMDI files will be automatically closed.

"View" Menu

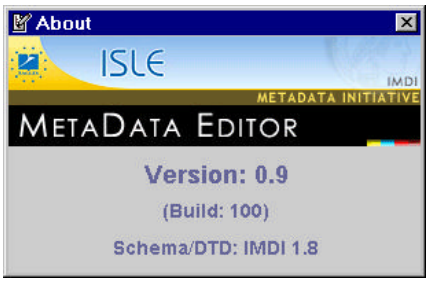
Please ignore this menu item.

"Window" Menu

The "Window" menu is used to jump between open IMDI files. Go to "Window" to see all files that are currently open. Click on the name of the file that you want to jump to. This will become the active file.

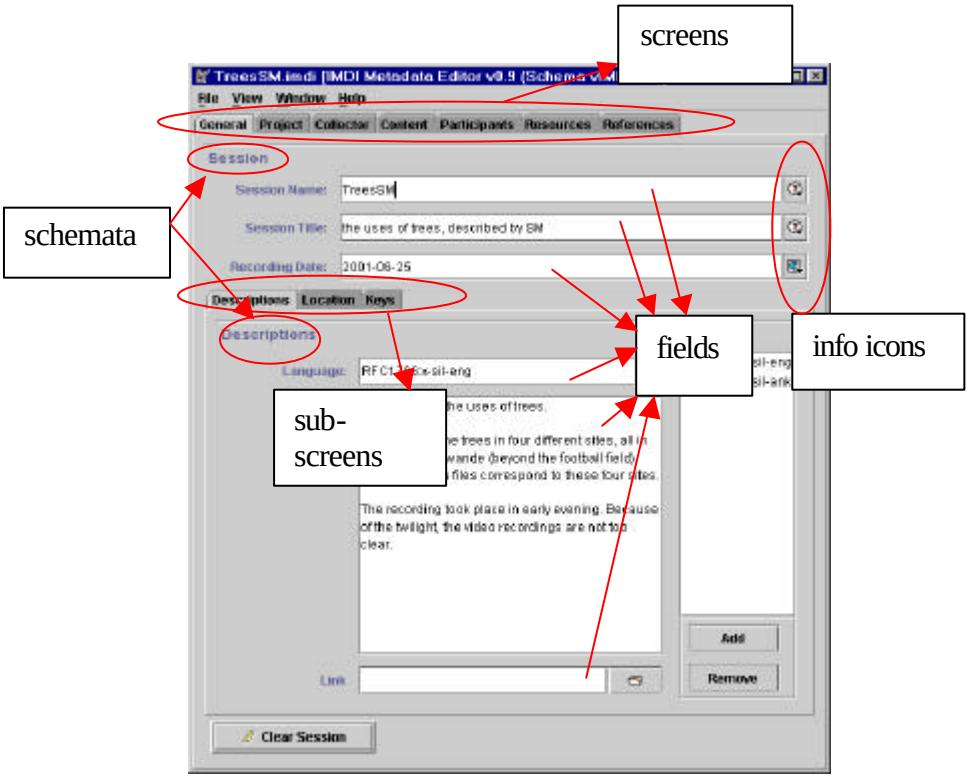
"Help" Menu

The "Help" menu displays the copyright and version information. Go to "Help", and then click on "About …" to view this information:



7 Screens

The IMDI Editor contains a number of screens and schemata with fields to be filled in.



You have two options for navigating through the IMDI Editor window:

(1) You can use the mouse.

- Click with the mouse button somewhere in the field where you want to enter the information.
- Click with the mouse button on a (sub-) screen header to activate the corresponding (sub-) screen.
- Click with the mouse button on any icon to display its corresponding textbox.

(2) You can use the following shortcut keys.

- Press TAB to move to the next field, icon or sub-screen header.
- Press SHIFT+TAB to move to the previous field, icon or (sub-) screen header.
- When an icon is highlighted, press SPACE to display its corresponding textbox.
- Press the left or right arrow key to move from screen to screen header, or from sub-screen to sub-screen header.

In the following sections, instructions and examples are given of how to fill in the various screens, schemata and fields. In addition, it is shown how the information entered into the IMDI Editor will be displayed in the IMDI Browser.

When entering information into the IMDI Editor, please keep the following three points in mind:

(1) You do not have to fill in every field!

When the IMDI Editor was created, it was attempted to include features that would be useful to all kinds of data and purposes - as a result, a lot of eventualities are covered that may be simply irrelevant for your individual purposes. In this case, feel free to leave fields blank.

And also keep the following in mind: Obviously, it would be nice to have elaborate descriptions and extensive keyword lists available for each session. But you have to weigh this against the time and effort it will take to enter this information. In the end, you will have to make this decision yourself. Please take the following considerations into account:

- Which parameters would you want to search for? Make sure that this information is entered into standardized fields or into user-defined Key schemata.
- How much descriptive information do you need to remind yourself about the circumstances of the session? This will determine how extensive your descriptions will be in the end.

(2) All information entered on a screen is only relevant to that screen!

This point is especially important for a recurring schema such as Descriptions, Keys, Language, or Access. They occur on various screens, and although they always look the same, they, in fact, ask for different information - information that is relevant to the corresponding main screen only. So, when you are asked to enter yet another description, you can safely assume that a different type of information is required this time.

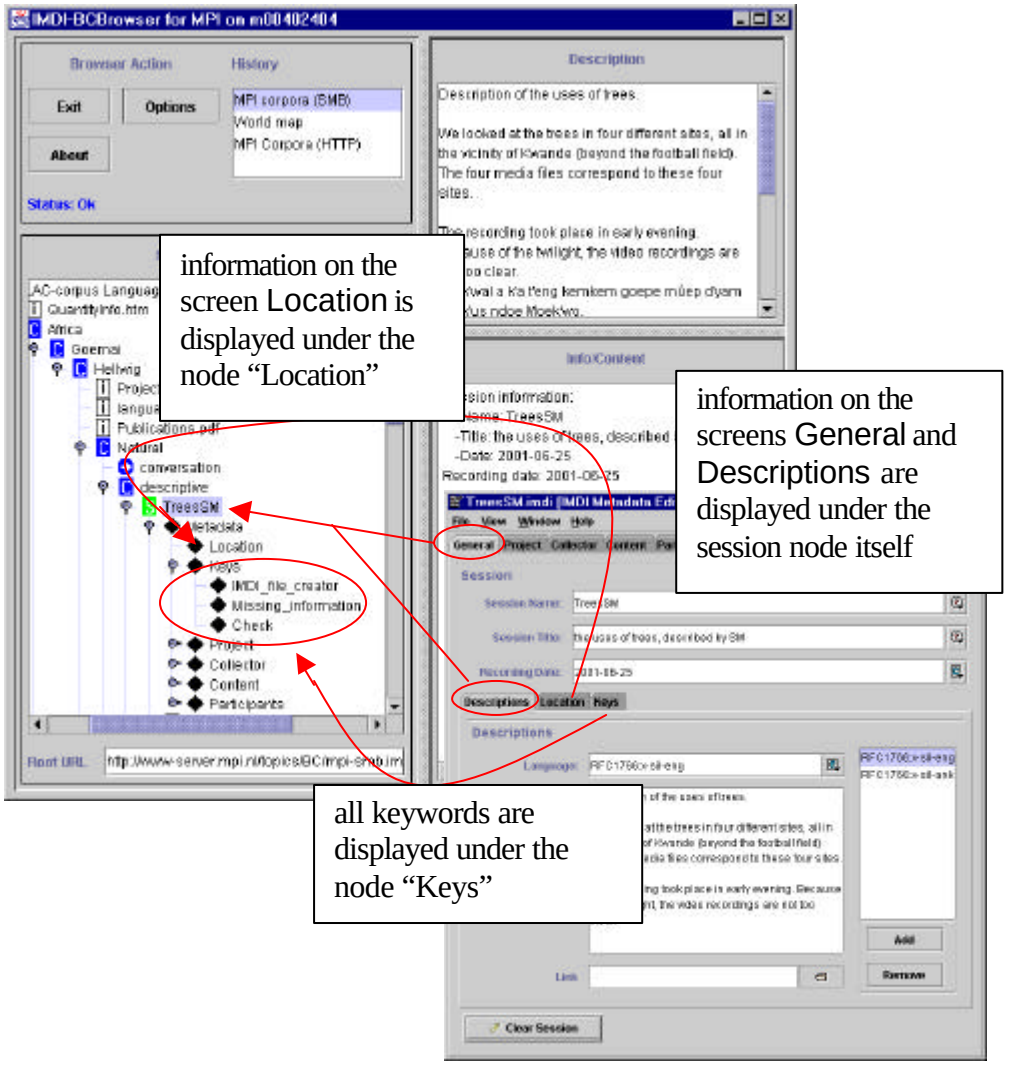
(3) Remember to make use of master IMDI files and templates to enter all recurring information (cf. section 5.3)!

7.1General

This screen contains general information about (a) the session (name, date, location) and (b) the IMDI file. The following schemata and sub-screens are available:

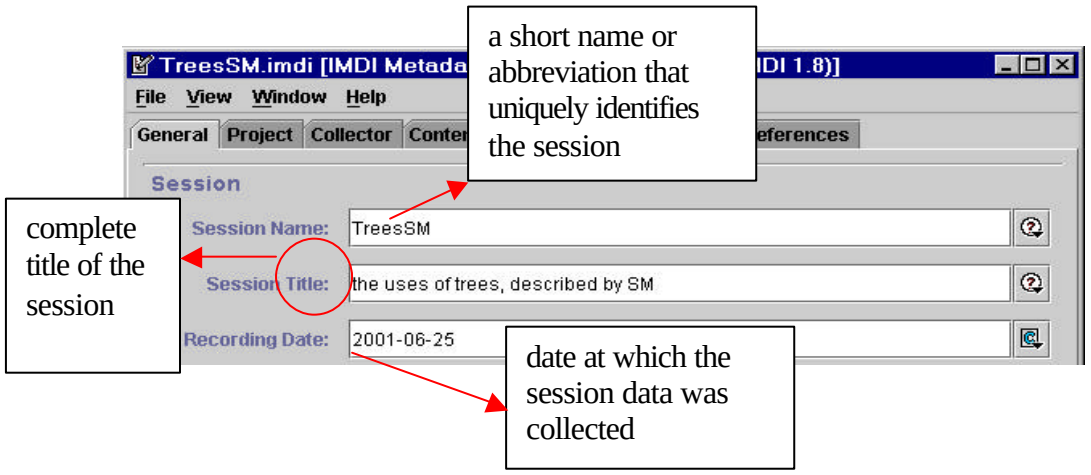
Session schema	Name, title and recording date of the session.
Descriptions sub-screen	Description of the circumstances under which the data for the session was collected.
Location sub-screen	Information about the location at which the data for the session was collected.
Keys sub-screen	Keywords that are relevant to (a) either the collection of the data or (b) to the creation of the IMDI file.

The information that you enter on these four screens is displayed in the IMDI Browser as follows:



General: Session

This schema contains the name, title and recording date of the session. For example:



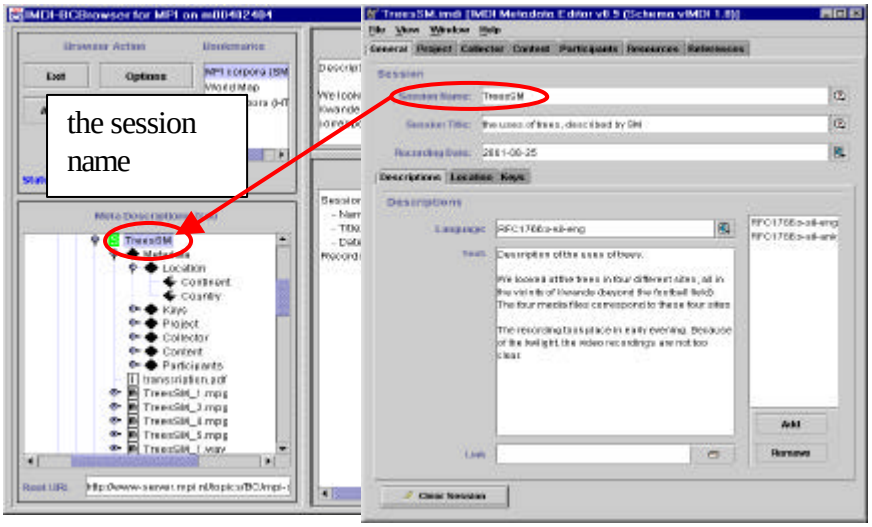
Name

A short name or abbreviation that uniquely identifies the session. It should be identical to the name of the IMDI file, the annotation file(s) and the media file(s) (cf. section 5.1).

! Note: This field is obligatory.

! Note: The name has to be Unix compatible: do not use names longer than 14 characters, do not use non-letter or non-number characters (except for the underscore: _), and do not use blank spaces.

! Note: The Name is displayed as the session name in the IMDI Browser. It should therefore be descriptive enough to remind you about the session's content.



Title

The complete title of the session. Usually, it is the spelled out version of the abbreviated Name.

Recording Date

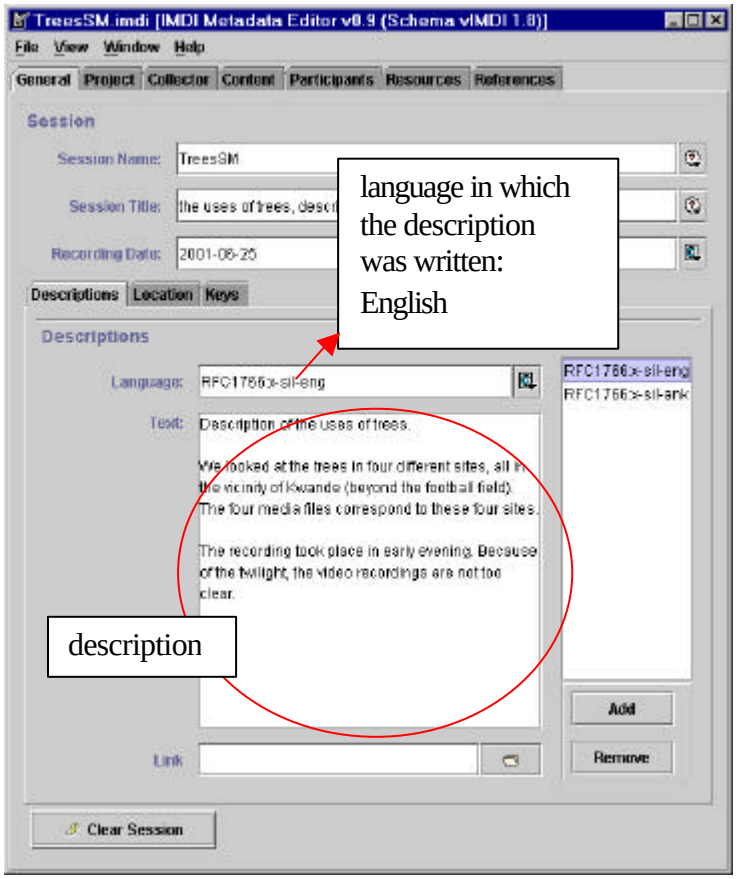
The date at which the session data was collected. Once you start typing a date, the IMDI Editor will automatically display the format (highlighted in blue color). Type over this format, to enter the date.

> **!** Note: Please enter the date in the following format: **YYYY-MM-DD**, e.g. 2000-12-30.

> **!** Note: You have to enter a complete date. If you do not know the exact date, we suggest to enter an approximate date. You can then make use of the Descriptions schema to remind you that it is only an approximation.

Descriptions

This sub-screen contains a description of the circumstances under which the data for the session was collected. It should **not** contain an elaborate description of the content (reserve such information for the Content Screen), but it can contain a brief reminder of its content, e.g.:

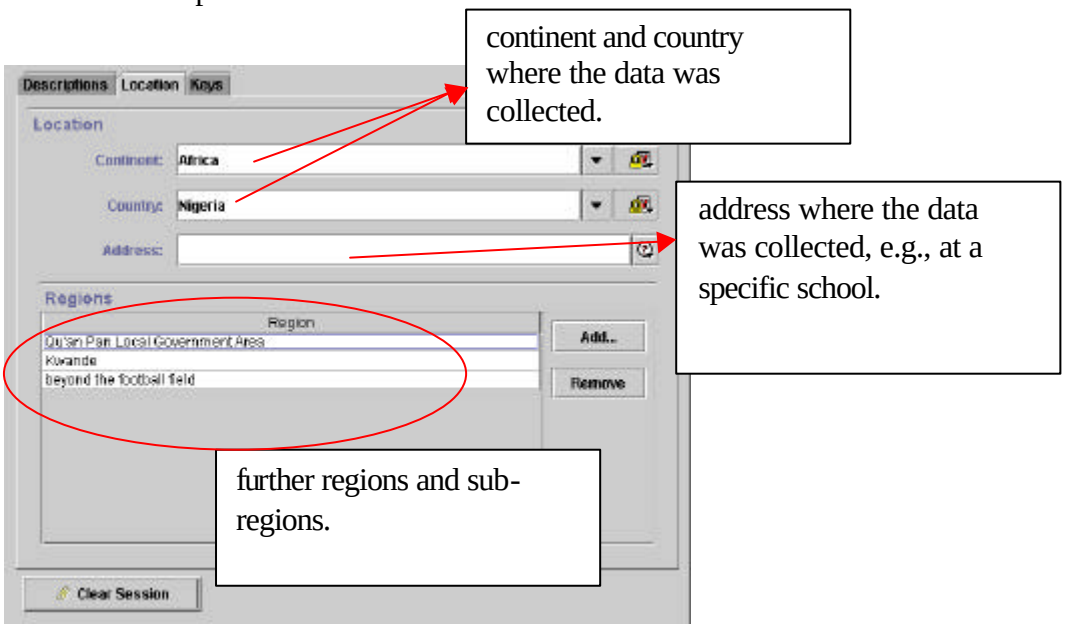


> **!** Note: The Descriptions schema on the screen General is the schema that is displayed when you click on the session node in the IMDI Browser. It should therefore contain all information that you would need in order to quickly remind you what this session was about.

Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

Location

This sub-screen contains information about the location at which the data for the session was collected. For example:



Continent

The continent where the data for the session was collected. You can either choose the continent from the pull-down menu. Or you can start typing, in which case the pull-down menu will automatically open to display the available options.

Country

The country where the data for the session was collected. You can either choose the country from the pull-down menu. Or you can start typing, in which case the pull-down menu will automatically open to display the available options.

Address

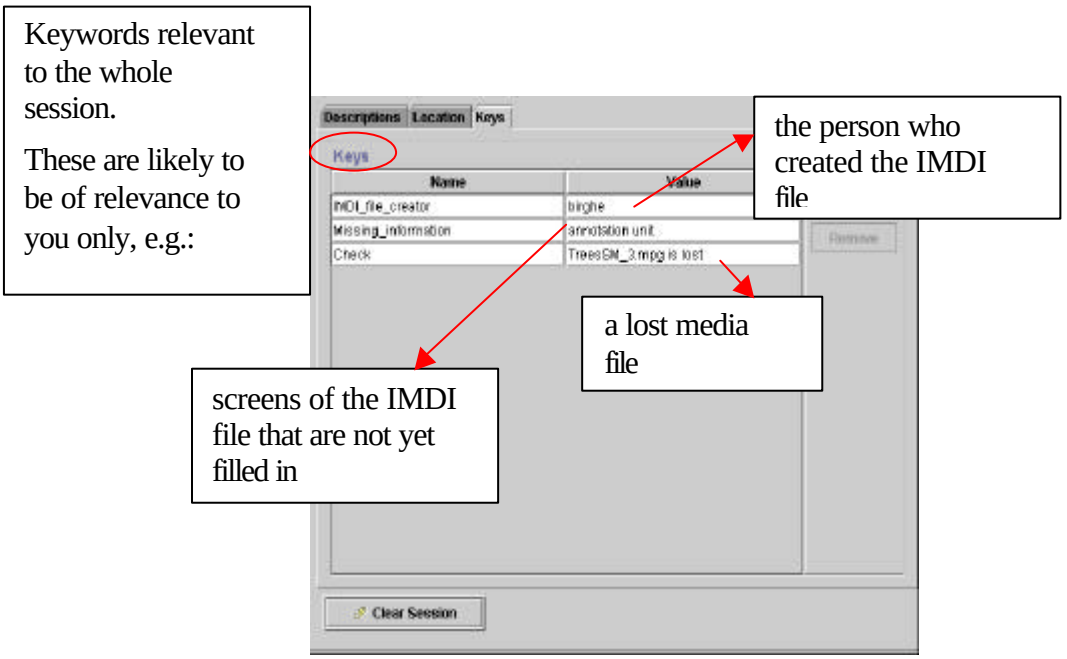
The address where the data for the session was collected (e.g., at a specific school or another institution).

Region

The region (province, town, suburb, etc.) where the data for the session was collected. Use the "Add" button to add another region, and use the "Remove" button to remove the highlighted region.

Keys

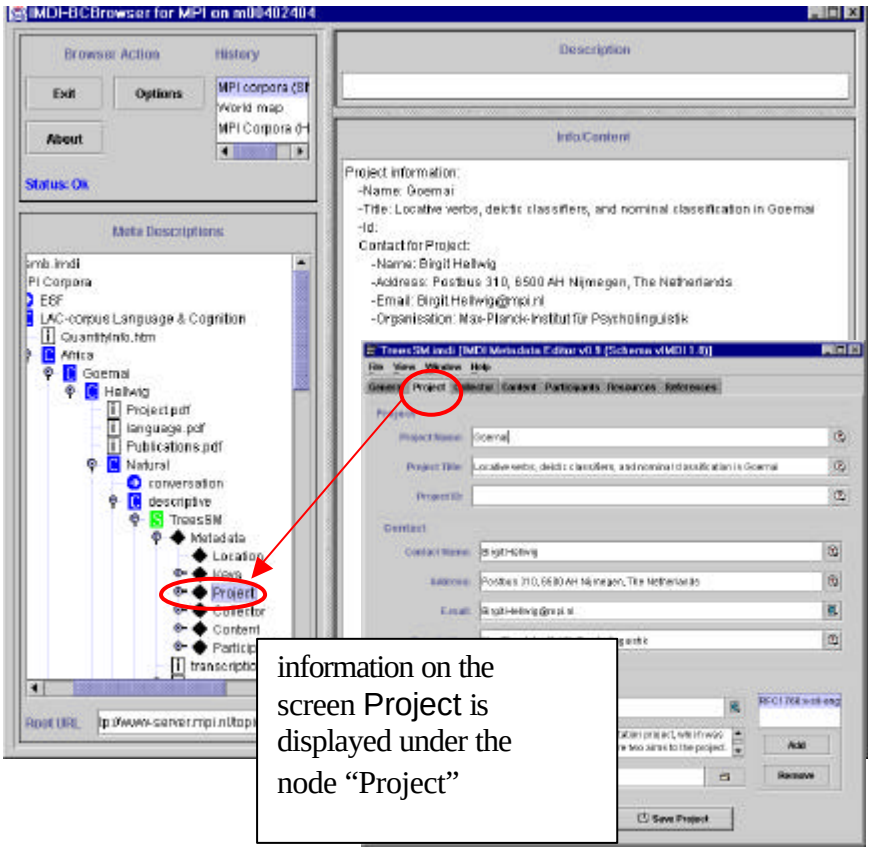
This sub-screen contains keywords that are relevant to (a) either the collection of the data or (b) to the creation of the IMDI file. For example:



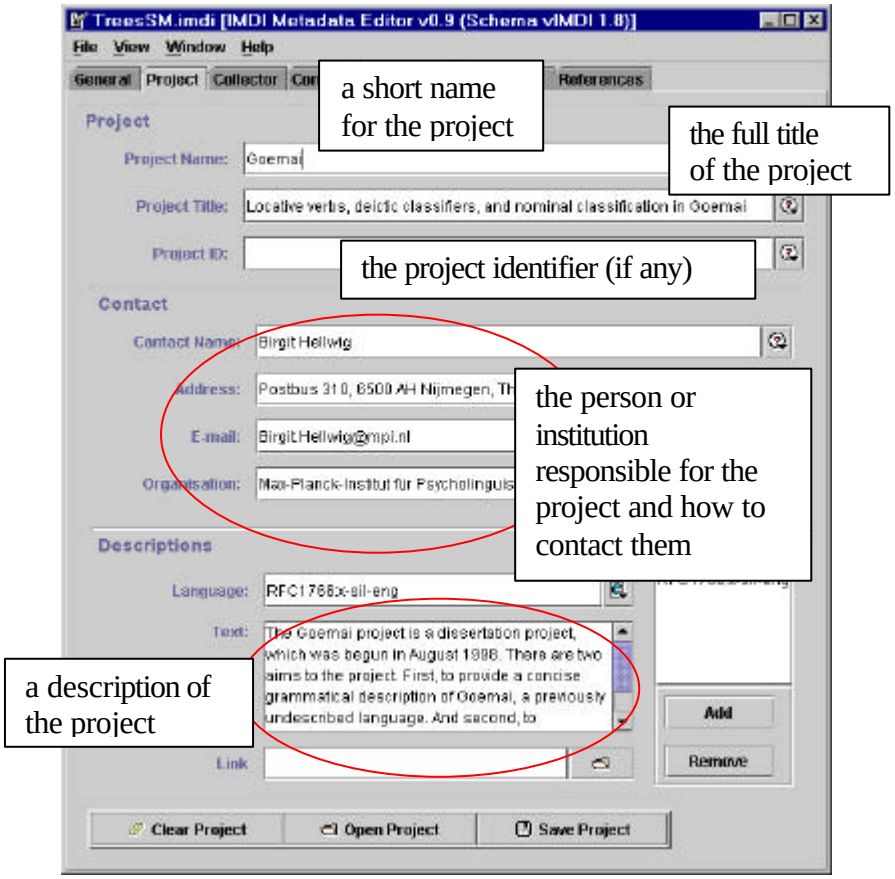
Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Keys schema.

7.2Project

When the data for the session was collected within a larger project, this screen contains all the information about the project. The information that you enter on this screen is displayed in the IMDI Browser as follows:



Enter the information into the IMDI Editor as follows:



Project

- Name
  A short name or abbreviation that uniquely identifies the project.
- Project Title
  The full title of the project.
- Project ID
  A unique identifier for the project (if there is any), e.g. "IST-1999-10651".

Contact

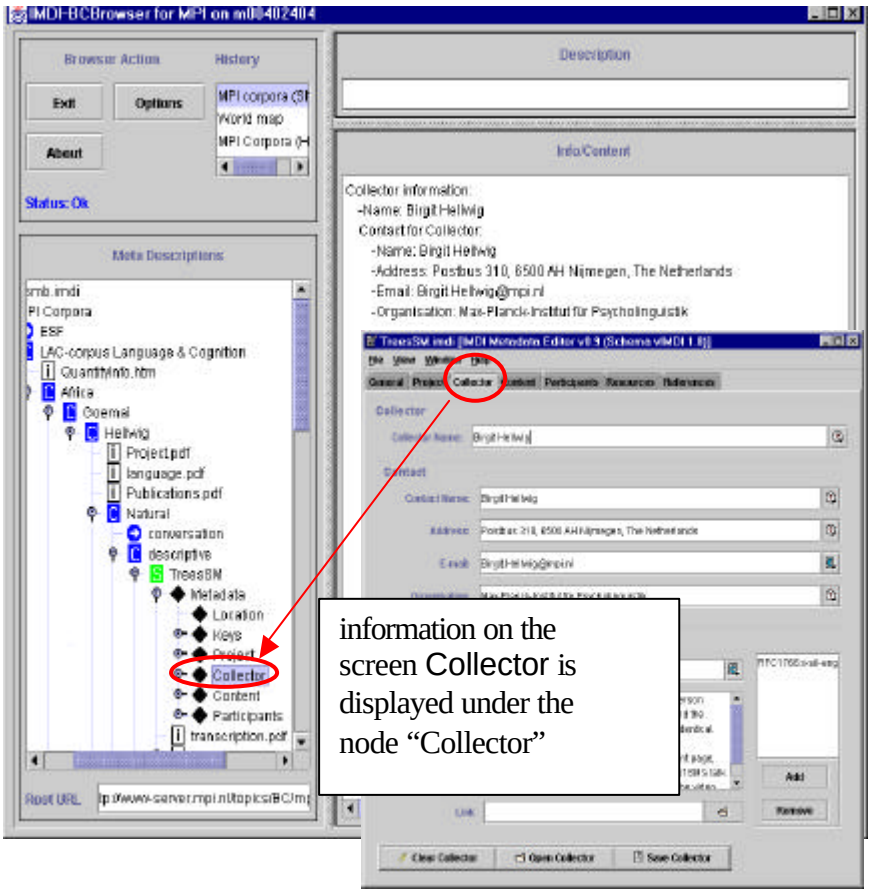
The contact information about the person or institution responsible for the project (the name, the address, the e-mail address and the organization he/she belongs to).

Descriptions

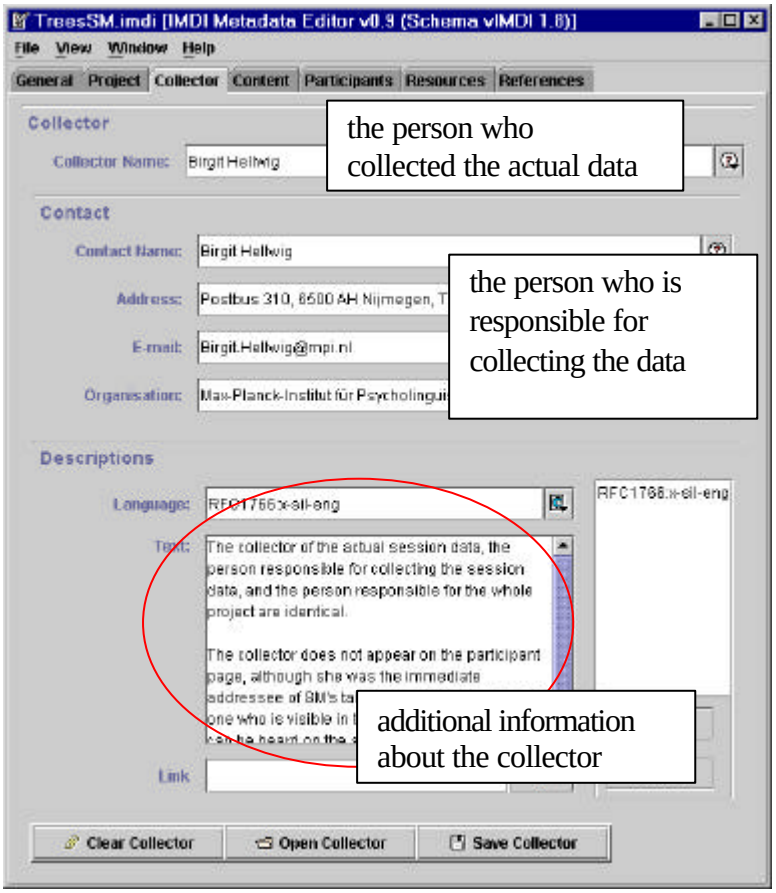
This schema contains a description of the scope and the goals of the project. Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

7.3Collector

This screen contains information about the person who collected the session data, and about the person who is responsible for the collection of the session data. Note that this may be the same person. The information that you enter on this screen is displayed in the IMDI Browser as follows:



Enter the information into the IMDI Editor as follows:



Collector Name

The name (or ID code) of the person who collected the actual data.

Contact

The contact information about the person responsible for collecting the session data (the name, the address, the e-mail address and the organization he/she belongs to).

Descriptions

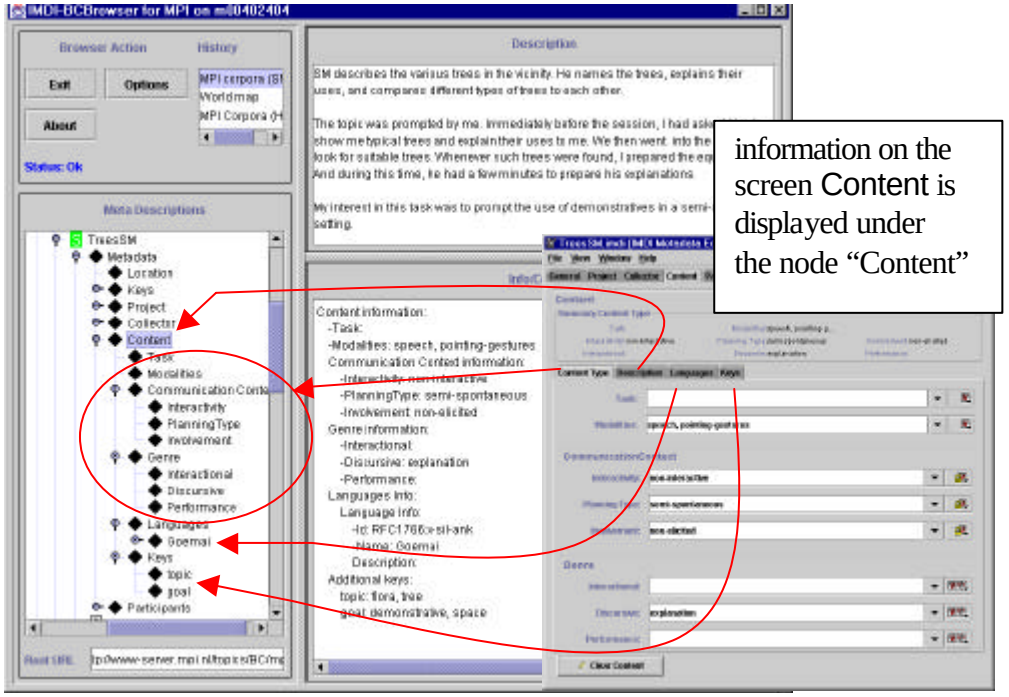
This schema contains additional information about the person who collected the session data, and about the person who is responsible for the collection of the session data. Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

7.4Content

This screen contains information about the content of the session. It is probably the most important screen in the sense that it contains most of the session-specific information, and especially, most of the information that you would want to search for. The following sub-screens are available:

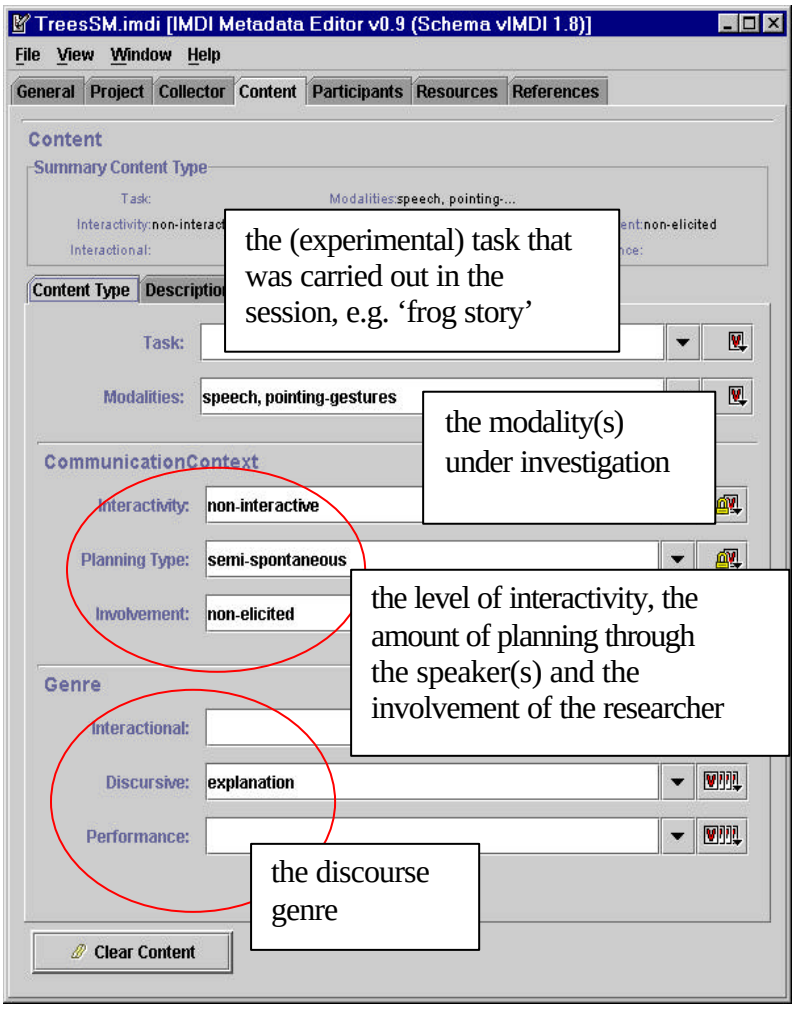
Content Type sub-screen	Information about the task, the modalities, the communication context, and the genre of the session.
Description sub-screen	Description of the content of the session.
Languages sub-screen	Information about the language(s) used in the session.
Keys sub-screen	Keywords that are relevant to the content of the session.

The information that you enter on these four screens is displayed in the IMDI Browser as follows:



Content Type

This sub-screen contains information about the task, the modalities, the communication context, and the genre of the session. For example:



The information to be entered into these fields is standardized to a certain extent (cf. "Appendix 3: Lists of recommended vocabularies"). And although there is room for variation, we recommend that you keep as closely as possible to the standardized lists as this will facilitate the search process.

There are pull-down menus available for all fields, which contain the suggested vocabularies. You can either choose an item from the pull-down menu. Or you can start typing, in which case the pull-down menu will automatically open to display the available options. Note that the pull-down menus will be continuously updated in future versions to include further options.

You can enter more than one item in each field. In this case, please separate them with a comma.

Task

This field contains the name of the (experimental) task that was carried out in the session. Choose one of the following items or type in another item:

"info-kiosk"
"wizard-of-oz"
"travel-planning"
"room reservation"
"frog story"

Modalities

This field contains the modalities that are under investigation. Choose one of the following items or type in another item:

"speech"
"writing"
"gestures"
"pointing-gestures"
"signs"
"eye-gaze"
"facial-expressions"
"emotional-state"
"haptic"

CommunicationContext

This schema contains information about the communication context, i.e., levels of participant interaction, the degree of planning through the speaker(s) and the involvement of the researcher. In all fields, you have to choose an item from the pull-down menu; other items are not accepted.

- Interactivity

  This field specifies the level of participant interaction. Choose from the following entries:

name	comments
• "interactive"	a verbal interaction between at least two participants. It may or may not include an investigator, e.g.: • conversation • many narratives • matching game
• "non-interactive"	a monologue, produced without expecting extended verbal responses from the hearer(s), e.g.: • many oratory texts and songs • some narratives • procedural texts
• "semi-interactive"	primarily a monologue, but punctuated by repeated interjections from the hearer(s), e.g.: • a child interrupting a narrative • hearer(s) repeatedly prompting a narrator

- Planning Type

  This field specifies the degree of planning through the consultant. Choose from the following entries:

name	comments
• "spontaneous"	an unprompted speech whose topic is not determined by the investigator or an observer, e.g.: • conversation • chatting • joke-telling • singing while harvesting
• "semi-spontaneous"	a prompted speech whose topic is determined in some way by an investigator or a community member, but whose participants speak freely within this context, e.g.: • interview • queries (e.g., 'Tell me about the history of your village.' 'Show me how to make tortillas.') • retellings (e.g., the speaker is asked to re-tell a story from a picture book, or to describe a task in his/her own words) • promptings (e.g., children answering a teacher's questions)
• "planned"	the structure and content of the speech is planned in advance by the consultant/performer, e.g.: • political or ritual speech • poem recitation ! Note: This entry does not (necessarily) refer to an elicitation session, where a consultant is given a framework but does not plan his/her answer.

- Involvement

  This field specifies the involvement of the researcher. Choose from the following entries:

• "elicited"	the investigator asks the speaker(s) to produce isolated phonemes, words, utterances or grammatical structures, e.g.: • production of sounds in different phonological environments • responses to (morphological, lexical) questionnaires
• "non-elicited"	the investigator does not interfere verbally with the speech event (other than with his presence)
• "no-observer"	no outside observer is present (only a tape recorder)

Genre

This schema contains information about the discourse type of the session. Again, wherever possible, the discourse types should be chosen from lists of recommended vocabularies.

- Interactional

  If the session contains an interactional genre, enter the type of genre. Choose one of the following items or type in another item:

name	comments
• "conversation"	
• "verbal-contest"	**!** Note: this also includes debates.
• "interview"	
• "meeting"	
• "riddles"	
• "consultation"	**!** Note: this does not refer to an interview with the investigator, but rather to, e.g. a visit to a shaman.
• "greetings"	
• "leavetakings"	
• "humor"	

- Discursive

  If the session contains a discursive genre, enter the type of genre. Choose one of the following items or type in another item:

name	comments
• "procedure"	A directive description of the procedures involved in the preparation or production of something, e.g. 'how to make tortillas'.
• "explanation"	Practical, theoretical, or historical reality statements, e.g. 'how the monkey got its tail'.

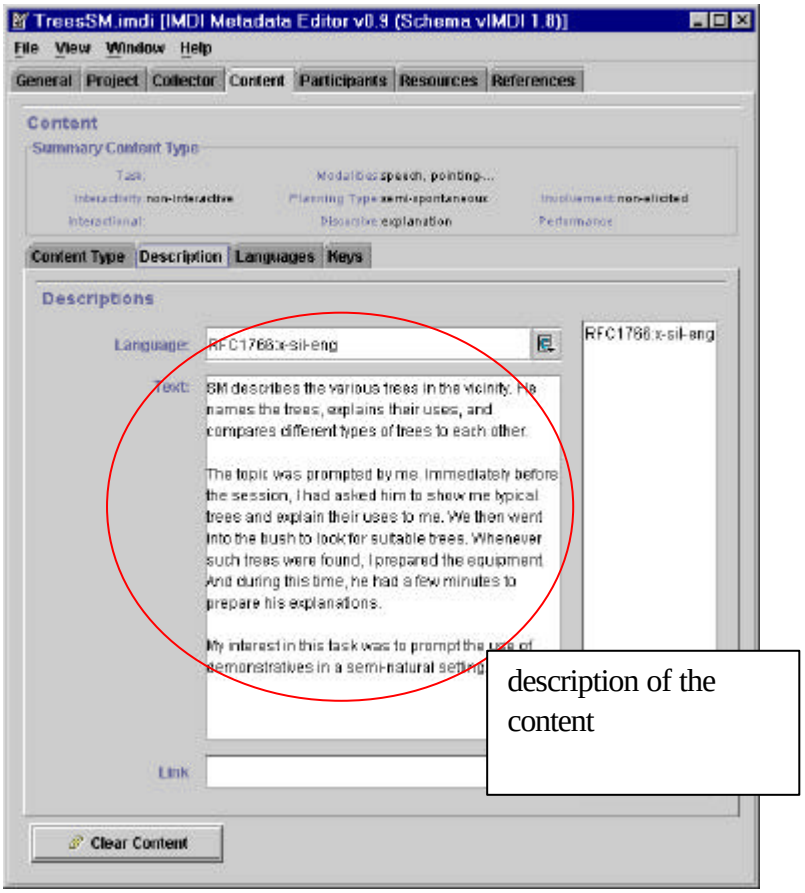
- Performance

  If the session contains a performative genre, enter the type of genre Choose one of the following items or type in another item:

name	comments
• "oratory"	Using speech effectively in a conventionalized format to address an audience within political, legal, ceremonial, or religious settings.
• "oral-history"	An account of firsthand experience, recalled retrospectively and communicated to an interviewer for historical purposes.
• "historical-narrative"	A secondhand account of the experience of historical figures and events, which may be partly or wholly fictional, communicated to both locals and outsiders for both historical purposes and entertainment.
• "narrative"	A recounting of one or more fictional events by one or more narrators to an audience of at least one.
• "oral-poetry"	Spoken or sung, in a relatively structured form (in prosody and syntax), often with distinctive language, e.g. oral epics, narrative poetry, ballads (shorter, lyrical narratives), and panegyric odes.
• "song"	A tune with recognizably structured lyrics, e.g. popular and love songs, lullabies.
• "proverb"	A summary of the wisdom of collective experience, often one line long; formulaic.
• "lament"	
• "insult"	An insolent verbal act creating animosity.

Description

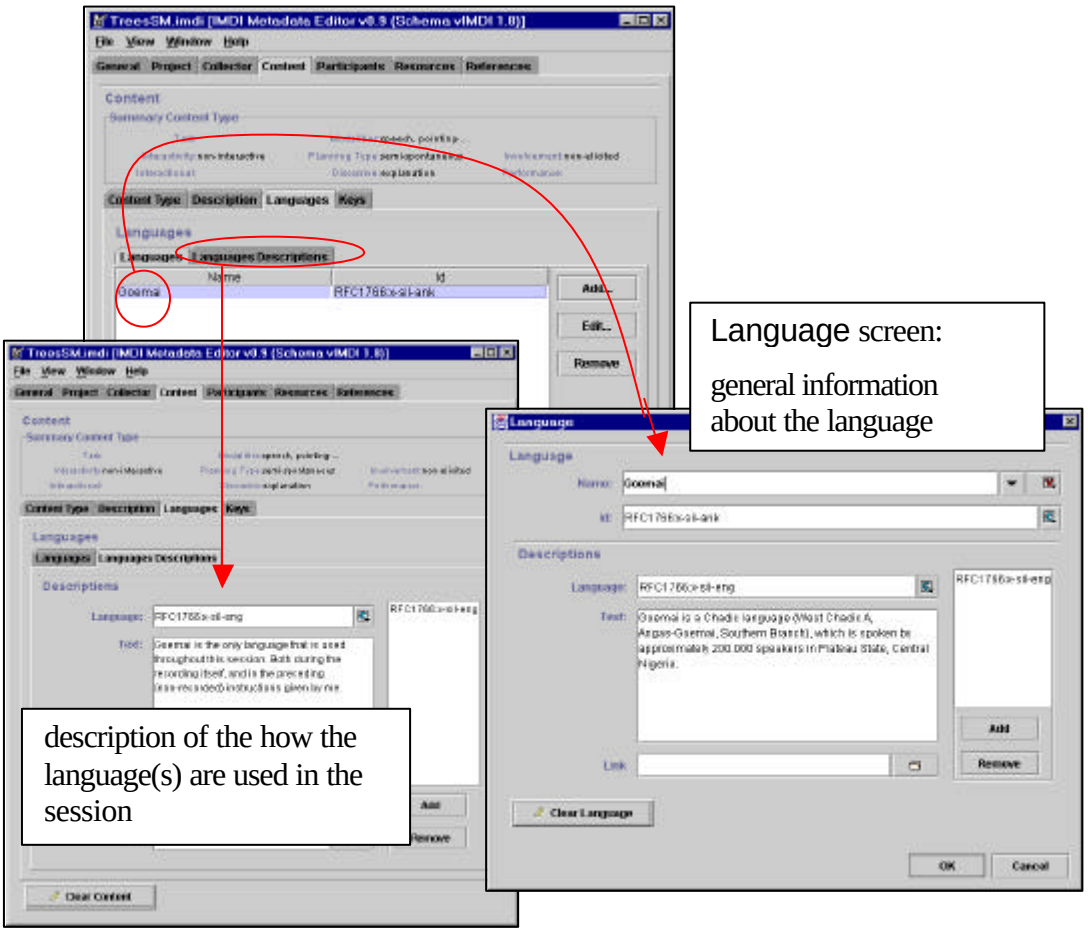
This sub-screen contains a description of the content of the session. For example:



Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

Languages: Languages and Languages Descriptions

These two sub-screens contain information about the language(s) **used** in the session.



Languages

On this sub-screen, enter all languages that are used in the session. The first language to be entered should be the main language of the session. For every language, click on the "Add" button.

If you want to remove a language from the list, click on it to highlight it and then click the "Remove" button.

If you want to modify a language on the list, click on it to highlight it and then click the "Edit" button.

The "Add" and "Edit" buttons give you access to the Language screen, which contains general information about the language (independent of its use in the session).

- Name

  The name of the language.

  ! Note: The name of the language is standardized (cf. "Appendix 4: Lists of languages and language abbreviations"). You can either choose a name from the pull-down menu. Or you can start typing, in which case the pull-down menu will

automatically open to display the available options. You may have to wait a while before the Editor will able to process the letters.

! Note: You have to enter the name in capital letters.

- ID

  The identifier of the language. It has to be entered in a standard format. Do the following:

  1. Click with the mouse button into the ID field and press the key SHIFT+R (or the key SHIFT+I). A dummy code "RFC1766:x-sil-aaa" (or "ISO639:aaa") is displayed.
  2. Replace the last three characters "aaa" with the proper language identifier (cf. "Appendix 4: Lists of languages and language abbreviations" for a list of language identifiers).

- Descriptions

  A description that gives background information about the language in general. Note that the description is **not** about the role of the language in the particular session (reserve this for the Languages Descriptions sub-screen below).

  Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

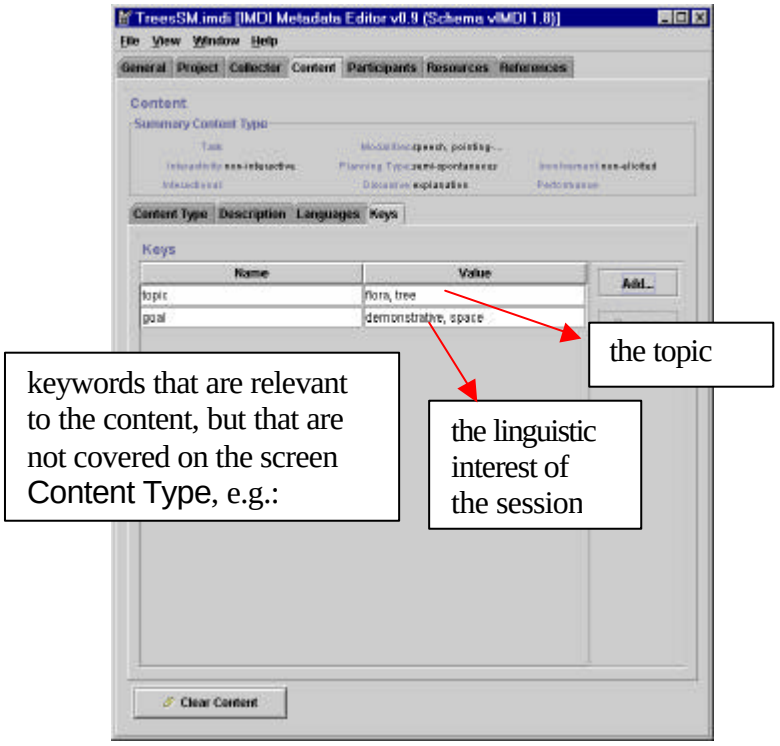
Languages Descriptions

On this sub-screen, enter a description of (a) the set of languages used in the session and (b) of the role of each language as it is used in the session (e.g., language of elicitation, main language, code-switching, etc.). Note that the description does **not** contain background information about the language in general (reserve this for the Language/ Descriptions sub-screen above).

Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

Keys

This sub-screen contains keywords that are relevant to the content of the session. For example:



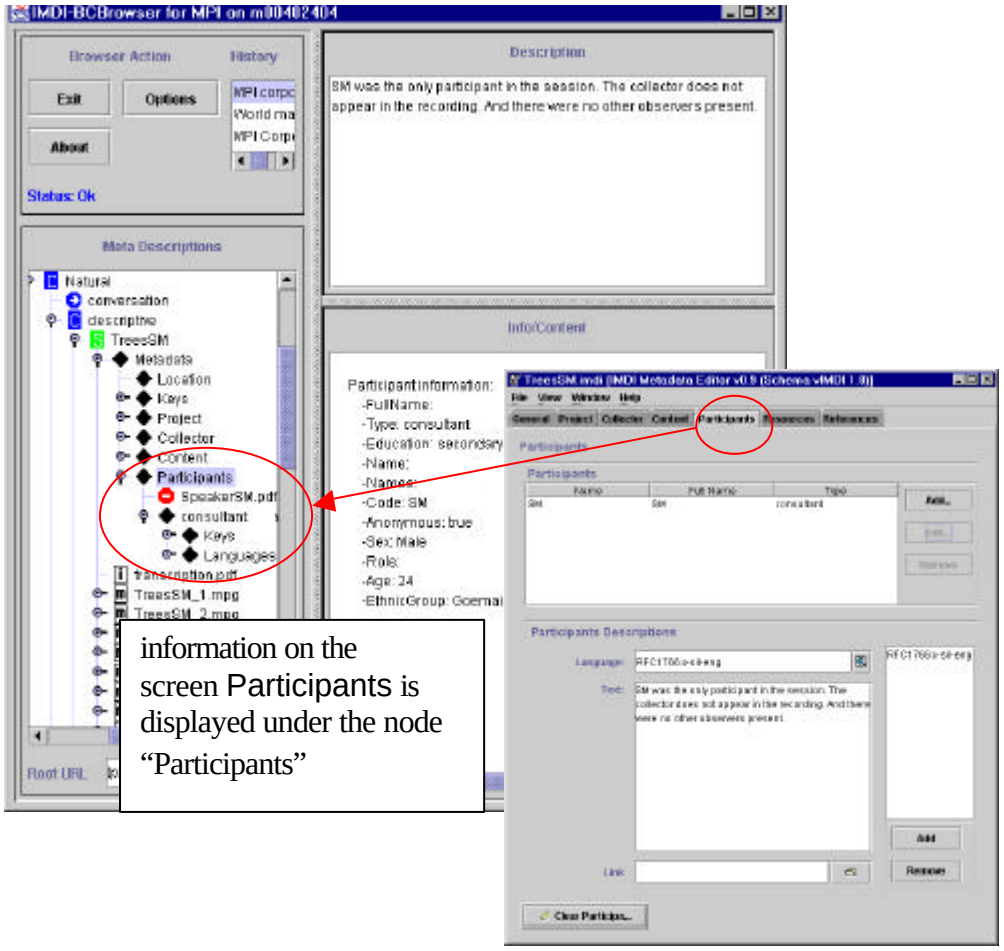
Cf. “Appendix 2: Screens and recurring schemata” for instructions on how to fill in a Keys schema.

7.5Participants

This screen contains information about (a) each participant of the session and (b) the interrelations among different participants of the session. The following schemata are available:

Participants schema	Information about each participant in the session.
Participants Descriptions schema	Description of the interactions and interrelations among different participants of the session.

The information that you enter on this screen is displayed in the IMDI Browser as follows:



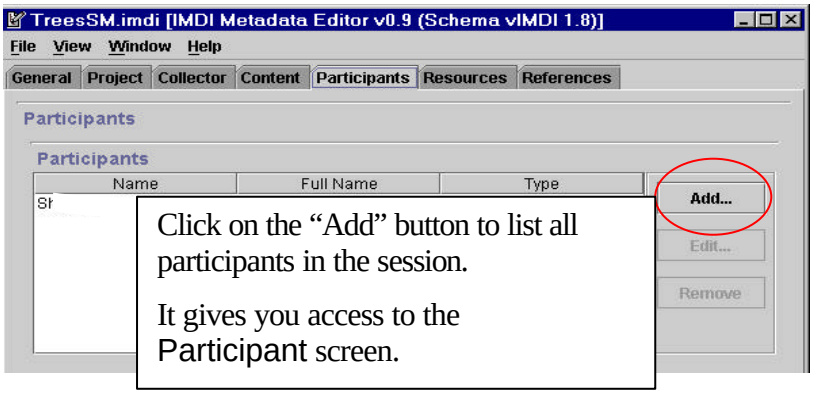
Participants: Participant Information, Descriptions, Languages and Keys

The Participants schema gives you access to the Participant screen and its four sub-screens that contain information about each individual participant of the session.

For every participant, click on the Add button.

If you want to remove a participant from the list, click on it to highlight it and then click the "Remove" button.

If you want to modify a participant on the list, click on it to highlight it and then click the "Edit" button.



The "Add" and "Edit" buttons give you access to the Participant screen. The following four sub-screens are available:

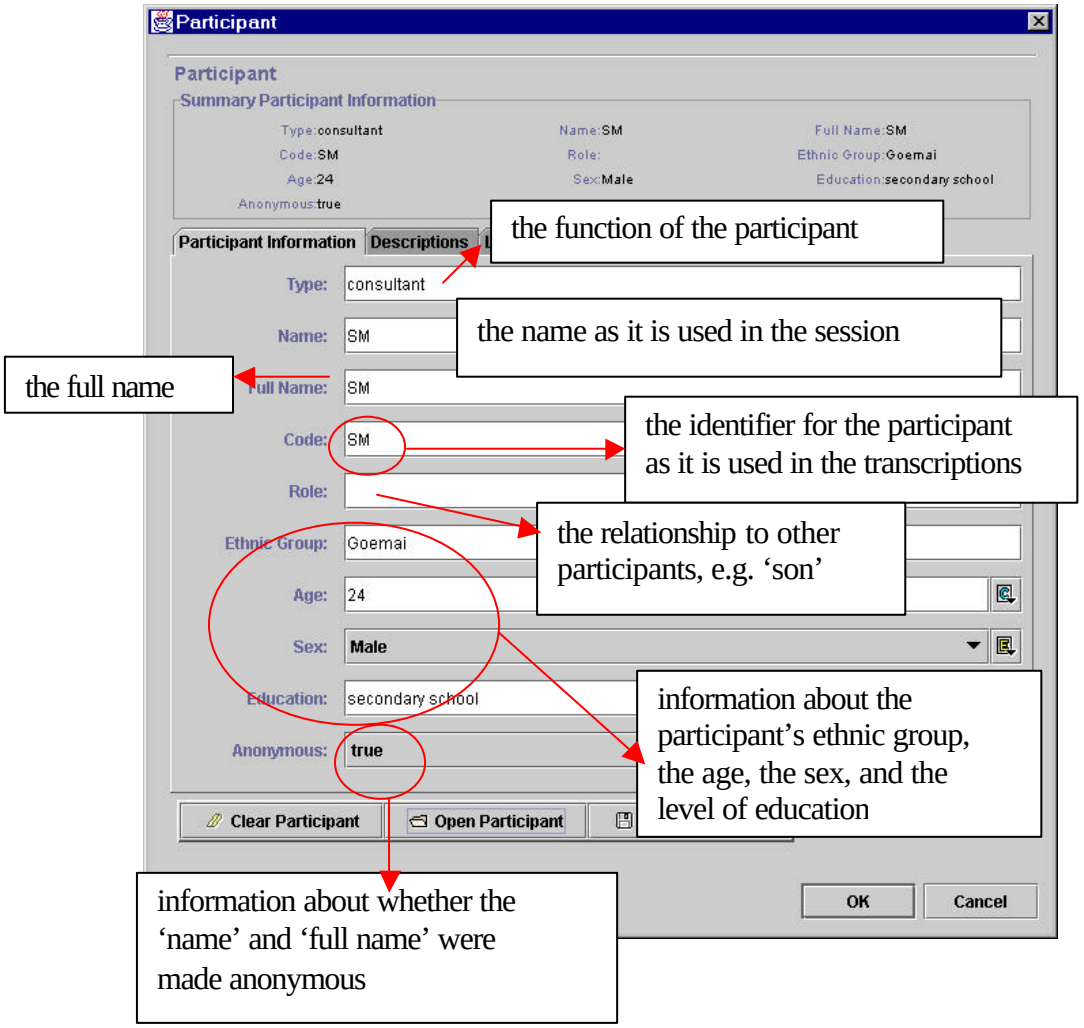
Participant Information sub-screen	Information about the individual participant.
Descriptions sub-screen	General description of the individual participant (independent of his/her role in the session).
Languages sub-screen	Information about all the languages that the participant is familiar with (independent of whether or not the participant uses them in the session).
Keys sub-screen	Keywords that are relevant to the individual participant (independent of his/her role in the session).

Note that this schema concerns the individual participant - information about the interrelations among different participants of the session should be described with the Participants Descriptions schema (cf. below).

! Note: The collector should only appear on this screen if he/she takes an active part in the session, i.e., if utterances of him/her are recorded.

Participant Information

This sub-screen contains information about the individual participant. For example:



- Type

  The function of the participant in the session. For example:

  interviewer
  consultant
  contributor
  computer

- Name

  The name of the participant that is used by others in the session to identify him/her. This is usually not the same as his/her full name.

  ! Note: If you do not want the name to be publicly available, enter a code instead of the name.

- Full Name

  The full name of the participant.

  ! Note: If you do not want the name to be publicly available, enter a code instead of the name.

- Code

  A short unique code to identify the participant. This corresponds to the code that is used in transcriptions and annotations to identify parts belonging to this specific participant.

- Role

  The role of the participant, i.e. his/her relationship to other participants taking part in the session. For example:

  Family relations, e.g. mother, father, child, husband, etc.
  Work relations, e.g. boss, partner, student, teacher, etc.
  Visitor, host, etc.

- Ethnic Group

  The ethnic group of the participant.

- Age

  The age of the participant. Enter the age in the following format: **YY;MM.DD**

  ! Note: We are aware of the fact that the exact age is often not known. In this case, we would nevertheless ask you to give the approximate age in years. This will allow you to later conduct searches on all participants who are in the age group between, e.g., 20 and 30 years of age.

  ! Note: The options "unknown" (for an unknown age) and "undefined" (for an artificial participant such as a computer) are available, too.

- Sex

  The sex of the participant. Choose it from the pull-down menu.

  ! Note: The option "undefined" refers to an artificial participant such as a computer.

- Education

  The education or literacy level of the participant. For example:

  primary school, secondary school, etc.
  literate, illiterate, etc.

- Anonymous

  If you chose to make the name of a participant anonymous (in the fields Name and Full name above), please specify this here.

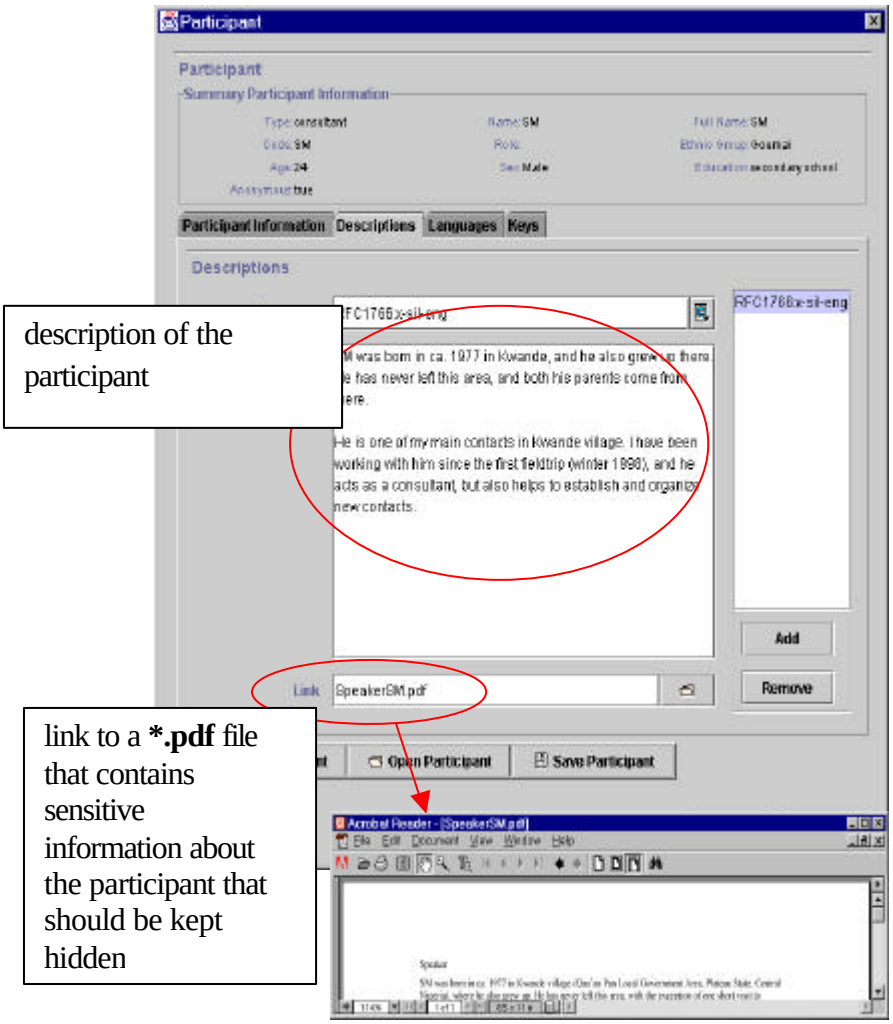
  Choose "true" if the name was made anonymous.

  Choose "false" if you entered the real name.

  ! Note: If you have chosen the option "true", please create a conversion file that specifies the real name of the participant (cf. the screen "Anonymous Info: Anonymous and Access").

Descriptions

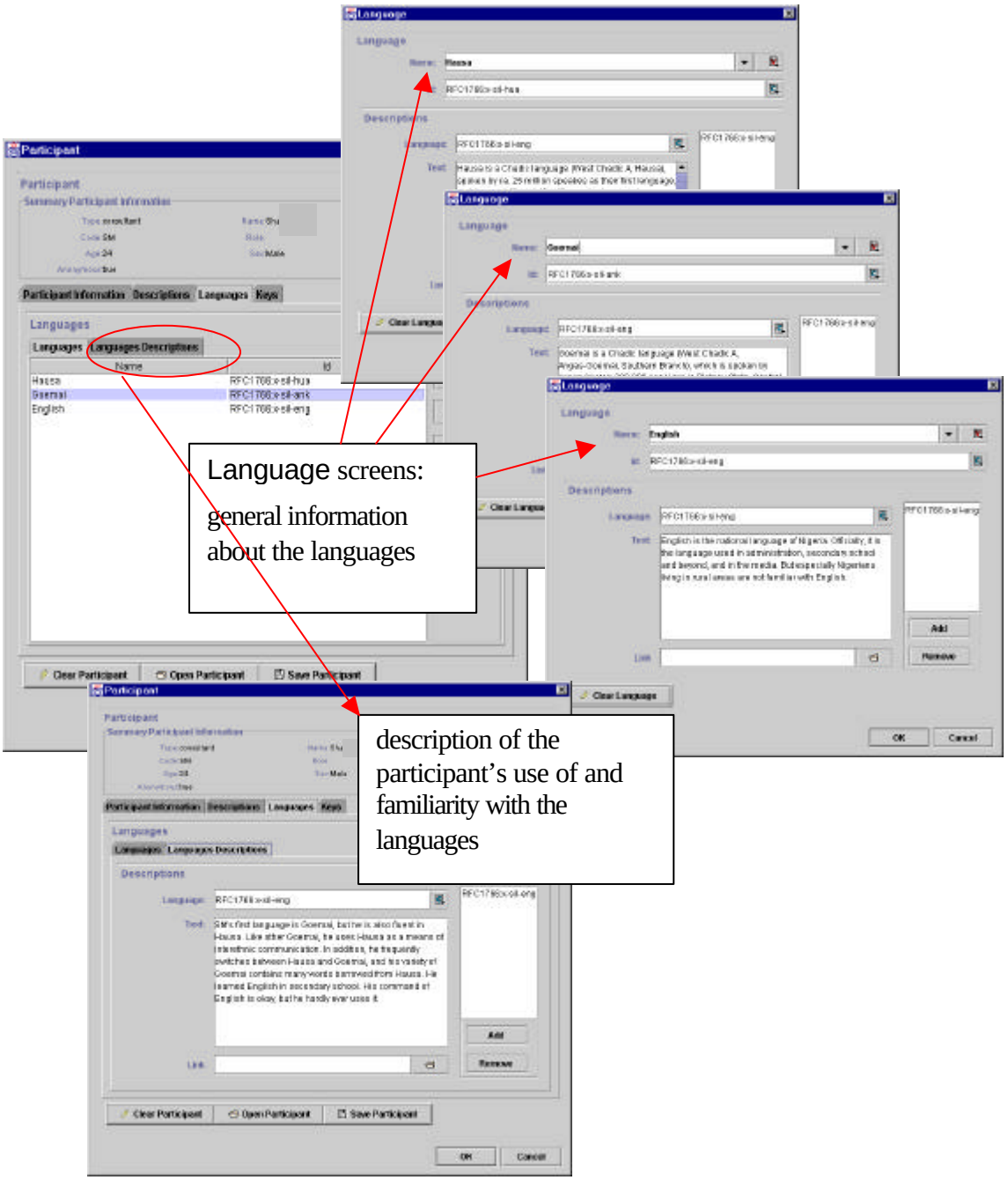
This sub-screen contains a general description of the individual participant (independent of his/her role in the session). For example:



Cf. “Appendix 2: Screens and recurring schemata” for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

Languages

These two sub-screens contain information about all the languages **that the participant is familiar with** (independent of whether or not the participant uses them in the session).



- Languages

  On this sub-screen, enter all the languages that the participant is familiar with. For every language, click on the Add button.

  If you want to remove a language from the list, click on it to highlight it and then click the "Remove" button.

  If you want to modify a language on the list, click on it to highlight it and then click the "Edit" button.

  The "Add" and "Edit" buttons give you access to the Language screen, which contains general information about the language (independent of the participant's familiarity with it).

  - Name

    The name of the language.

    ! Note: The name of the language is standardized (cf. "Appendix 4: Lists of languages and language abbreviations"). You can either choose a name from the pull-down menu. Or you can start typing, in which case the pull-down menu will automatically open to display the available options. You may have to wait a while before the Editor will able to process the letters.

    ! Note: You have to enter the name in capital letters.

  - ID

    The identifier of the language. It has to be entered in a standard format. Do the following:

    1. Click with the mouse button into the ID field and press the key SHIFT+R (or the key SHIFT+I). A dummy code "RFC1766:x-sil-aaa" (or "ISO639:aaa") is displayed.
    2. Replace the last three characters "aaa" with the proper language identifier (cf. "Appendix 4: Lists of languages and language abbreviations" for a list of language identifiers).

  - Descriptions

    A description that gives background information about the language in general. Note that the description is independent of the participant's familiarity with it (reserve such information for the Languages Descriptions sub-screen below).

    Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

- Languages Descriptions

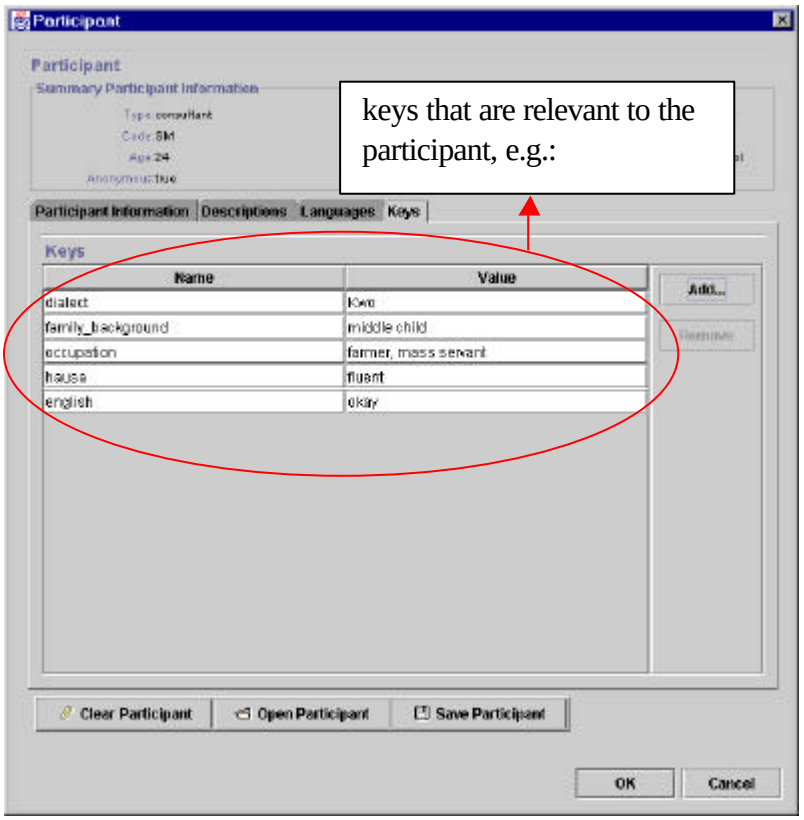
  On this sub-screen, enter a description of the set of languages that the participant is familiar with and specify the participant's familiarity with it. Note that the description

does **not** contain background information about the language in general (reserve this for the Language/ Descriptions sub-screen above).

Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

Keys

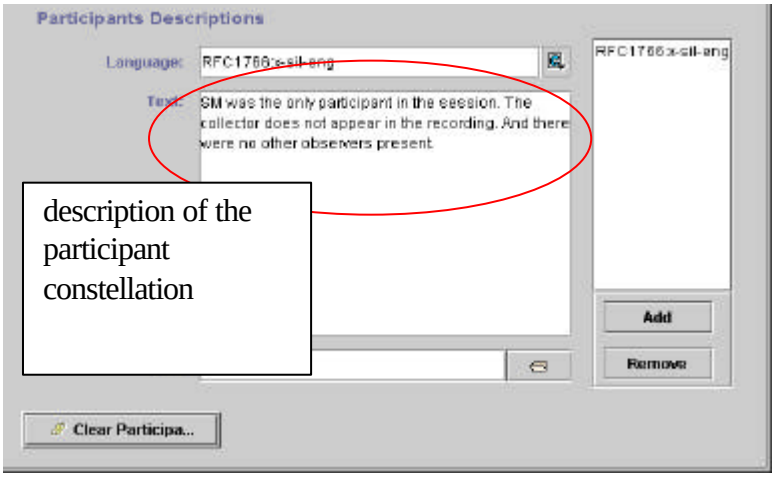
This sub-screen contains keywords that are relevant to the individual participant (independent of his/her role in the session).



Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Keys schema.

Participants Description

This schema contains a description of the interactions and interrelations among different participants of the session. Note that this description concerns the set of participants as a whole - information about specific participants should be described with the Participants/ Participant schema (cf. above).



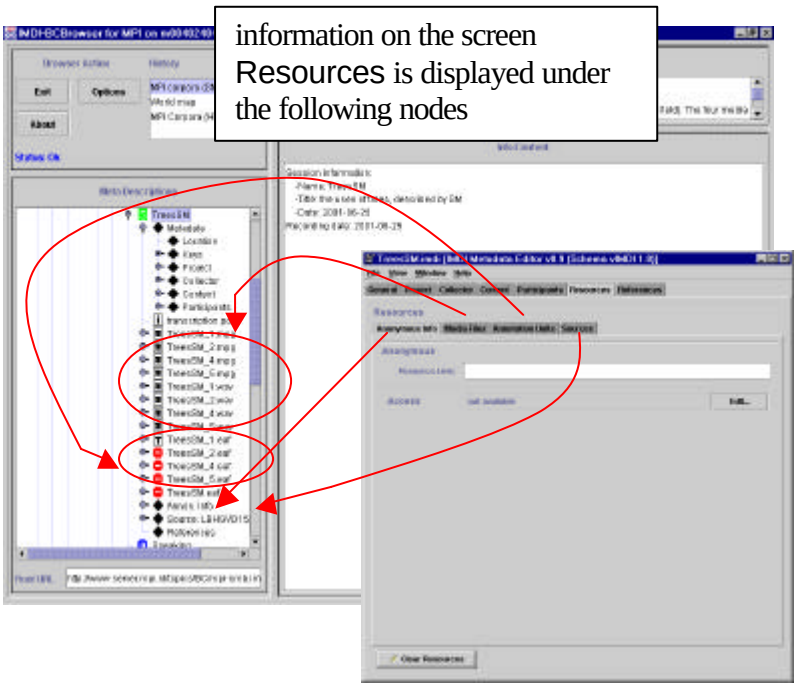
Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

7.6Resources

This screen contains information about (a) access rights, (b) the annotation and digitized media files, and (c) the original media tapes. The following sub-screens are available:

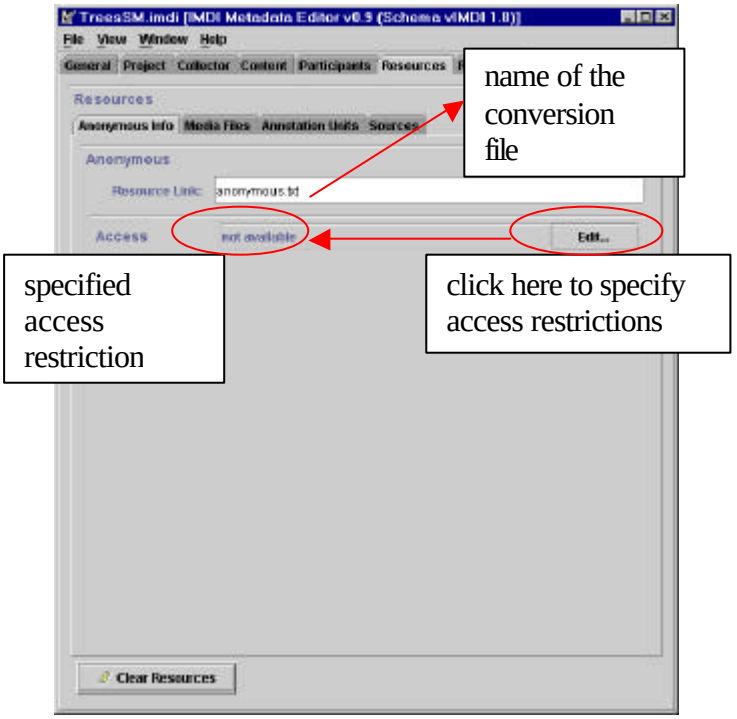
Anonymous Info sub-screen	Information about access rights to the conversion file that contains the real participant names in the IMDI file and the annotation file(s).
Media Files sub-screen	Information about the digitized media file(s), i.e. audio, video or image files.
Annotation Units sub-screen	Information about the annotation units (i.e., the annotation layers and tiers).
Sources sub-screen	Information about the sources, i.e. the original tape(s) of the session.

The information that you enter on these four screens is displayed in the IMDI Browser as follows:



Anonymous Info: Anonymous and Access

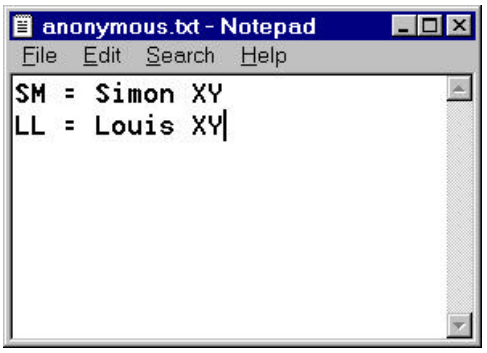
This sub-screen contains information about access rights to the **conversion file** that contains the real names of participants. This information is only necessary in case you chose to make these names anonymous on the Participant Information sub-screen and on the Annotation Unit sub-screen.



Resource Link

Enter the name of the conversion file.

! Note: You can create such a conversion file with any text editor program, e.g.:



Access

This schema contains information about the access rights to the **conversion file**. Click on the "Edit …" button to gain access to the Access schema. Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in an Access schema.

Media Files: Media File, Time Position, Access Policy and Descriptions

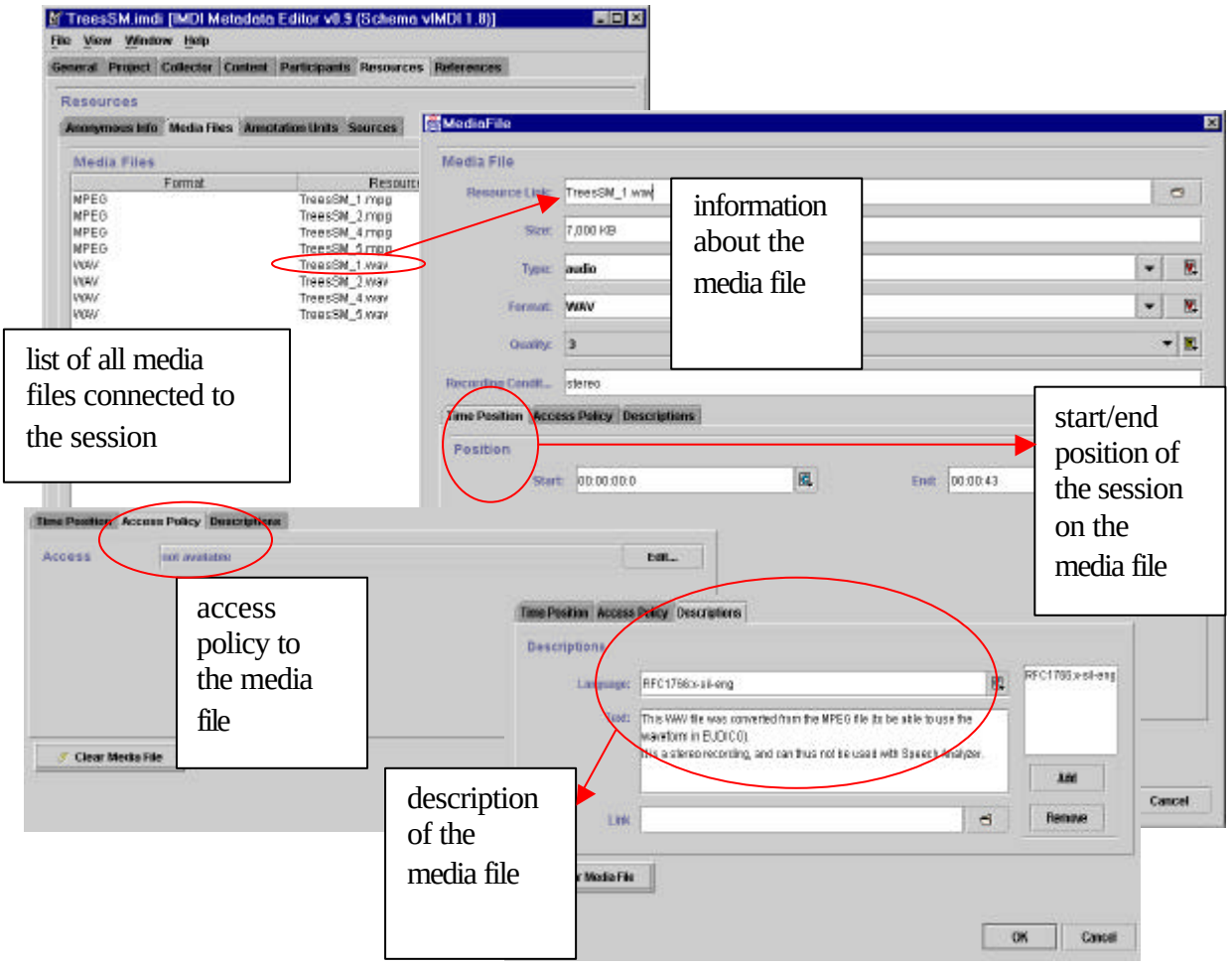
This sub-screen contains information about the digitized media file(s), i.e. audio, video or image files. Enter all the media files that are connected with the session. For every media file, click on the "Add" button.

If you want to remove a media file from the list, click on it to highlight it and then click the "Remove" button.

If you want to modify a media file on the list, click on it to highlight it and then click the "Edit" button.

The "Add" and "Edit" buttons give you access to the Media File screen, which contains the following schemata and sub-screens:

Media File schema	General information about the media file.
Time Position sub-screen	The start and end position of the session within the media file.
Access Policy sub-screen	Access rights to the media file.
Descriptions sub-screen	Description of the media file.



Media File

This schema contains general information about the media file.

- Resource Link

  The link to the corresponding media file. Enter the name of the media file. Please do **not** make use of the 'open folder' icon to the right of that field.

- Size

  The size of the media file in bytes. Note: You do not have to fill in this field.

- Type

  The general type of the media file. Please choose either an option available from the pull-down menu, or specify a different type, e.g.:

  "unknown"
  "unspecified"
  "audio"
  "video"
  "image" (i.e., for photo image)

"drawing"

- Format

  The specific format of the media file. Please choose either an option available from the pull-down menu, or specify a different format, e.g.:

  "unknown"
  "unspecified"
  "MPEG1"
  "MPEG2"
  "MPEG4"
  "WAV"

- Quality

  The quality of the media file. Choose an available option from the pull-down menu (whereby 1 stands for low and 5 for high quality).

- Recording Conditions

  The technical conditions under which the media file was recorded, i.e., the equipment used in the recording (e.g., microphone type, amplifier type, mono/stereo recording, etc.).

Time Position

This sub-screen specifies the start/end positions of the session on the media file (if the media file contains several sessions). Please enter the start/end position in the following format: **hh:mm:ss:f** (i.e., hours:minutes:seconds:frames).

Access Policy

These sub-screens contain information about the access rights to the **media file**. Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in an Access schema.

Descriptions

This sub-screen contains a description of the media file. Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

Annotation Units: Annotation Unit, Information, Access Policy and Descriptions

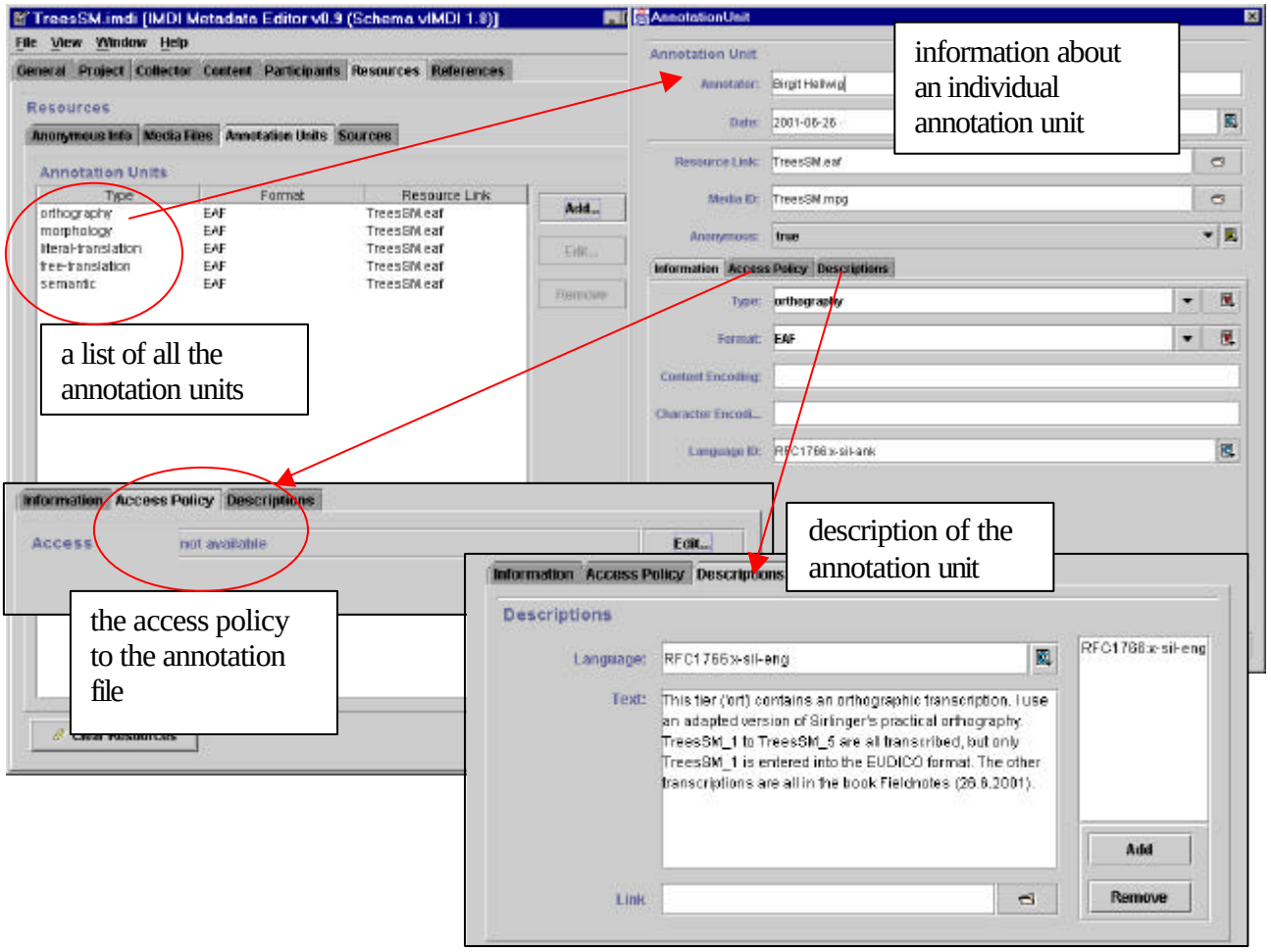
This sub-screen contains information about the annotation units (i.e., the annotation layers and tiers). Enter all annotation units that are connected with the session. For every annotation unit, click on the "Add" button.

If you want to remove an annotation unit from the list, click on it to highlight it and then click the "Remove" button.

If you want to modify an annotation unit on the list, click on it to highlight it and then click the "Edit" button.

The "Add" and "Edit" buttons give you access to the Annotation Unit screen, which contains the following schemata and sub-screens:

Annotation Unit schema	General information about the annotation unit (tier, layer) and its corresponding annotation file.
Information sub-screen	Information about the annotation unit (tier, layer).
Access Policy sub-screen	Access rights to the annotation file.
Descriptions sub-screen	Description of the annotation unit, its tiers and the tools used.



! Note: The label 'annotation unit' does **not** refer to the annotation file itself, but to one layer or tier of annotations (e.g., the free translation, or the phonetic transcription).

! Note: You do not have to enter all annotation units, but it is necessary that you list at least one annotation unit per annotation file. This is needed for technical reasons in order to create a link to the file(s).

Annotation Unit

This schema contains general information about the annotation unit (tier, layer) and its corresponding annotation file.

- Annotator

  The name of the person who did the annotation/transcription. If it is more than one person, list them all in this field (separated by commas).

- Date

  The date when the annotation unit was created. Please enter the date in the following format: **YYYY-MM-DD**, e.g. 2000-12-30.

- Resource Link

  The link to the file containing the annotation unit. Enter the name of the annotation file. Please do not make use of the 'open folder' icon to the right of that field.

- Media ID

  The link to the media file from which the annotation unit originated. Enter the name of the media file. Please do not make use of the 'open folder' icon to the right of that field.

- Anonymous

  If you have made the name of the participant(s) anonymous in the annotation file, please specify this here.

  Choose "true" if the name was made anonymous.

  Choose "false" if you used the real name.

  ! Note: If you have chosen the option "true", please create a conversion file that specifies the real name of the participant (cf. the screen Resources/ Anonymous).

Information

This sub-screen contains information about the annotation unit (tier, layer). For example:

- Type

  The type of annotation unit. Please choose either an option available from the pull-down menu, or specify a different type, e.g.:

  "orthography"
  "phonemic"
  "phonetic"
  "morphology"
  "morphosyntax"
  "syntax"
  "free-translation"
  "literal-translation"
  "semantic"

! Note: You can choose to enter all annotation units of a single annotation file into this field instead of entering them on separate screens. In this case, specify all types that occur and separate them through a comma.

- Format

  The file format that is used for creating the annotation unit. Please choose either an option available from the pull-down menu, or type in a different format, e.g.:

  "CHAT"
  "Shoebox"
  "RDBMS"
  "TRS"
  "EAF"
  "AIF"
  "BAS"

- Content Encoding

  The name of the encoding scheme used for creating the annotation unit (if applicable). For example:

  Eurotype (i.e., following the Eurotype guidelines)

- Character Encoding

  The name of the character encoding used for the annotation unit, i.e. the name of the font used.

- Language ID

  The identifier of the language that is used for the annotation unit, e.g. 'English' for an English translation. It has to be entered in a standard format. Do the following:

  1. Click with the mouse button into the ID field and press the key SHIFT+R (or the key SHIFT+I). A dummy code "RFC1766:x-sil-aaa" (or "ISO639:aaa") is displayed.
  2. Replace the last three characters "aaa" with the proper language identifier (cf. "Appendix 4: Lists of languages and language abbreviations" for a list of language identifiers).

Access Policy

These sub-screens contain information about the access rights to the **annotation file**. Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in an Access schema.

Descriptions

This sub-screen contains a description of the annotation unit, its tiers and the tools used. Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

Sources: Source, Position, Access Policy and Descriptions

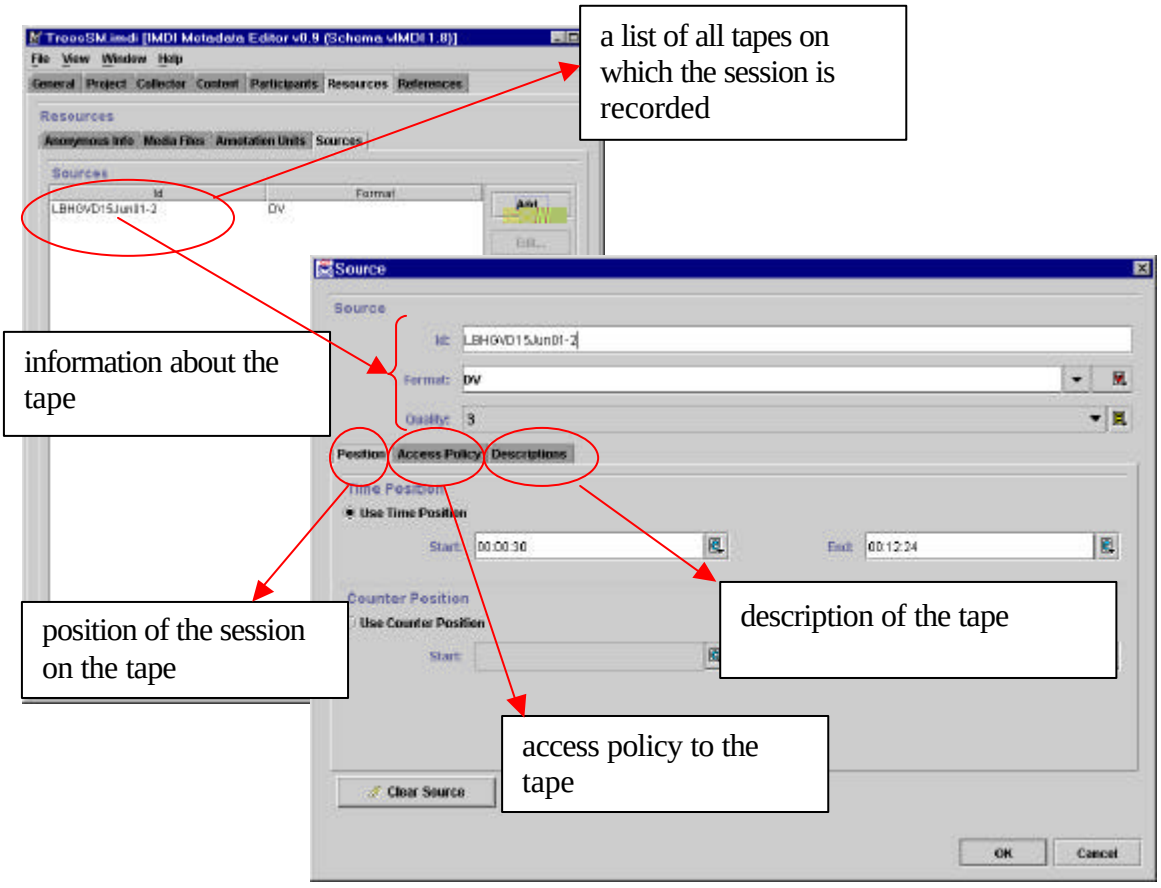
This sub-screen contains information about the sources, i.e. the original tape(s) of the session. Enter all the sources that are connected with the session. For every source, click on the "Add" button.

If you want to remove a source from the list, click on it to highlight it and then click the "Remove" button.

If you want to modify a source on the list, click on it to highlight it and then click the "Edit" button.

The "Add" and "Edit" buttons give you access to the Source screen, which contains the following schemata and sub-screens:

Source schema	General information about the source.
Position sub-screen	The start and end position of the session on the source.
Access Policy sub-screen	Access rights to the source.
Descriptions sub-screen	Description of the source.



Source

This schema contains general information about the source.

- ID

  A short code to identify the source. Please make use of the tape labeling conventions (cf. "Appendix 6: Tape labeling conventions").

- Format

  The physical storage format of the tape. For example:

name	comments
• CC	Compact Cassette (i.e., normal audio cassette)
• CD	Compact Disc
• CDROM	Compact Disc - Read-Only Memory
• DAT	Digital Audio Tape
• MD	Mini Disc
• Reel	Reel-to-reel tape
• DVD	Digital Video Disc
• DVDROM	Digital Video Disc - Read-Only Memory
• Hi8	Hi8 Video Tape
• VHS	VHS Video Tape
• DV	Digital Video
• U-matic	U-matic Tape

- Quality

  The quality of the tape. Choose an available option from the pull-down menu (whereby 1 stands for low and 5 for high quality).

Position

This sub-screen contains the start and end position of the session on the corresponding tape. Please enter the start/end position in the following formats:

- In the case of digital audio tapes use the field Time Position: **hh:mm:ss** (i.e., hours:minutes:seconds).
- In the case of digital video tapes use the field Time Position: **hh:mm:ss:f** (i.e., hours:minutes:seconds:frames).
- In the case of non-digital tapes, use the field Counter Position: enter any number of digits to represent the start/end position.

! Note: Please use DMF or MediaPlayer to determine this the start/end position.

Access Policy

These sub-screens contain information about the access rights to the **source**. Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in an Access schema.

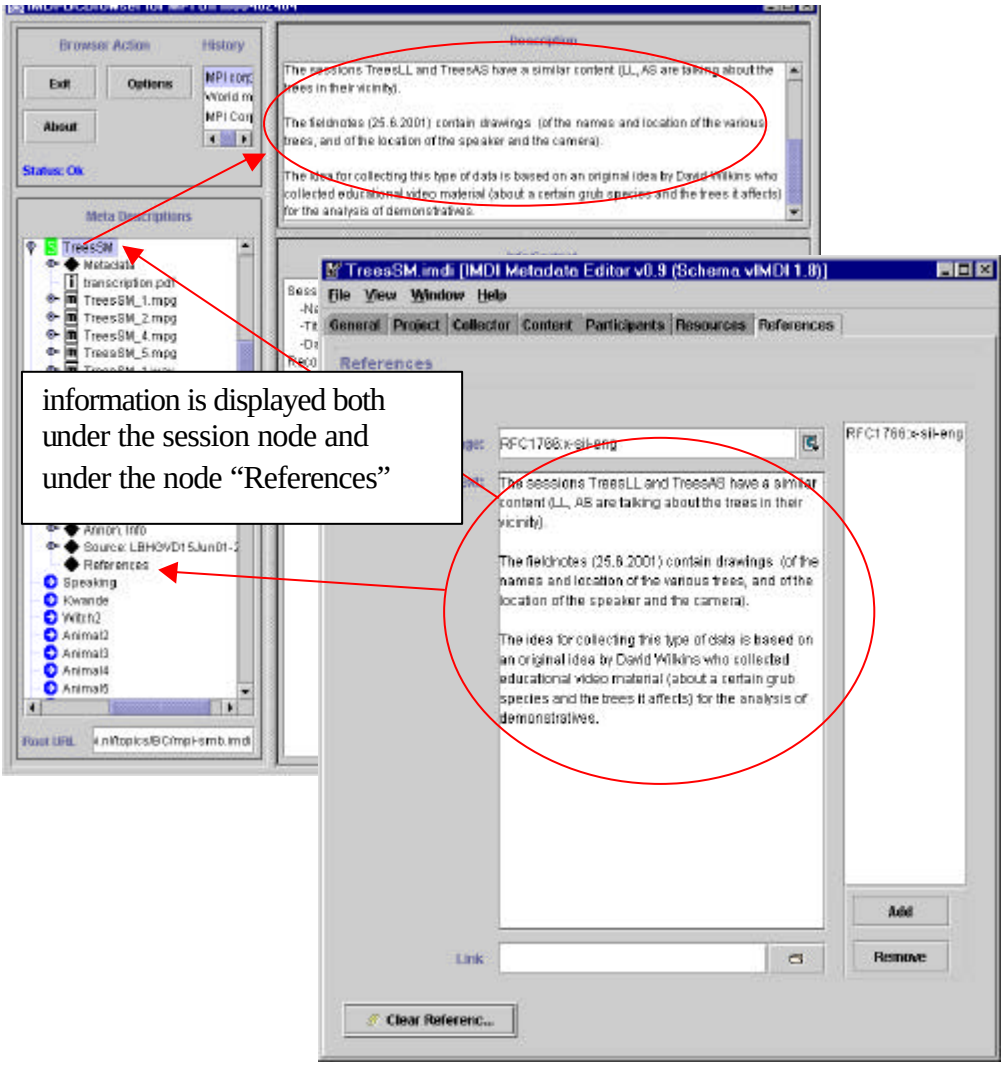
Descriptions

This sub-screen contains a description of the source. Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

7.7References

Descriptions

This schema contains cross-references to other sessions, fieldnotes, or publications, which are relevant to the content of the session. For example:



Cf. "Appendix 2: Screens and recurring schemata" for instructions on how to fill in a Descriptions schema. Remember: The field Language refers to the language in which the description was written - **not** to the language under investigation.

PART IV: APPENDICES

Appendix 1: Key concepts of the IMDI Editor

This appendix briefly introduces and defines the key concepts and termini used in the IMDI Editor.

IMDI Editor

The IMDI Editor is a program that provides a standardized format into which information is entered concerning the circumstances of the data collection (e.g., the date and location, the participants, the content, the original audio or video tape, the corresponding annotation and digitized media files). The data that is entered is saved in form of an IMDI file and is displayed in the IMDI Browser.

IMDI File

An IMDI file is the file that is created with the IMDI Editor. It contains all the information about the data collection. One IMDI file corresponds to one session, and to any number of annotation and digitized media files.

IMDI Browser

The IMDI Browser displays the content of every IMDI file in a standardized way. It links the IMDI file to the corresponding annotation and digitized media files, allowing simultaneous access to all the information available for a specific session. In addition, it supports the possibility to search the content of all IMDI files; and its hierarchical structure allows easy navigation down to the session level.

Session

The session is the basic unit that is described by one IMDI file. It refers to one linguistic unit having the same overall content, the same set of participants, and the same location and time. For example, one elicitation session on topic X, or one folktale, or one matching game. Note that it can correspond to several different annotation and media files (and tapes), and conversely, it can also correspond to just one part within an annotation and media file (and tape).

Screens, schemata and fields

The IMDI Editor is organized around a number of screens, schemata and fields. Screens bundle all the information on an overall topic, e.g., on the participants, the content or resources. Schemata are units with a specific internal structure that can recur on different screens such as a description or a keyword schema. Fields are the smallest units into which the actual information is entered.

Template

A template contains a recurring piece of information on a specific topic, e.g. a participant. A template can be created within the IMDI Editor, saved and then re-loaded every time it is needed.

Info file

An Info file contains additional information that, for various reasons, is not entered into the IMDI file itself. The IMDI file contains a link to the Info file.

Link

A link relates an IMDI file to any other file: info file(s), annotation file(s) and media file(s). The names of the corresponding files are inserted into the IMDI file, and the IMDI Browser then relates all these files to each other, allowing for simultaneous access.

Appendix 2: Screens and recurring schemata

This appendix gives an overview of (a) all (sub-) screens and their content and (b) of all recurring schemata.

1. Screens and their content

2. Recurring schemata and their content

- Descriptions
- Keys
- Access

1. Screens and their content

The following table gives an overview of all (sub-) screens and their content.

General	General information about (a) the session (name, date, location) and (b) the IMDI file.
Descriptions	Description of the circumstances under which the data for the session was collected.
Location	Information about the location at which the data for the session was collected.
Keys	Keywords that are relevant to (a) either the collection of the data or (b) to the creation of the IMDI file.
Project	Information about the larger project within which the data for the session was collected.
Collector	Information about the person who collected the session data, and about the person who is responsible for the collection of the session data.
Content	Information about the content of the session.
Content Type	Information about the task, the modalities, the communication context, and the genre of the session.
Description	Description of the content of the session.
Languages	Information about the language(s) used in the session.
Language	General information about the language (independent of its use in the session).
Descriptions	Description of (a) the set of languages used in the session and (b) of the role of each language as it is used in the session.
Keys	Keywords that are relevant to the content of the session.

Participants	Information about (a) each participant in the session and (b) the interrelations among different participants of the session.
Participant(s)	Information about each participant of the session.
Information	Information about the individual participant.
Descriptions	General description of the individual participant (independent of his/her role in the session).
Languages	Information about all the languages that the participant is familiar with (independent of whether or not the participant uses them in the session).
Language	General background information about the language itself (independent of the participant's familiarity with it).
Descriptions	Description of the set of languages as a whole that the participant is familiar with.
Keys	Keywords that are relevant to the individual participant (independent of his/her role in the session).
Descriptions	Description of the interactions and interrelations among different participants of the session.
Resources	Information about (a) access rights, (b) the annotation and digitized media files, and (c) the original media tapes.
Anonymous Info	Information about access rights to the conversion file that contains the real participant names in the IMDI file and the annotation file(s).
Anonymous	Name of the conversion file.
Access	Access rights to the conversion file.
Media Files	Information about the digitized media file(s), i.e. audio, video or image files.
Media File	General information about the media file.
Time Position	The start and end position of the session within the media file.
Access Policy	Access rights to the media file(s).
Descriptions	Description of the media file.
Annotation Units	Information about the annotation unit(s) (i.e., the annotation layers and tiers).
Annotation Unit	General information about the annotation unit (tier, layer) and its corresponding annotation file.
Information	Information about the annotation unit (tier, layer).
Access Policy	Access rights to the annotation file(s).
Descriptions	Description of the annotation unit, its tiers and the tools used.

Sources	Information about the source(s), i.e. the original tape(s) of the session.
Source	General information about the source.
Position	The start and end position of the session on the source.
Access Policy	Access rights to the source(s).
Descriptions	Description of the source.
References	Cross-references to other sessions, fieldnotes, or publications, which are relevant to the content of the session.

Table (2): Screens and their content

2. Recurring schemata and their content

Descriptions

A number of different screens contain a Descriptions schema. The following screens contain such a schema:

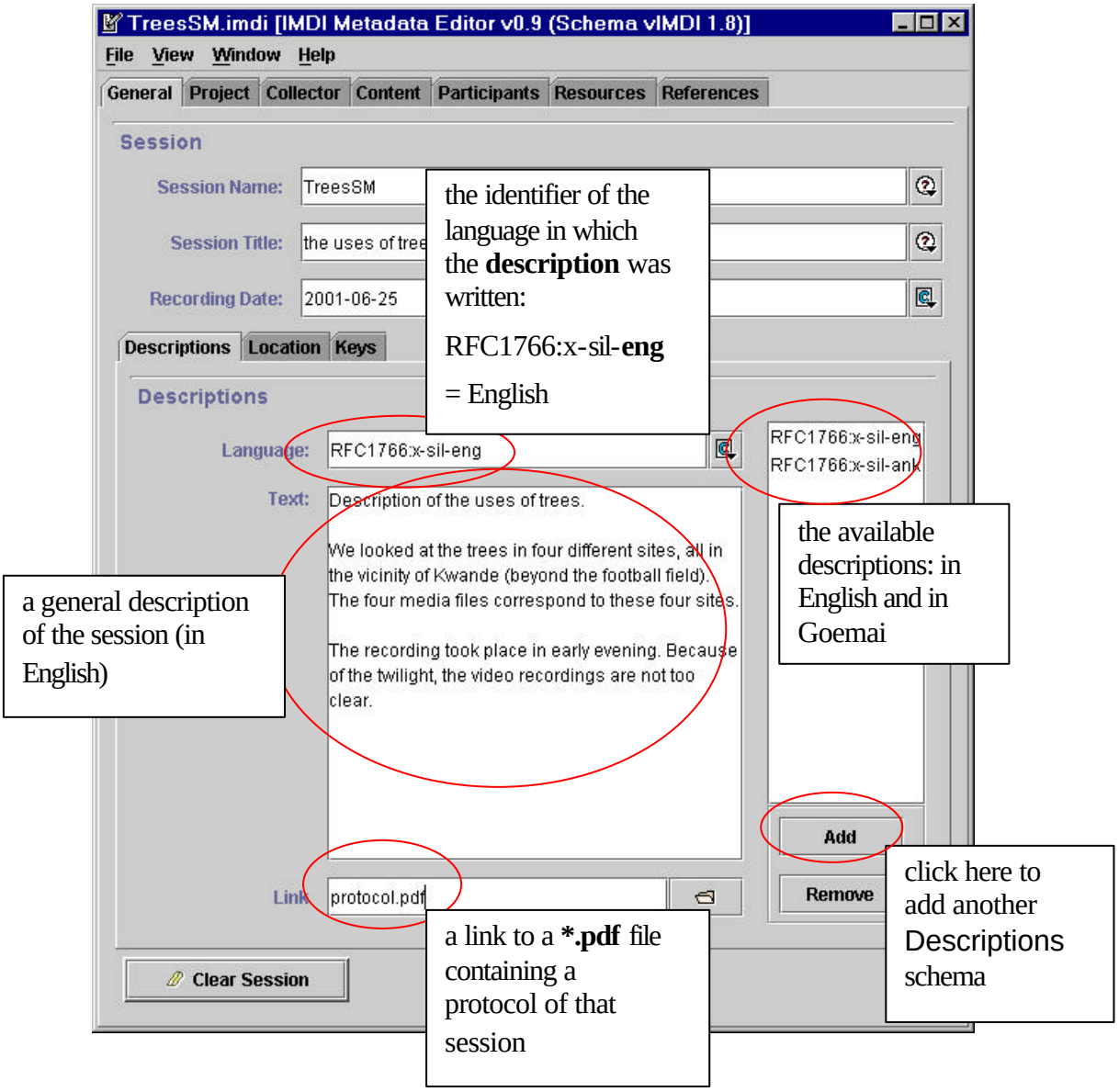
General	Description of the circumstances under which the data for the session was collected.
Project	Description of the scope and the goals of the project, within which the data for the session was collected.
Collector	Description of the person who collected the session data, and about the person who is responsible for the collection of the session data.
Content	Description of the content of the session.
Content/ Languages	Description of (a) the set of languages used in the session and (b) of the role of each language as it is used in the session.
Participants/ Participant	Description of the individual participant (independent of his/her role in the session).
Participants/ Participant/ Languages	Description of the set of languages as a whole that the participant is familiar with.
Participants	Description of the interactions and interrelations among different participants of the session.
Resources/ Media File	Description of the media file.
Resources/ Annotation Unit	Description of the annotation unit, its tiers and the tools used.
Resources/ Source	Description of the source.
References	Description of cross-references to other sessions, fieldnotes, or publications, which are relevant to the content of the session.

Table (3): Descriptions schemata

In addition, the following recurring screens contain a Descriptions schema:

Access	Description of the applied access restrictions (to the conversion file, to the media file, to the annotation file, to the source).
Language	Description of the language (independent of its use in the session or the participant's familiarity with it).

A Descriptions schema contains a prose description about a specific topic, which serves as a reminder of the circumstances of data collection and which is displayed whenever you access the relevant session in the IMDI Browser (cf. section 4.3.2). For example:



This schema always has the same structure:

Language — The language in which the description is written (it does **not** refer to any language used in the session). The language has to be entered in a standard format. Do the following:

1. Click with the mouse button into the ID field and press the key SHIFT+R (or the key SHIFT+I). A dummy code "RFC1766:x-sil-aaa" (or "ISO639:aaa") is displayed.

2. Replace the last three characters “aaa” with the proper language identifier (cf. “Appendix 4: Lists of languages and language abbreviations” for a list of the language identifiers).

Text — A prose text that gives information relevant to the specific topic.

! Note: Unfortunately, the description in the text field may get distorted as new lines and tabs are sometimes automatically inserted by the xml parser. We hope to fix this problem in a later version of the IMDI Editor.

Link — A link to an Info file that contains information relevant to the specific topic. Do the following:

1. Enter the name of the Info file.

! Note for all corpora that are managed by the Max Planck Institut für Psycholinguistik, Nijmegen: It is only necessary to enter the name of the file, but not its location. If you use the IMDI Editor for your own individual purposes, you will have to enter both the name and the location.

! Note: Remember to include only session-specific information (session protocols, digitized photos, sensitive information) into Info files that are linked to the session (cf. section 4.3.3).

! Note: The Info file has to be saved as a ***.txt** file, as a ***.pdf** file or as an ***.html** file. Unfortunately, the IMDI Browser cannot display other kinds of files, e.g., Word files. If your need help with converting files into a ***.txt**, ***.pdf** or ***.html** format, please contact corpus.manager@mpi.nl.

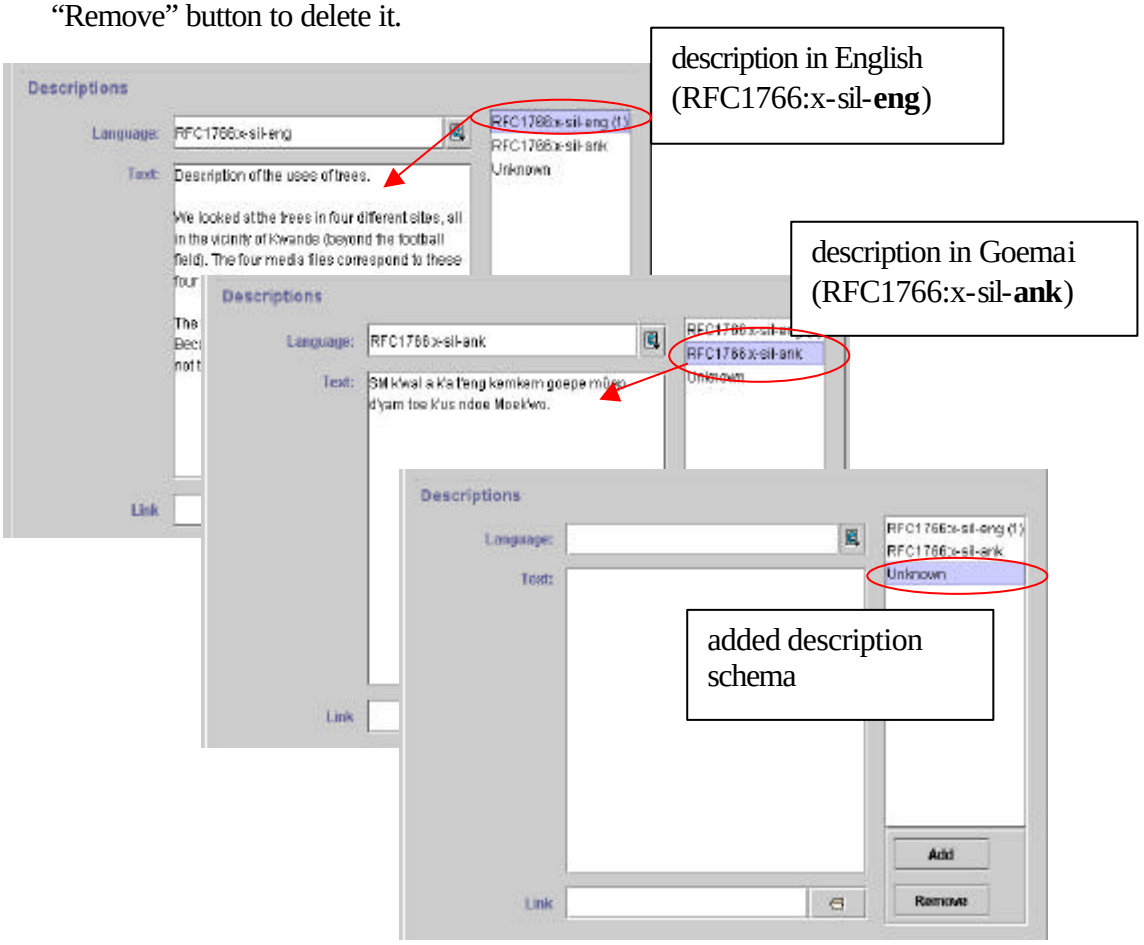
You can create descriptions in different languages, e.g. one description in English (for the research community) and one in the language under investigation (for the language community). In this case, you have to add another Descriptions schema. Do the following:

1. Click on the "Add" button to add a new schema.

   The names of all schemata are displayed at the right of the Descriptions schema. They appear under the identifier of the language (as you have defined it in the Language field).

   To access a schema, highlight its identifier by clicking on it. It appears in blue color, and its content is displayed. The content of all other schemata is hidden.

   To delete a schema, highlight its identifier by clicking on it. Then click on the "Remove" button to delete it.



! Note: If the content of the Text field is not visible, click on the language identifier to the right of the window to highlight it. This will display its content.

! Note: When you remove the last available language identifier from the list, the fields Language, Text and Link will disappear. To make them visible again, you have to close the Editor and re-open it.

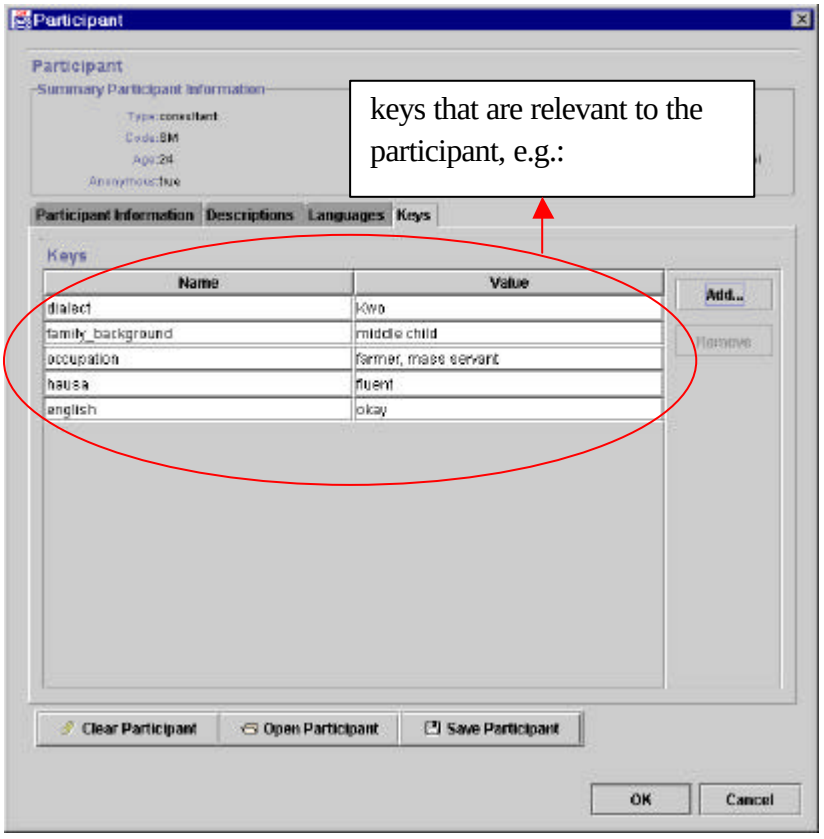
Keys

A number of different screens contain a Keys schema. The following screens contain such a schema:

General	Keywords that are relevant to (a) either the collection of the data or (b) to the creation of the IMDI file.
Content	Keywords that are relevant to the content of the session.
Participant	Keywords that are relevant to the individual participant (independent of his/her role in the session).

Table (4): Keys schemata

Keys contain user-defined information that is (a) project-specific, (b) not taken care of in the standardized fields, and (c) meant to be searchable (cf. section 4.3.2). For example:



To add a key, do the following:

1. Click on the "Add" button. A new key will be added.
2. Specify a "name" for the key. For technical reasons, the name has to be one single word (e.g., 'MetaDescriptionCreator'). Do not use blank spaces.
3. Specify a "value" for the key. Multiple words are allowed (e.g., 'Student Assistant XY').

To delete a key, highlight it by clicking on it and then click on the "Remove" button.

Access

A number of different screens contain an Access schema. The following screens contain such a schema:

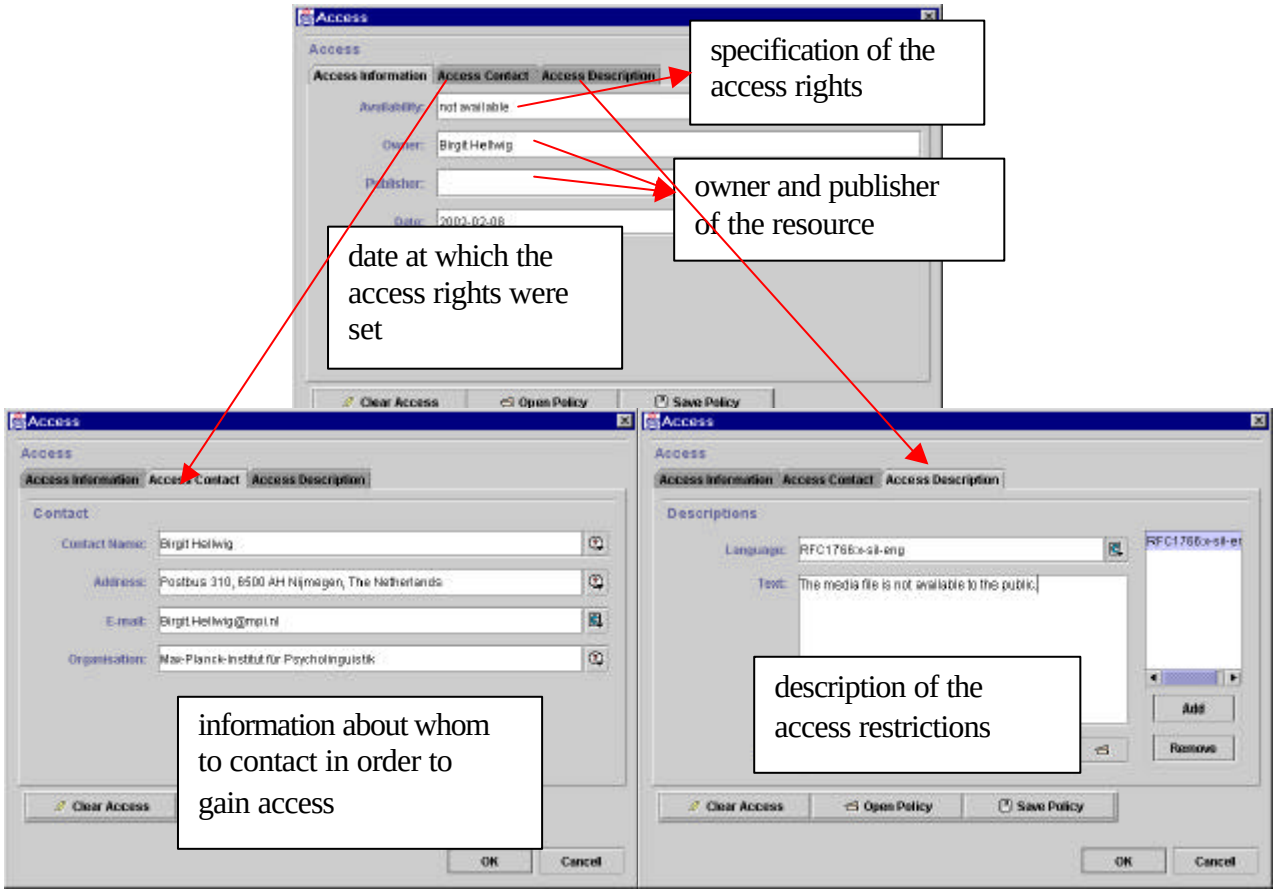
Resources/ Anonymous	Access rights to the conversion file that assigns pseudo participant names to the real participant names in the IMDI file and the annotation file(s).
Resources/ Media File	Access rights to the media file(s).
Resources/ Annotation Unit	Access rights to the annotation file(s).
Resources/ Source	Access rights to the source(s).

Table (5): Access schemata

This schema groups information about access rights to a resource.

> **!** Note: For the moment, access to the actual data (media files, annotation and transcript files, and sources) is automatically denied to everybody except you. For future developments, we would nevertheless ask you to provide the required information.

Specify the access rights as follows:



Information	General information about access rights to the resource.
Availability	Information about the availability of the resource. This field is not standardized yet - for the moment, please enter a prose text (e.g., not available, available in 5 years, available to person XY).
Date	Date at which the access rights to the resource was set (e.g., when you have set the access restrictions to 'available in 5 years', it is necessary to know when the 5 years end). Please enter the date in the following format: **YYYY-MM-DD**, e.g. 2000-12-30.
Owner	Name of the person/institution who owns the resource.
Publisher	Name of the publisher who is responsible for the distribution of the resource (i.e., if the data has been published).

Contact	Information about whom to contact in order to gain access to the resource.
Name	Name of the person to contact.
Address	Address of the person to contact.
E-mail	E-mail address of the person to contact.
Organization	Name of the organization to which the contact person belongs.
Descriptions	Prose description of the applied access restrictions.

! Note: The information entered into an Access schema is not immediately visible after you have closed and re-opened an IMDI file. The information is not lost - it will become visible once you click on the "Edit …" button.

! Note: There are bugs in the fields Date and Descriptions: the entered information will not be saved. Please ignore these fields for the moment.

Appendix 3: Lists of recommended vocabularies

This appendix contains lists of recommended vocabularies to be used in several of the screens of the IMDI Editor. And although there is room for variation, we recommend that you keep as closely as possible to the standardized lists as this will facilitate the search process.

1. Content: Tasks

This list contains a set of recommended names for experiments, elicitation tools and matching games commonly used by the Language and Cognition Group and the Acquisition Group at the Max Planck Institut für Psycholinguistik, Nijmegen (to be used on the screen Content/ Content Type/ Tasks). Note that the list is not closed, i.e., you can add other tasks.

name	comments
• "info-kiosk"	
• "wizard-of-oz"	
• "travel-planning"	
• "room reservation"	
• "frog story"	

2. Content: Modalities

This list contains a set of recommended labels for the modalities (to be used on the screen Content/ Content Type/ Modalities). Note that the list is not closed, i.e., you can add other modalities.

name	comments
• "speech"	
• "writing"	
• "gestures"	
• "pointing-gestures"	
• "signs"	
• "eye-gaze"	
• "facial-expressions"	
• "emotional-state"	
• "haptics"	

3. Content: CommunicationContext: Interactivity

This list contains a set of vocabularies that specifies the level of participant interaction (to be used on the screen Content/ CommunicationContext/ Interactivity). Note that this list is closed, i.e., you cannot add other items.

name	comments
• "interactive"	a verbal interaction between at least two participants. It may or may not include an investigator, e.g.: • conversation • many narratives • matching game
• "non-interactive"	a monologue, produced without expecting extended verbal responses from the hearer(s), e.g.: • many oratory texts and songs • some narratives • procedural texts
• "semi-interactive"	primarily a monologue, but punctuated by repeated interjections from the hearer(s), e.g.: • a child interrupting a narrative • hearer(s) repeatedly prompting a narrator

4. Content: CommunicationContext: Planning Type

This list contains a set of vocabularies that specifies the degree of planning through the consultant (to be used on the screen Content/ CommunicationContext/ Planning Type). Note that this list is closed, i.e., you cannot add other items.

name	comments
• "spontaneous"	an unprompted speech whose topic is not determined by the investigator or an observer, e.g.: • conversation • chatting • joke-telling • singing while harvesting
• "semi-spontaneous"	a prompted speech whose topic is determined in some way by an investigator or a community member, but whose participants speak freely within this context, e.g.: • interview • queries (e.g., 'Tell me about the history of your village.' 'Show me how to make tortillas.') • retellings (e.g., the speaker is asked to re-tell a story from a picture book, or to describe a task in his/her own words) • promptings (e.g., children answering a teacher's questions)
• "planned"	the structure and content of the speech is planned in advance by the consultant/performer, e.g.: • political or ritual speech • poem recitation

5. Content: CommunicationContext: Planning Type

This list contains a set of vocabularies that specifies the degree of involvement of the researcher (to be used on the screen Content/ CommunicationContext/ Involvement). Note that this list is closed, i.e., you cannot add other items.

- "elicited" — the investigator asks the speaker(s) to produce isolated phonemes, words, utterances or grammatical structures, e.g.:
  - production of sounds in different phonological environments
  - responses to (morphological, lexical) questionnaires
- "non-elicited" — the investigator does not interfere verbally with the speech event (other than with his presence)
- "no-observer" — no outside observer is present (only a tape recorder)

6. Content: Interactional Genre

This list contains a set of recommended names to refer to interactional genres (to be used on the screen Content/ Content Type/ Interactional). Note that the list is not closed, i.e., you can add other interactional genres.

name	comments
"conversation"	
"verbal-contest"	**!** Note: this also includes debates.
"interview"	
"meeting"	
"riddles"	
"consultation"	**!** Note: this does not refer to an interview with the investigator, but rather to, e.g. a visit to a shaman.
"greetings"	
"leavetakings"	
"humor"	

7. Content: Discursive Genre

This list contains a set of recommended names to refer to discursive genres (to be used on the screen Content/ Content Type/ Discursive). Note that the list is not closed, i.e., you can add other discursive genres.

name	comments
"procedure"	A directive description of the procedures involved in the preparation or production of something, e.g. 'how to make tortillas'.
"explanation"	Practical, theoretical, or historical reality statements, e.g. 'how the monkey got its tail'.

8. Content: Performance Genre

This list contains a set of recommended names to refer to performative genres (to be used on the screen Content/ Content Type/ Performance). Note that the list is not closed, i.e., you can add other performative genres.

name	comments
• "oratory"	Using speech effectively in a conventionalized format to address an audience within political, legal, ceremonial, or religious settings.
• "oral-history"	An account of firsthand experience, recalled retrospectively and communicated to an interviewer for historical purposes.
• "historical-narrative"	A secondhand account of the experience of historical figures and events, which may be partly or wholly fictional, communicated to both locals and outsiders for both historical purposes and entertainment.
• "narrative"	A recounting of one or more fictional events by one or more narrators to an audience of at least one.
• "oral-poetry"	Spoken or sung, in a relatively structured form (in prosody and syntax), often with distinctive language, e.g. oral epics, narrative poetry, ballads (shorter, lyrical narratives), and panegyric odes.
• "song"	A tune with recognizably structured lyrics, e.g. popular and love songs, lullabies.
• "proverb"	A summary of the wisdom of collective experience, often one line long; formulaic.
• "lament"	
• "insult"	An insolent verbal act creating animosity.

Appendix 4: Lists of languages and language abbreviations

This appendix lists some of the language names and identifiers (as they are used in the Ethnologue). Please use these standardized names and identifiers in all those fields of the IMDI Editor that ask for the name/identifier of a language. When a language name and identifier that you need is not in this list, please look it up under www.ethnologue.com/web.asp.

name	identifier	comment
American Sign Language	ASE	
Arabic, Moroccan (spoken)	ARY	
Arrernte, Eastern	AER	
Basque	BSQ	
Belhariya	BYW	Belhare
Bora	BOA	Mirana
Chontal de Oaxaca - Costa	CLO	
Czech	CZC	
Dutch	DUT	
Dutch Sign Language	DSE	
English	ENG	
Ewe	EWE	
Farsi (Western)	PES	
Finnish	FIN	
French	FRE	
Fulfulde Fuuta Jalon	FUF	
Galician	GLE	
German	GER	
German Sign Language	GSG	
Goemai	ANK	
Guguyimidjir	KKY	Guugu Yimithirr
Hausa	HUA	
Hebrew	HBR	
Hindi	HND	
Italian	ITN	
Inuktitut, Greenlandic	ESG	
Inuktitut, Eastern Canadian	ESB	
Japanese	JPN	
Kgalagadi	XKV	
Katang	KGD	
Khmer (Central)	KMR	
Kilivila	KIJ	
Korean	KKN	
Kota	KFE/KOQ?	
Lao	NOL	
Longgu	LGU	
Marquesan, North	MRQ	
Maya, Yucatán	YUA	Yukatek
Maya Mopán	MOP	
Nicaraguan Sign Language	NCS	

Popoluca, Oluta	PLO	
Portuguese	POR	
Punjabi		(not in Ethnologue)
Russian	RUS	
Saliba	SBE	
Samoan	SMY	
Sekpele	LIP	Likpe
Spanish	SPN	
Swedish	SWD	
Tamil	TCV	
Tidore	TVO	
Tongan	TOV	
Totonaca, Papantla	TOP	
Turkish Sign Language	TSM	
Trumaí	TPY	
Turkish	TRK	
Tzeltal, Oxchuc	TZH	
Tzeltal, Tenejapa	TZL	(not in Ethnologue)
Tzotzil, Zinacantán	TZZ	
Yalunka	YAL	
Yele	YLE	Yélî Dnye
Zapotec, Zoogocho	ZPQ	

Appendix 5: Digitization Policy

1. Contact dobesarc@mpi.nl with your digitization wishes. A digital master file (DMF) will be created from your master tape and placed in the appropriate network folder.
2. Go through the DMF and use the Media Player program to identify the relevant units of analysis (sessions). Make a note of where each session begins and ends (on the screen Resources/ Media Files/ Media File/ Time Position).
3. Create a metadata description file for each session on your tape by using the IMDI Editor.
4. Submit the metadata description to dobesarc@mpi.nl for the digital segmentation and conversion of your data.

> **!** Note: Only data that is accompanied by metadata description files will be further processed by the technical group. If you do not plan on meta-describing your data, the DMF will be saved on a DLT tape that will then be given to you for storage.

Appendix 6: Tape labeling conventions

A label, containing the ID, written in block letters, and readable by everyone must be put on the media. The ID has to contain the following characters.

- one character to identify the team or group:

  L (Language and Cognition)
  A (Acquisition)

- two characters to identify the researcher
- one character to identify the sub-project of the researcher

  Language and Cognition Group:
  S (Space)
  G (Gesture)
  E (Event)

  Acquisition Group:
  A (Argument Structure)
  F (Finiteness)
  S (Second Language Acquisition)

- three characters to identify the type:
  - First position:

    A (Audio)
    V (Video)
    P (Photo)
    C (CD)

  - Second position:

    in case of Audio:
    C (audio cassette)
    T (reel-to-reel-tape)
    D (DAT tape)
    M (minidisk)

    in case of Video:
    D (digital video)
    8 (Hi-8)
    S ((S)VHS)
    U (U-matic)

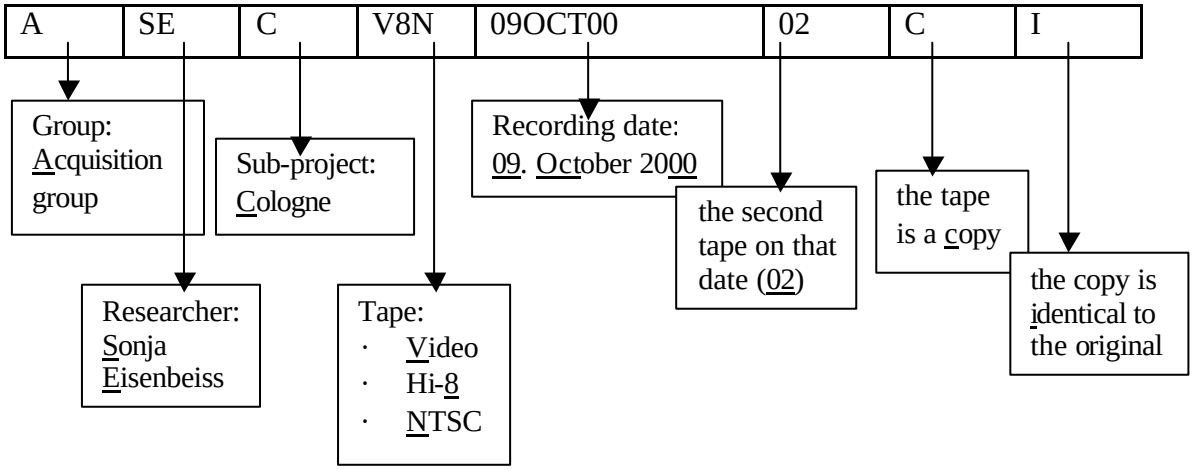
    in case of Photo:
    D (digital)
    A (analog)

  - Third position (in case of video): N=NTSC, P=PAL

    N (NTSC)
    P (PAL)

- seven characters to specify the date of recording: 2 digits, 3 letters and 2 digits, e.g. 09Oct00
- two digits to identify the number of the tape/photo related to the recording date
- one digit to indicate if it is a copy "C" (otherwise omit it)
- one digit to indicate if the copy is identical to the original "I" (otherwise omit it)

For Example, the tape label "ASECV8N09OCT0002CI" has the following meaning:



Appendix 2

IMDI Browser, version 1.4

Manual

This manual was last updated: 12 Sep 2002
The latest version of the manual can be downloaded from the following webpage: www.mpi.nl/tools
Author: Birgit Hellwig

Introduction

The IMDI (ISLE Metadata Initiative) Browser was developed at the Max Planck Institute for Psycholinguistics, Nijmegen, The Netherlands, with the aim of managing data and metadata files.

It supports the following features:

- accessing data through a hierarchical tree structure and a world map;
- linking of media, annotation and metadata files;
- direct access to media, annotation and metadata files;
- searching of metadata files.

This manual explains and exemplifies the features of the IMDI Browser. It is organized around the following three parts:

1. Basic information about the IMDI Browser (chapter 8).
2. Displaying and accessing data (chapter 9).
3. Searching data (chapter 10).

Notation Conventions

The following notation conventions are used:

- Menu items, icons and screen displays are written in the font MS Sans Serif.
- (SHORTCUT) KEYS ARE WRITTEN IN SMALL CAPS.
- Information on troubleshooting starts as follows: **!**

Table of Contents

8 Basic information

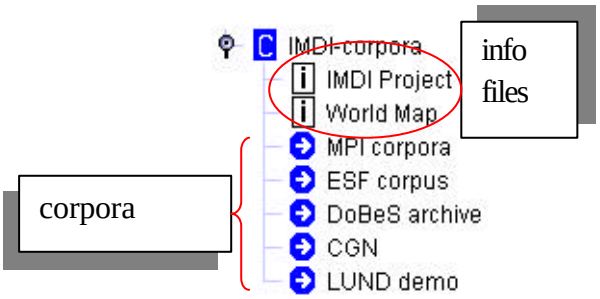
This section of the manual introduces you to the basic concepts and possibilities of the IMDI Browser. It is structured as follows:

1. Corpora (section 8.1)
2. Sessions (section 8.2)

8.1Corpora

The IMDI Browser gives access to all corpora that are managed through the Max Planck Institute, Nijmegen.

Currently, the following corpora are available:



MPI corpora	Corpora collected by researchers affiliated with the Max Planck Institute for Psycholinguistics.
ESF corpus	European Science Foundation Second Language Acquisition Database.
DoBeS archive	Corpora collected by researchers affiliated with the DoBeS (Dokumentation Bedrohter Sprachen) project of the Volkswagen foundation.
CGN	Corpus Gesproken Nederlands.[39]
LUND demo	Demonstration created for the beginning of the Year of the Language (2001) in Lund, Sweden.

! Note: Depending on the version of the IMDI Browser, it may not always be possible to access all corpora.

Each corpus contains further subcorpora collected by individual researchers or project teams. The internal structure of each subcorpus varies according to the purposes and needs of the project.

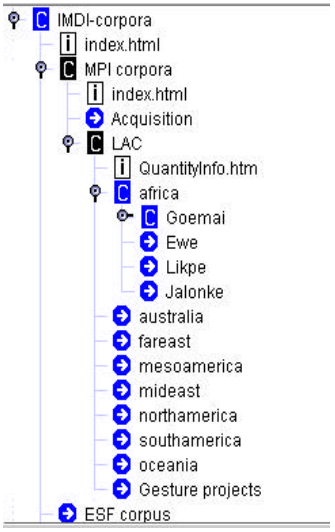
[39] If you access the Corpus Gesproken Nederlands via the IMDI Browser, content searches are not possible. To conduct content searches in the CGN corpus, you need the corresponding CDs (see the separate manual "Corpus Gesproken Nederlands (COREX)" for details).

Information about the content and structure of each corpus and subcorpus can be found in info files that are accessed through the IMDI Browser (see section 9.3.4.4).

Each corpus contains metadata, media, annotation and info files (see section 9.3.4). These files can be accessed by two mechanisms:

(a) By navigating through the hierarchical tree structure (see sections 9.3.1 and 9.3.2).

Each corpus is organized hierarchically in form of a tree structure (see the illustration to the right).



The files are grouped together based on their content, the speakers involved and/or the languages spoken. For example, they are grouped under nodes such as the discourse genre, the sex or age of the speaker, the dialect of the speaker, the target/source language etc.

All nodes are displayed in the IMDI Browser, allowing you to navigate through the hierarchy.

! Note: The tree structure does **not** display the physical location of files. This means that the same file can be displayed in different parts of the corpus, e.g., the same file may be displayed under the nodes 'folktale', 'male speaker', and 'age-group 20 to 30 years'.

(b) By searching the metadata files (see section 10).

All data files are accompanied by metadata files, i.e., files that give information about the data. These metadata files are searchable. For example, as illustrated below, you could search for all texts by female speakers above 60 years of age. The IMDI Browser displays the search results, and allows you to directly access them.

! Note: Data files may not always be accessible (see section 8.2).

! Note: If you would like to integrate your corpus into the IMDI Browser, please contact corpus.manager@mpi.nl who will advise you on the procedure.

8.2Sessions

The session is the lowest level of the corpus hierarchy. A session contains data that has the same overall content, the same set of participants, and the same location and time. For example, one session could correspond to one elicitation session on a specific topic or to one particular narration of a folktale.

A session can contain the following four types of files:

(1) Metadata files (see section 9.3.4.1).

These files contain information about the session, e.g., its date and location, its content and its participants. They are of the IMDI Editor format (see the separate manual "IMDI Editor" for details).

(2) Media files (see section 9.3.4.2).

These files contain audio or video recording of the session. They are digitized in one of the following formats: MPEG (***.mpg**), Cinepak-Quicktime-Movies (***.mov**), WAVE (***.wav**).

(3) Annotation files (see section 9.3.4.3).

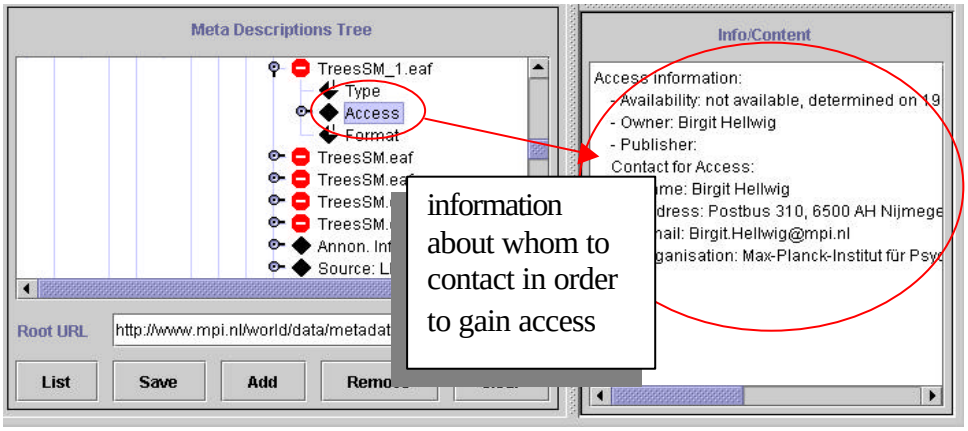
These files contain the transcripts, codings and annotations of the session. Their format varies (e.g., ELAN, Media Tagger, Shoebox, CHAT, etc.)

(4) Info files (see section 9.3.4.4).

These files contain further background information on specific topics. They are in PDF or HTML formats.

Metadata and info files are generally accessible to everybody – although sensitive information may be suppressed (see the separate manual "IMDI Editor" for details). This policy enables researchers to search metadata and to thereby gain an overview of the available material.

Annotation and media files are handled differently in the different corpora: some corpora allow for general access (e.g., the CGN corpus), while others do not (e.g., the MPI corpora). In the latter case, if you have not been explicitly granted access by the responsible researcher(s), you will not be able to access them. To gain access, ask the responsible researcher(s) for permission. Contact details are displayed in each corpus, e.g.:



The responsible researcher(s) have unrestricted access to all files. This includes the following possibilities:

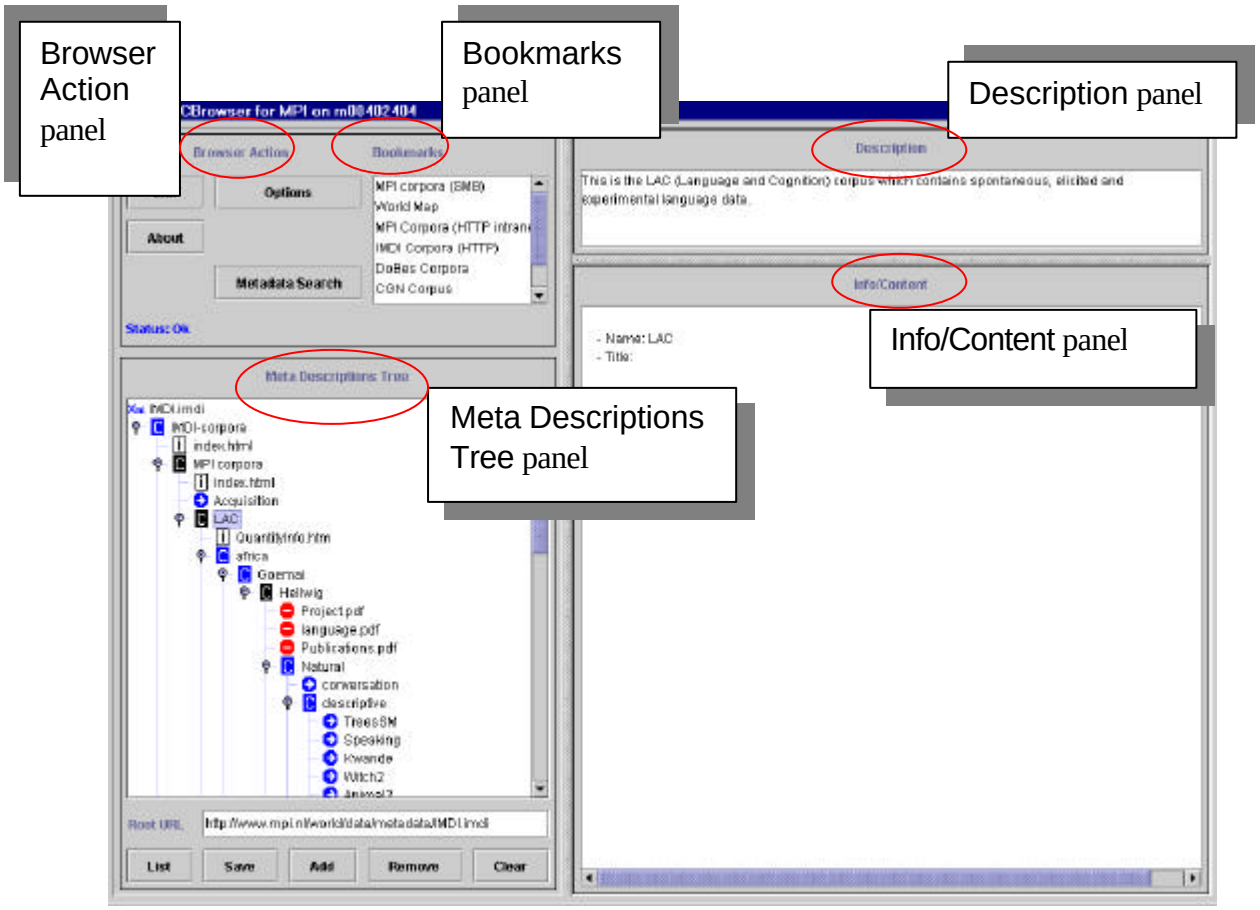
- Access to metadata and info files, including (a) all suppressed information and (b) the possibility to update the information. See sections 9.3.4.1 and 9.3.4.4.
- Access to media files. See section 9.3.4.3.
- Access to annotation files, including the possibility to continuously update them. See section 9.3.4.3.

9 Displaying and accessing data

Starting the IMDI Browser opens up the window IMDI-BCBrowser for MPI (referred to in this manual as the IMDI Browser window). In the IMDI Browser window, you can view and access the corpora, i.e., you can read information about the available data, access the files, and initiate searches. The IMDI Browser window contains the following five panels:

1. Browser Action panel (section 9.1)
2. Bookmarks panel (section 9.2)
3. Meta Descriptions Tree panel (section 9.3)
4. Info/Content panel (section 9.4)
5. Description panel (section 9.5)

These panels are described in the following sections.



9.1Browser Action panel

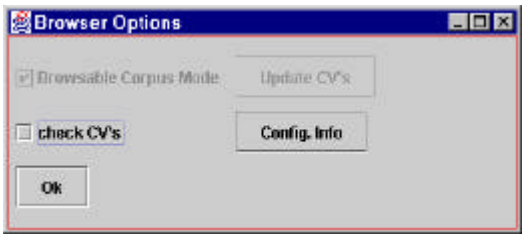
The Browser Action panel offers the following options:

Exit/Close — Click the Exit button to exit the IMDI Browser.

Click the Close button to close the current IMDI Browser window. This button is only available when you have opened a second IMDI Browser window, either through the option Clone Node (see section 9.3.1) or through accessing search results (see section 10.4).

About — Click this button to view the copyright and version information.

Options — Click this button to set options. The following options are available:



- Check CV's: checks the values in the metadata description files against the allowed values. Please ignore this option.
- Config. Info: displays information about the configuration. You might need this information when reporting a bug to the developers.

Metadata Search — Click this button to gain access to the IMDI Metadada Search window (see section 10).

9.2 Bookmarks panel

In the Bookmarks panel, you can save shortcuts, i.e., "bookmarks", to parts of the corpus. Such bookmarks have the advantage that you do not need to navigate through the entire corpus hierarchy to access parts of the corpus.

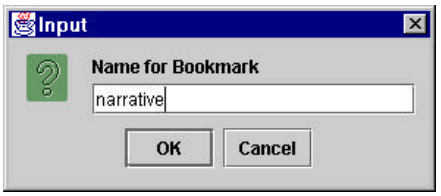
Depending on the version of the IMDI Browser, all or some of the following bookmarks are displayed:

MPI corpora (SMB)	Access to the MPI and ESF corpora (via the local network).
World Map	Access to a world map displaying the locations of documented languages (see section 9.3.2).
MPI Corpora (HTTP intranet)	Access to the MPI and ESF corpora (via the web server).
IMDI Corpora (HTTP)	Access to the MPI, ESF, DoBeS and CGN corpora (via the web server).
DoBeS Corpora	Access to the DoBeS corpora.
CGN Corpus	Access to the CGN corpus.
Search Results	Access to your search results (see section 10.5).

When you double-click on a bookmark, the corresponding node is displayed in the Meta Descriptions Tree panel (see section 9.3).

In addition to the predefined bookmarks, you can create your own bookmarks. Do the following:

1. In the Meta Descriptions Tree panel (see section 9.3), navigate to the corpus or session node for which you want to create a bookmark.
2. Click on the node to select it. It will be highlighted in blue color.
3. Click with the right mouse button on the node to open a pull-down menu.
4. Select Add to Bookmarks from the pull-down menu. The Input dialog box appears.



5. Specify a name for the bookmark.

   The new bookmark is added to the Bookmarks panel.

   The bookmarks are saved permanently by the IMDI Browser, i.e., they remain available every time you restart the Browser.

To remove a bookmark, do the following:

1. In the Bookmarks panel, click on the bookmark that you want to remove. It will be highlighted in blue color.
2. Click with the right mouse button on the bookmark to open a pull-down menu.
3. Select Delete Bookmark from the pull-down menu.

   The bookmark is deleted without further warning

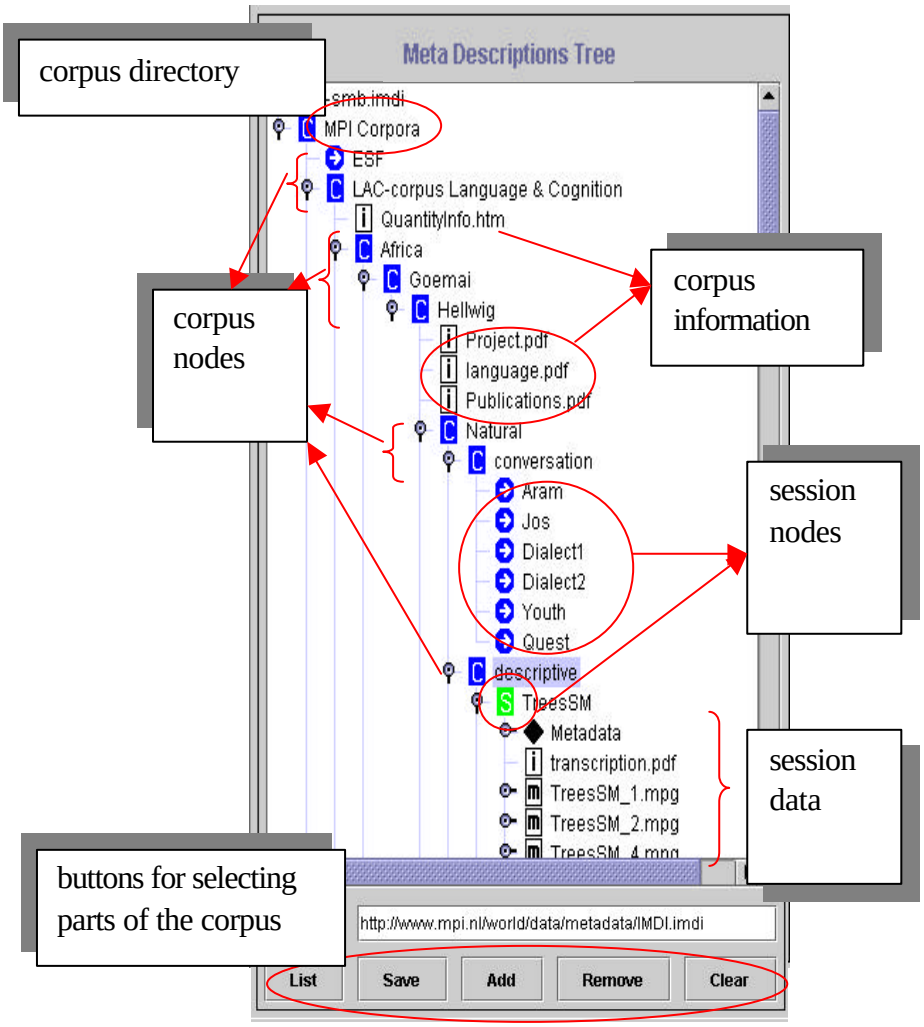
> **!** Note: You can only delete bookmarks created by yourself, but not the predefined bookmarks.

9.3The Meta Descriptions Tree panel

The Meta Descriptions Tree panel allows you to navigate through the corpus hierarchy. It serves the following three purposes:

1. The hierarchy of nodes allows you to easily navigate through the corpus to the session data (sections 9.3.1 and 9.3.2).
2. Pull-down menus and the five buttons at the bottom of the panel allow you to select parts of the corpus to be searched (section 9.3.3).
3. The linking of metadata, media and annotation files at the session level allows you to immediately access all relevant data (section 9.3.4).

The following screenshot illustrates the Meta Descriptions Tree panel.



9.3.1 Navigating through the corpus (tree structure)

Whenever the IMDI Browser window opens up, the node IMDI Corpora is automatically displayed in the Meta Descriptions Tree panel.

If you want to access a predefined part of the corpus, do the following:

1. Go to the Bookmarks panel (see section 9.2).
2. Double-click on one of the bookmarks.

   The corresponding node is displayed in the Meta Descriptions Tree panel.

In the Meta Descriptions Tree panel, make use of the following options to navigate through the corpus:

- Point with the mouse to any node to activate the on-line help that displays brief information about the node.
- Double-click on any node to open it and display the next level in the hierarchy.

> **!** Note: The Meta Descriptions Tree panel distinguishes between open and closed corpus and session nodes. These are represented through the following icons:
>
> 
>
> icon of an open corpus/session
>
> 
>
> icon of a closed corpus/session
>
> Some program commands do not work when a node is closed. If any of the commands do not seem to work, make sure that the node is open.

> **!** Note: Because of the large amount of data that is loaded, it may take some time until the IMDI Browser responds to your command and opens a node.

- Click on any node to select it. It will be highlighted in blue color.

  The information relevant to the selected node is displayed in the Info/Content (see section 9.4) and Description panels (see section 9.5).
- Right-click on any selected (i.e., highlighted) node to open a pull-down menu that displays the available options. Click on any item in the pull-down menu to select the corresponding option.

  The following options are available:

Add to basket	Adds the node to the list of nodes to be searched (see section 9.3.3).
Add to Bookmarks	Adds the node to the Bookmarks panel (see section 9.2).
Bookmark Info	Displays information about the bookmark in the Info/Content panel (see section 9.4).
Clone Node	Opens a second IMDI Browser window that displays only the corresponding node.
Delete Bookmark	Deletes the node from the Bookmarks panel (see section 9.2).
ELAN	Opens the media file in ELAN.
IMDI-BCEditor	Opens the metadata file in the IMDI Editor.

List Sessions	Lists all sessions contained under this node in the Info/Content panel (see section 9.4).
MetaData Search	Opens the IMDI Metadata Search window (see section 10).
Remove	Removes the node from the Meta Descriptions Tree panel.
Remove from basket	Removes the node from the list of nodes to be searched (see section 9.3.3).
Save File Content	Downloads the file in a non-compressed format.
Session Count	Displays the number of sessions contained under this node in the Info/Content panel.
Show Content-Type	Displays the file format in the Description panel (see section 9.5).
Show Description	Displays the file description in the Description panel (see section 9.5).
Show File Content	Displays the XML file of the node in the Info/Content panel (see section 9.4).
Show format	Displays the file format in the Info/Content panel (see section 9.4).
Show Info	Displays the file content in the Info/Content panel (see section 9.4).
Show LR's	Displays the directory information for all files contained under this node in the Info/Content panel (see section 9.4).
Show SearchService	Displays the name and directory information of the file containing the corresponding metadata information (see section 10.1).
Show Text Content	Displays the file content in the Info/Content panel (see section 9.4).
Show URL	Displays the directory information for the file in the Info/Content panel (see section 9.4).
Signal	Opens the media file in TK Signal.
Windows Media Player	Opens the media file in Windows Media Player.

! Note: For some nodes, one of the options is marked with the symbol #. In this case, the corresponding option will start automatically whenever you double-click on the open node.

! Note: Not all options are available for all nodes.

9.3.2 Navigating through the corpus (world map)

The IMDI Browser displays a world map with the locations of all languages that are documented in the corpora. To access the world map, do the following:

1. In the Bookmarks panel double-click on the bookmark World Map.

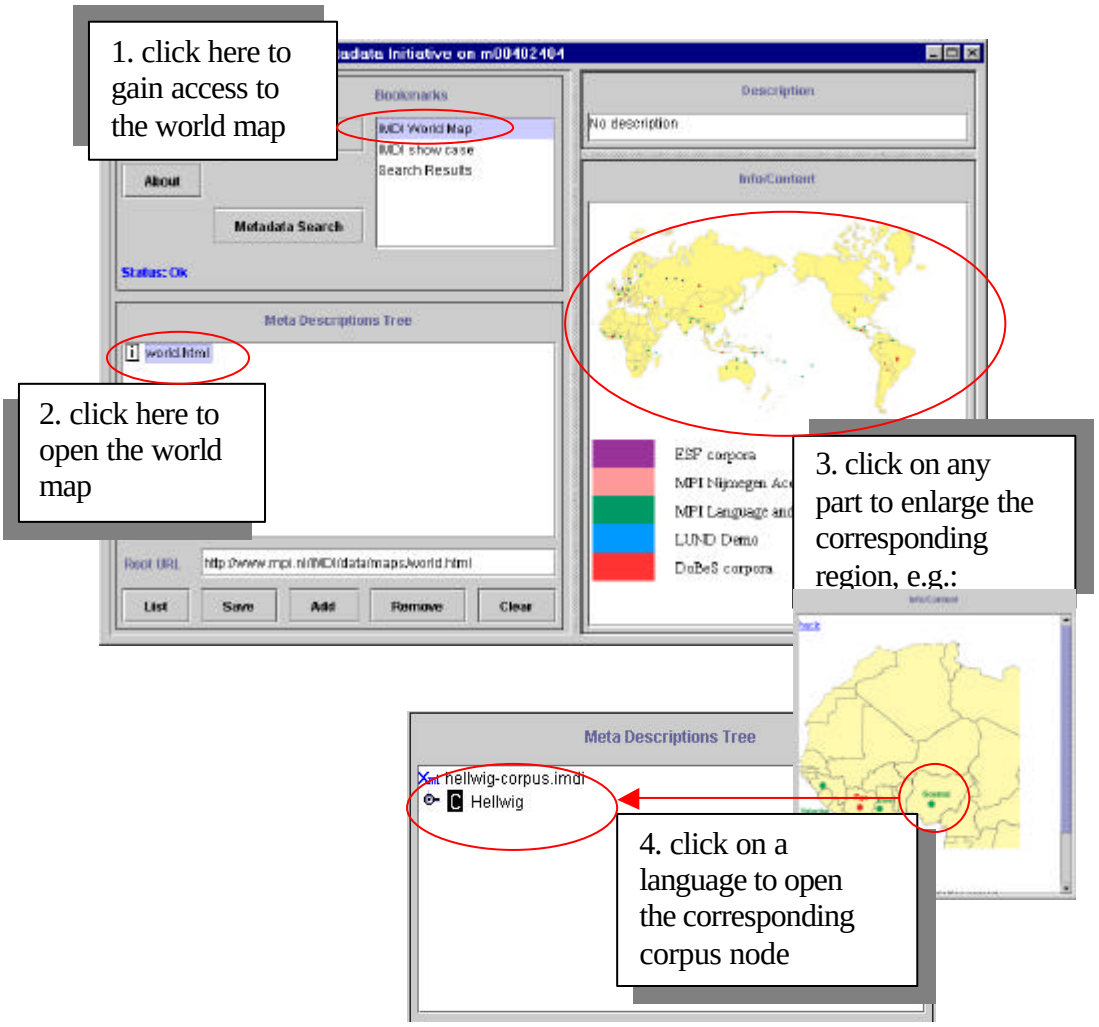
   The info file **world.html** is displayed in the Meta Descriptions Tree panel.

2. In the Meta Descriptions Tree panel click on the info file **world.html**.

   The Info/Content panel displays the world map.

3. Click on any part of the world map to enlarge the corresponding region.

4. Click on any language to open the corresponding node in the Meta Descriptions Tree panel.



9.3.3 Selecting parts of the corpus for purposes of analysis

The IMDI Browser allows you to search the metadata files (see section 10). By default, the search is done throughout the entire corpus, but it is possible to limit it to a subpart: to one (or several) selected corpus and/or session nodes.

To select a corpus or session node, do the following:

1. In the Meta Descriptions Tree panel, click on the node to select it. It will be highlighted in blue color.
2. Do one of the following:
   (a) Click the Add button at the bottom of the Meta Descriptions Tree panel.
   (b) Or click with the right mouse button on the highlighted node; then select Add to basket from the pull-down menu.

   The icon of any selected node will change its color to gray, e.g.:

non-selected node

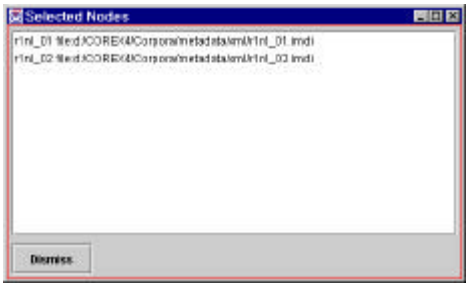
selected node

   Once an item is selected, the List button at the bottom of the panel will be highlighted in red color.
3. Repeat this process to add other nodes to your selection.

! Note: You can only select nodes that are open. Double-click on them to open them.

! Note: Select only those nodes that belong to a corpus for which there is a file containing all metadata information (see section 10.1).

Click the List button at the bottom of the Meta Descriptions Tree panel to view a list of all selected nodes, e.g.:



! Note: Whenever you have made a selection, metadata search will only search the listed nodes (see section 10.1). If the search process does not yield the expected results, please make sure that the list contains all relevant nodes.

To remove a node from the list, do the following:

1. In the Meta Descriptions Tree panel, click on the node to select it. It will be highlighted in blue color.
2. Do one of the following:
   (a) Click the Remove button at the bottom of the Meta Descriptions Tree panel.
   (b) Or click with the right mouse button on the highlighted node; then select Remove from basket from the pull-down menu.
3. Repeat this process to remove other nodes from your selection.

You can remove all selected nodes from this list by clicking the Clear button at the bottom of the Meta Descriptions Tree panel.

You can save the selected list for future uses. Click the Save button at the bottom of the Meta Descriptions Tree panel. The following message informs you that your list has been saved:

! Note: Once you have saved a selected list, you can only remove it by first clicking the Clear button (to remove all selected nodes) and then the Save button (to save an empty list).

When you are satisfied with your selection, you can initiate the search (see section 10).

9.3.4 Accessing data

The IMDI Browser contains links to a number of different files. These links are symbolized through the following icons:


link to metadata file(s) of a session (i.e., IMDI files).


link to digitized media file(s), containing audio/video data of a session.


link to transcription and annotation file(s) of a session.


link to info file(s) providing general background information.


link not available (either because the file does not yet exist, or because access is denied).

Click on any of these icons, and the IMDI Browser will display the relevant information.

9.3.4.1 Metadata information

Metadata information is displayed under each session node. It allows you to (a) read about the circumstances under which the session data was collected and (b) search for relevant data (see section 10).

It contains the following kind of information:

(a) Information about the date and location.

(b) Information about the project within which the data was collected.

(c) Information about the person who collected the data.

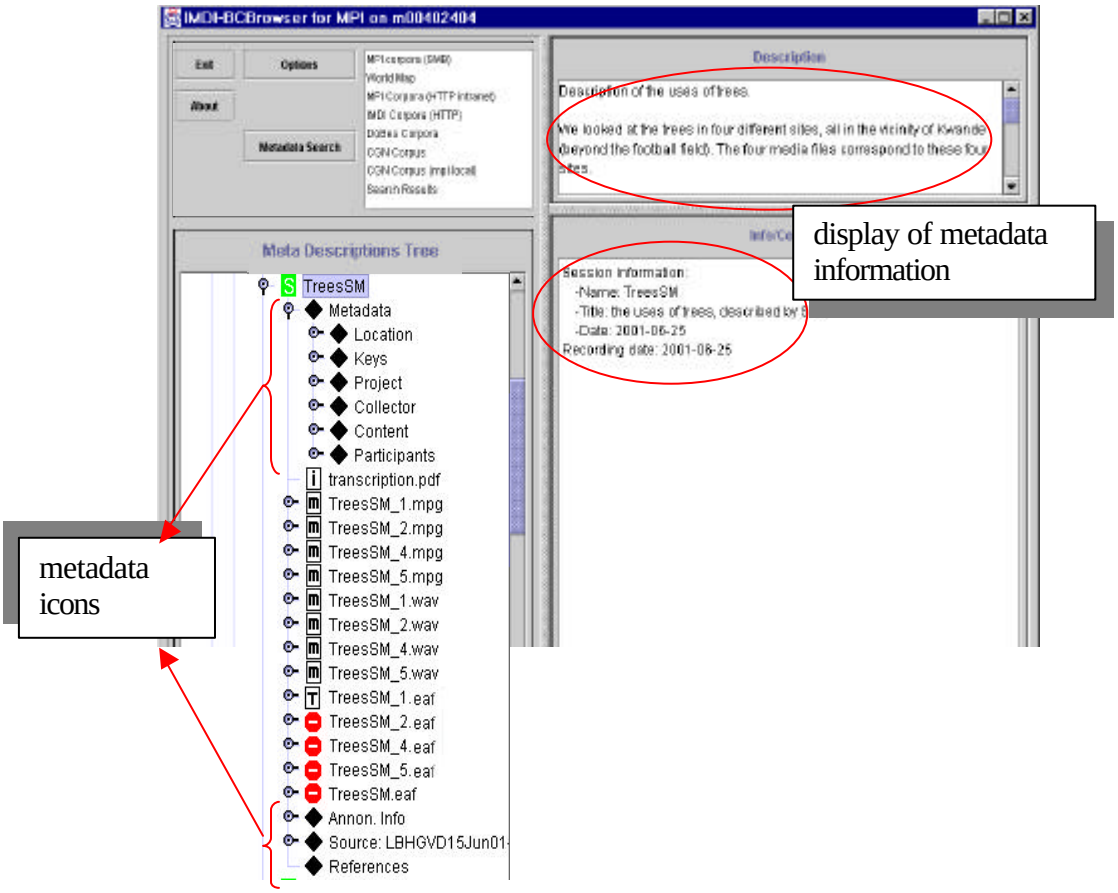
(d) Information about the content.

(e) Information about the participant(s).

(f) Information about the source (i.e., the audio/video tape), the digitized media file(s) and the annotation file(s).

(g) Cross-references to other relevant sessions and publications.

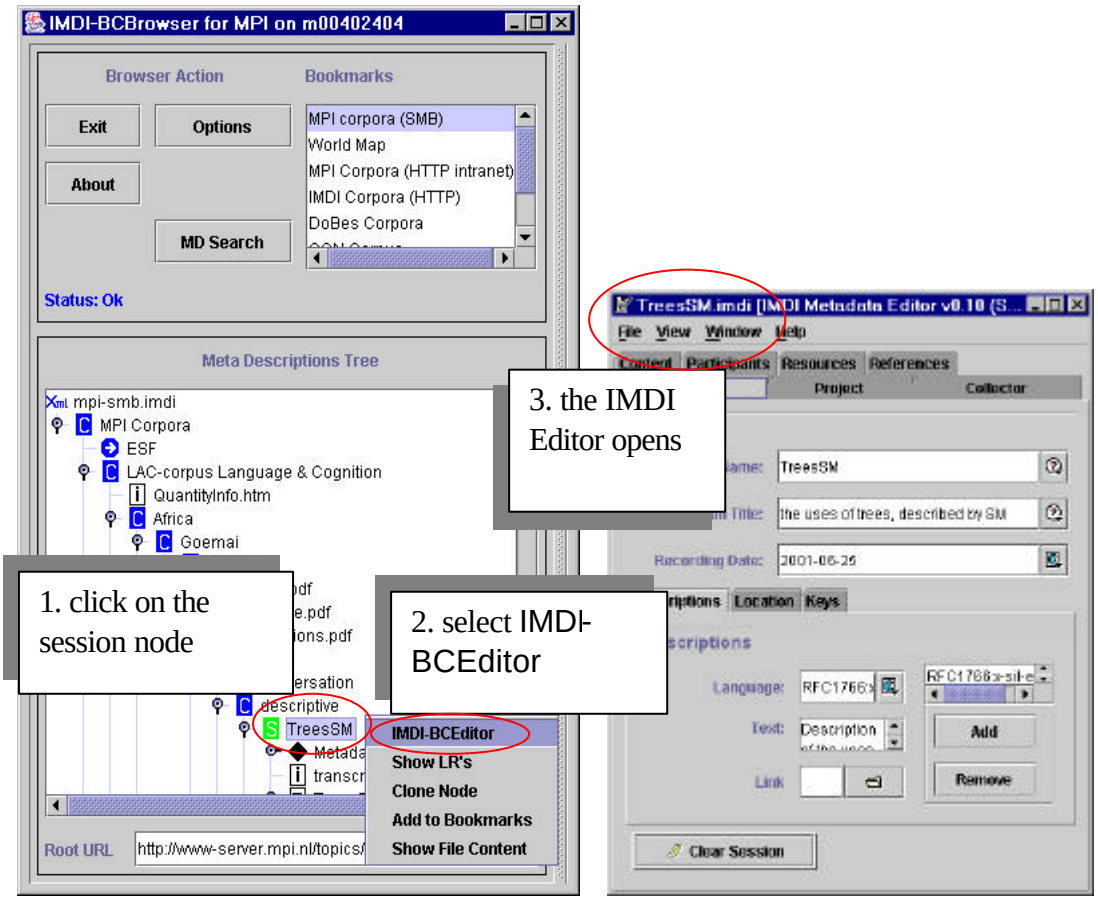
To view the metadata information of a session, click on the corresponding metadata icon. The information is displayed in the Info/Content panel (see section 9.4) or the Description panel (see section 9.5).



You can update the information in a metadata file. Do the following:

1. Double-click on the session node to open it.
2. Click on the open session node to select it. It will be highlighted in blue color.
3. Right-click on the selected session node to open a pull-down menu.
4. Select IMDI-BCEditor from the list.

   The IMDI Editor opens with the metadata file of the session. You can update the information (see the separate manual "IMDI Editor").



! Note: This option enables you to add information to existing metadata files. It is possible to change existing information, but please keep the following points in mind:

Do **not** change the names of media, annotation or info files.

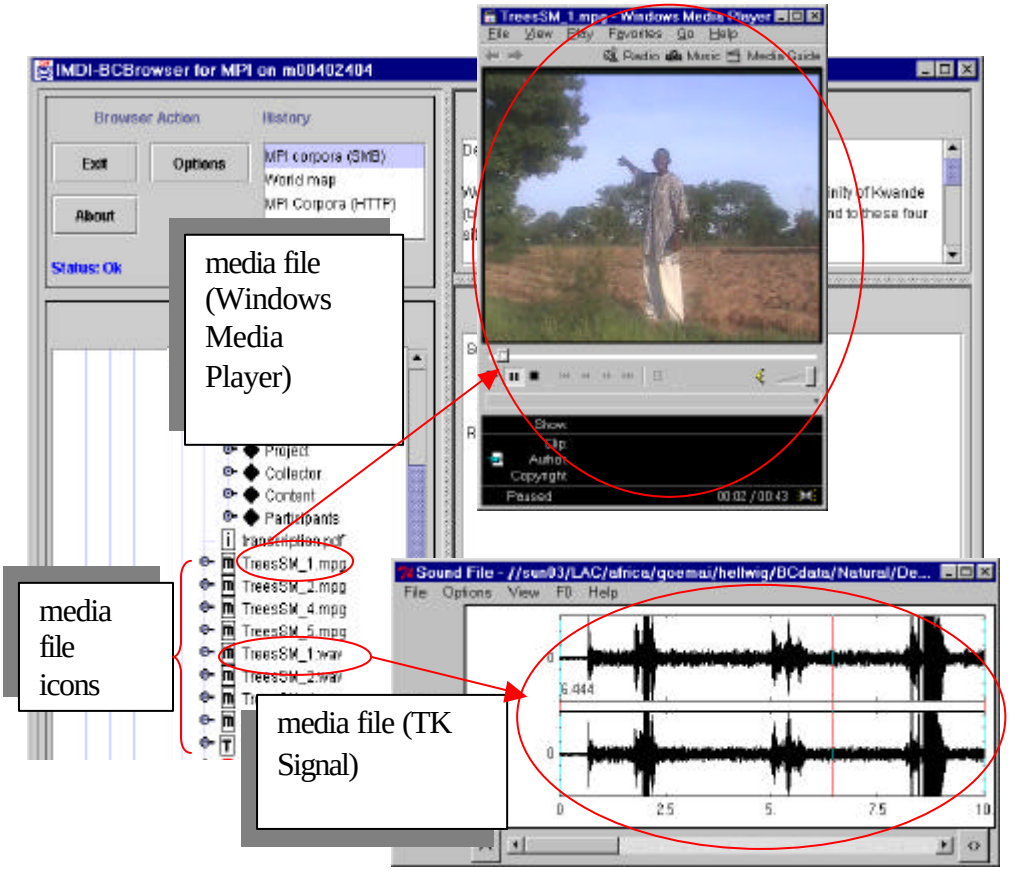
Before you make any substantial changes, or if you have doubts about changes, contact corpus.manager@mpi.nl to make sure that your changes do not interfere with the overall corpus management.

9.3.4.2 Digitized media files

A session node usually contains links to digitized media files, i.e., to audio and/or video files. To access a media file, do one of the following:

(a) Right-click on the media icon, and choose ELAN, Media Player or Signal from the pull-down menu. The media file will be opened in ELAN, Windows Media Player or TK Signal.

(b) Or double-click on the media file icon. By default, the IMDI Browser will open the file either in Windows Media Player (for video files) or TK Signal (for audio files).



9.3.4.3 Annotation files

A session node usually contains links to annotation files. To access an annotation file, do one of the following:

(a) If it is an ELAN file (***.eaf**):

- Either double-click on the annotation icon.
- Or right-click on the annotation icon; then select ELAN from the pull-down menu.

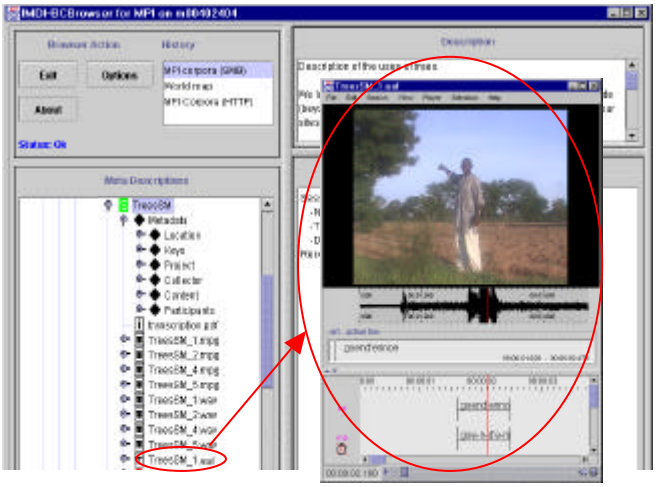
In both cases, the annotation file is opened in ELAN.

(b) If it is in any other format:

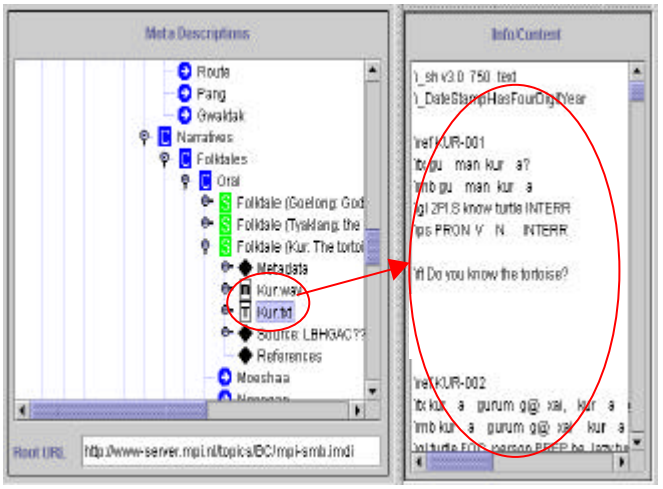
- Either double-click on the annotation icon.
- Or right-click on the annotation icon; then select Show Text Content from the pull-down menu.

In both cases, the annotation file is displayed in the Info/Content panel.

display of annotation file in ELAN:



display of annotation file in the Info/Content panel:



! Note: There are no restrictions on the format of your annotation files (ELAN, Media Tagger, Shoebox, CHAT, etc.).

! Note: You can constantly update your annotation files (provided you do not change their file names), which allows you to reflect changes in your analysis. But note that, for the moment, only ELAN can be directly accessed via the IMDI Browser. If you want to edit annotation files of a different format, please navigate to the physical location of that file (by using, e.g., Windows Explorer).

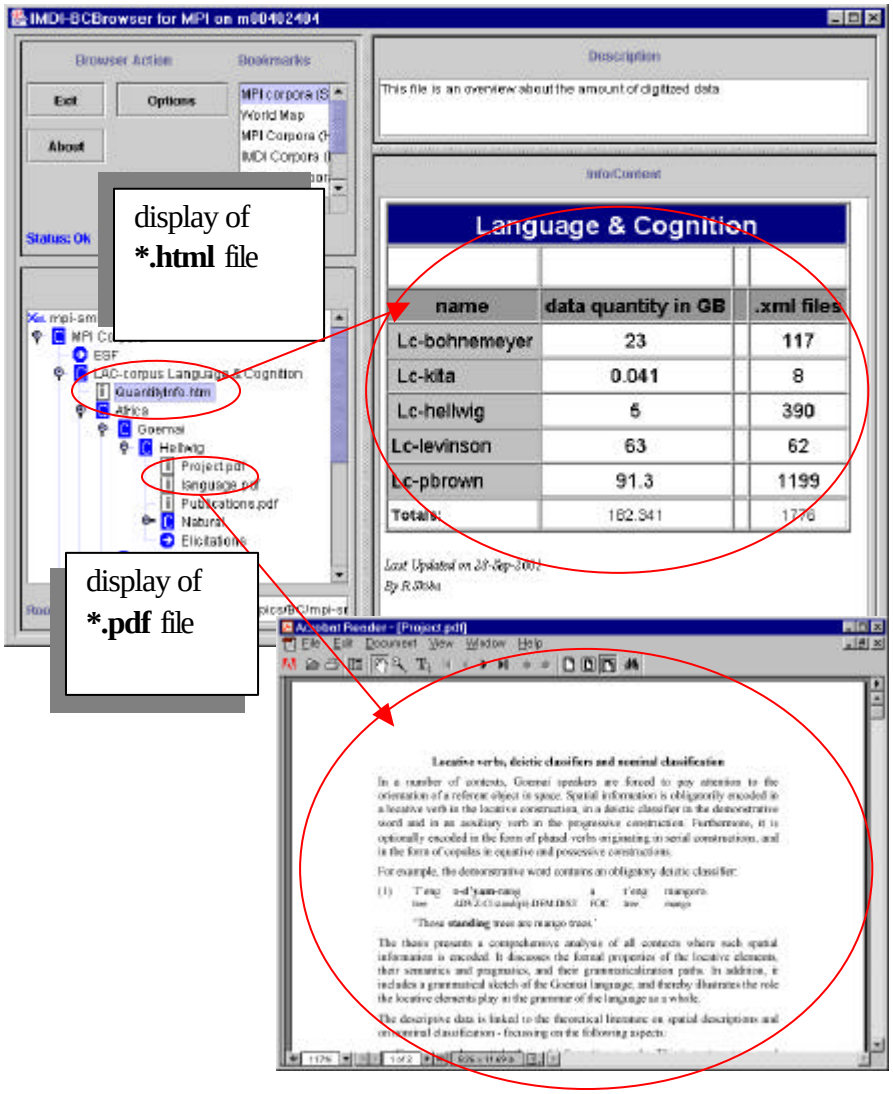
9.3.4.4 Info files

Both corpus and session nodes can contain links to info files. These files provide general background information on the corpus or additional information on the individual session.

To access such files, do one of the following:

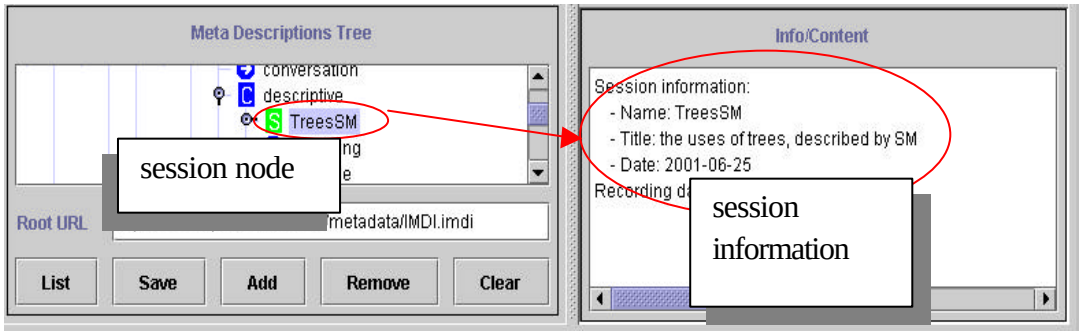
(a) If the file is in an HTML (***.html**) format, click on the info file icon. The file will be opened in the Info/Content panel.

(b) If the file in a PDF format (***.pdf**), double-click on the info file icon. Acrobat Reader will open up to display the file.



9.4Info/Content panel

The Info/Content panel displays information about corpus, session and file nodes. To read the information, click on the corresponding icon in the Meta Descriptions Tree panel, e.g.:



! Note: The information is only displayed when the node is open. Double-click on its icon to open it.

! Note: The information is displayed in form of searchable keywords. For prose information, look at the Description panel (see section 9.5).

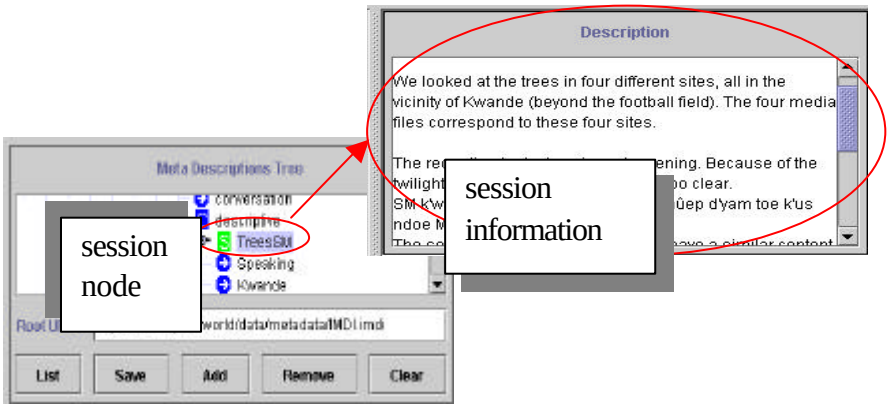
The Info/Content panel can display additional information. To view this information, do the following:

1. In the Meta Descriptions Tree panel, click on the node whose information you want to view. It will be highlighted in blue color.
2. Right-click on the highlighted node.
3. Select one of the following options from the pull-down menu (see section 9.3.1).
   - Bookmark Info: Displays information about the bookmark.
   - List Sessions: Lists all sessions contained under this node.
   - Session Count: Displays the number of sessions contained under this node.
   - Show File Content: Displays the XML file.
   - Show format: Displays the file format.
   - Show Info: Displays the file content.
   - Show LR's, Show URL: Displays the directory information for all files contained under this node.
   - Show Text Content: Displays the file content.

   The corresponding information is displayed in the Info/Content panel.

9.5Description panel

The Description panel displays a description of corpus, session and file nodes. To display the description, click on the corresponding icon in the Meta Description Tree panel, e.g.:



! Note: The information is only displayed when the node is open. Double-click on its icon to open it.

! Note: The Description panel displays prose information. For searchable keywords, look at the Info/Content panel (see section 9.4).

The Description panel can display additional information. To view this information, do the following:

1. In the Meta Descriptions Tree panel, click on the node whose information you want to view. It will be highlighted in blue color.
2. Right-click on the highlighted node.
3. Select one of the following options from the pull-down menu (see section 9.3.1).
   - Show Content-Type: Displays the file format.
   - Show Description: Displays the file description.

   The information is displayed in the Description panel.

10Searching data

All metadata information can be searched. To conduct a search, you need to access the IMDI Metadata Search window.

To access the IMDI Metadata Search window, do one of the following:

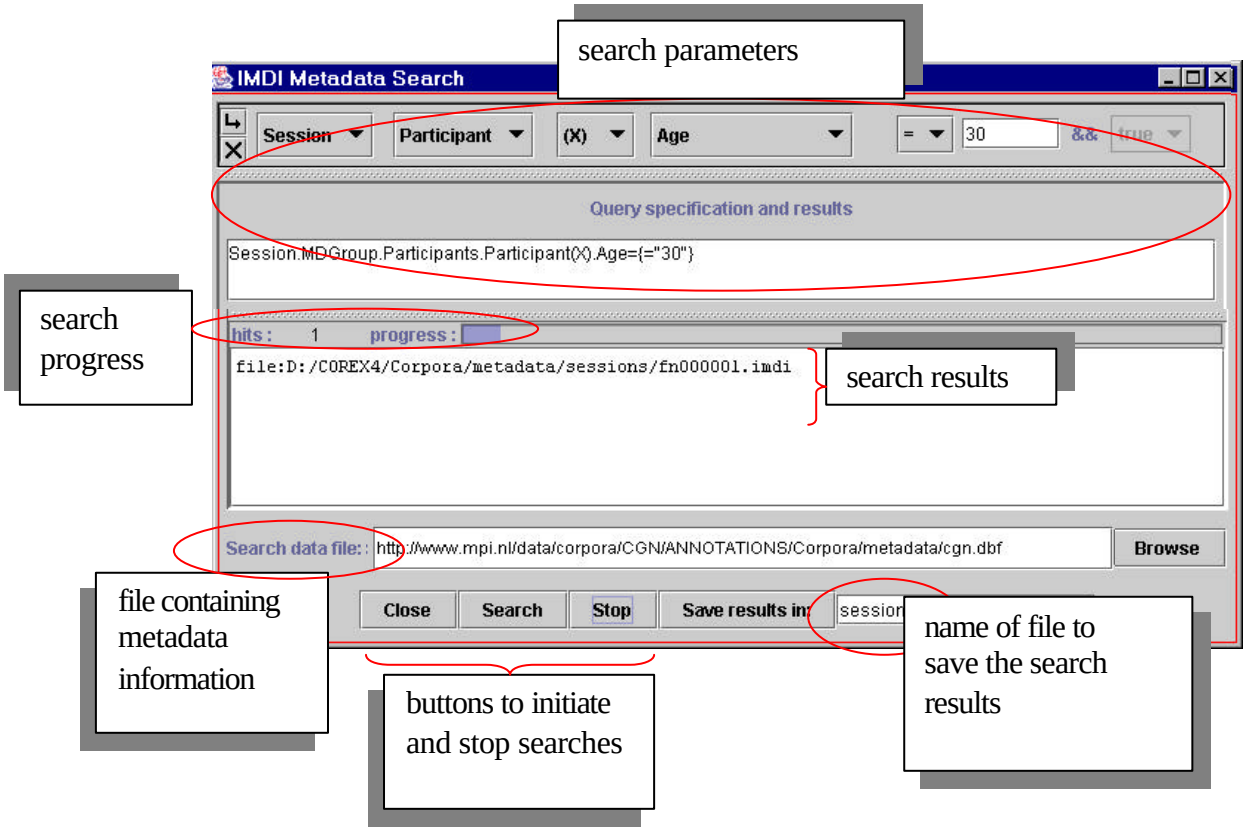
(a) Use the Browser Action panel (see section 9.1).

1. Go to the Browser Action panel.
2. Click on Metadata Search.

(b) Use the Meta Descriptions Tree panel (see section 9.3).

1. Go to the Meta Descriptions Tree panel.
2. Click on a node to select it. It will be highlighted in blue color.
3. Right-click on the node to open a pull-down menu.
4. Select MetaData Search from the pull-down menu.

In both cases, the IMDI Metadata Search window appears, e.g.:



This section introduces the search options. The following options are available:

1. Specify the corpus to be searched (section 10.1).
2. Specify the search parameters (section 10.2).
3. Initiate and stop the search (section 10.3).
4. Display the search results (section 10.4).
5. Save the search results (section 10.5).
6. Exit the IMDI Metadata Search window (section 10.6).

10.1 Specify the corpus to be searched

Metadata searches can be conducted throughout an entire corpus, or through selected parts of a corpus.

(a) Searches throughout an entire corpus.

To search an entire corpus, do the following:

1. In the Meta Description Tree panel (see section 9.3), double-click on the corresponding corpus node to open it.

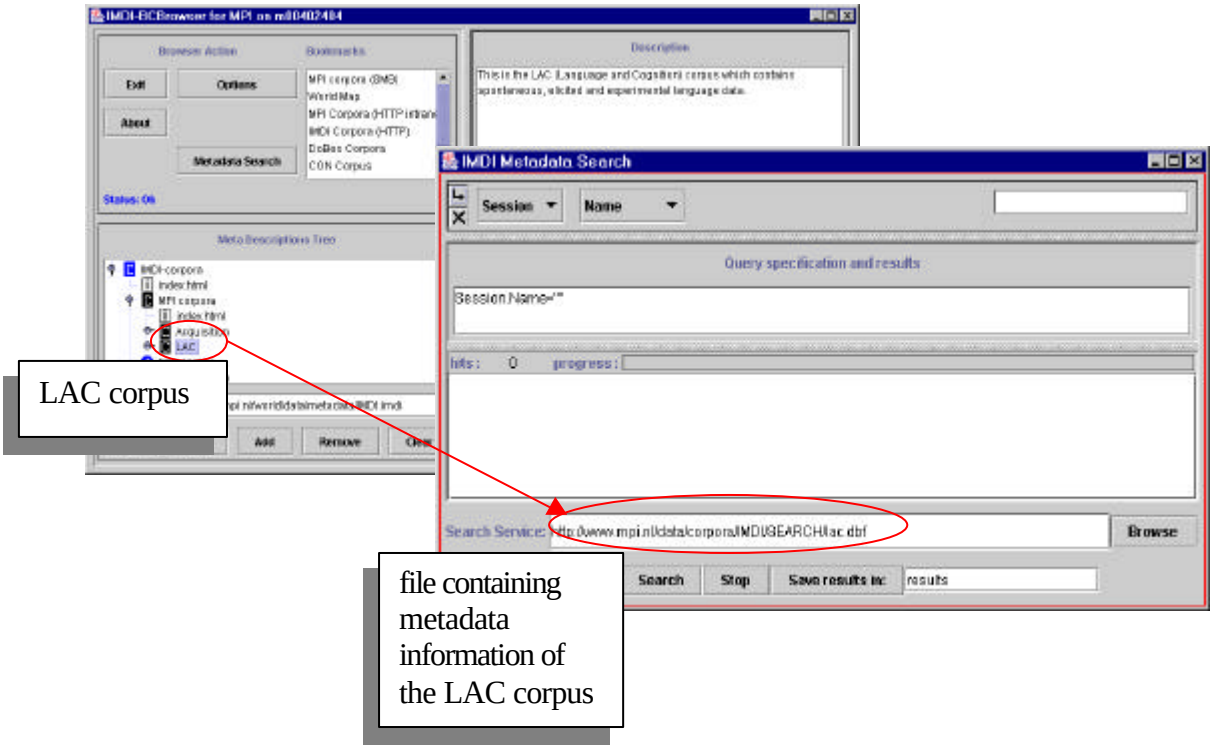
   Click on the node to select it. It will be highlighted in blue color.

2. Do one of the following:

   (a) In the Browser Action panel (see section 9.1), click on Metadata Search.

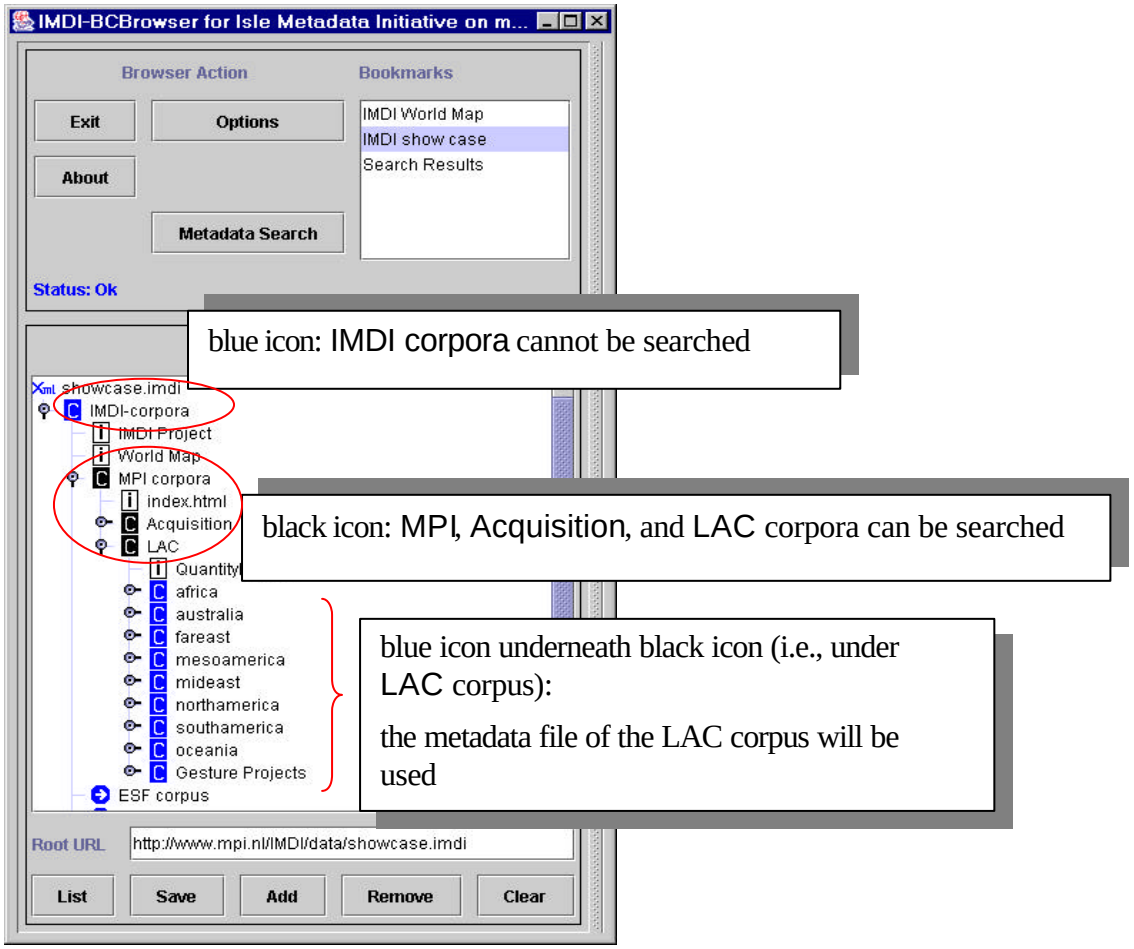
   (b) Or in the Meta Description Tree panel (see section 9.3), right-click on the selected node; then select MetaData Search from the pull-down menu.

   In the IMDI Metadata Search window, the file that contains the metadata information of the highlighted corpus is automatically entered into the box to the right of Search Service.



! Note: Corpora can only be searched if there is a corresponding file that contains all metadata information. Such corpora are symbolized through a white C on a black background. Corpora that have no corresponding metadata file are symbolized through a white C on a blue background.

You can select a subcorpus that is part of a larger corpus. In this case, the Search Service box will automatically display the metadata file corresponding to the larger corpus.



If the search process does not yield the expected results, please check the following possibilities:

(1) Make sure that you have not accidentally selected a subpart of the corpus (see below). If you have selected a subpart of the corpus, the List button at the bottom of the Meta Descriptions Tree panel appears in red color.

(2) Make sure that the metadata file displayed next to Search Service is the correct file. The box always displays the file that corresponds to the highlighted node in the Meta Descriptions Tree panel.

(b) Searches throughout subparts of a corpus.

You can search a subpart of the corpus. In this case, you have to select the nodes to be searched.

To select nodes, make use of the buttons Add, Remove and Clear at the bottom of the Meta Descriptions Tree panel (see section 9.3.3).

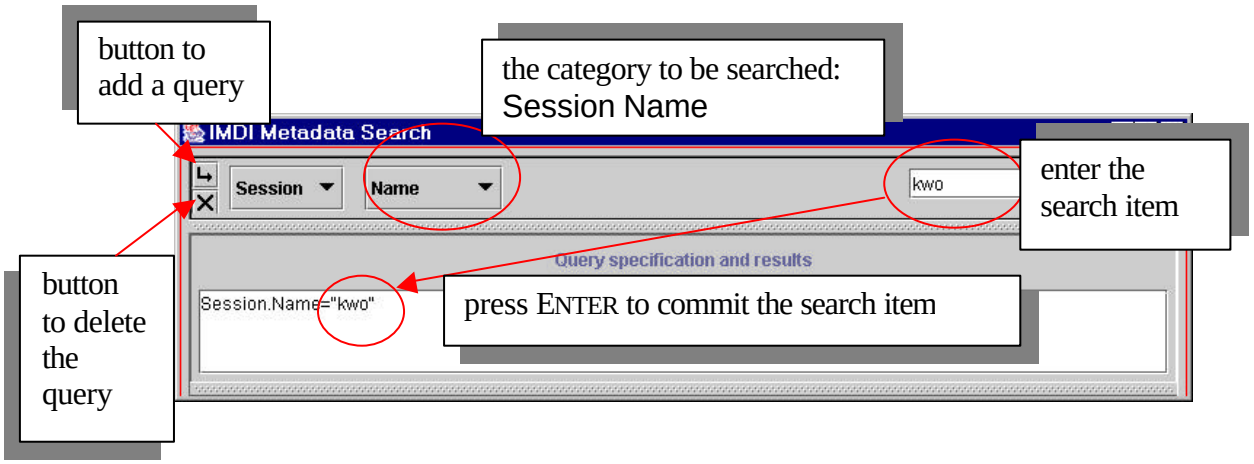
Once you have selected node(s), the search is automatically restricted to the selected node(s).

! Note: Again, you can only search subparts of corpora that have a corresponding file that contains all metadata information (see above).

! Note: Whenever you have selected parts of the corpus, metadata search will only search the listed nodes. If the search process does not yield the expected results, please make sure that the list contains all relevant nodes.

10.2 Specify the search parameters

The search options are specified in the topmost part of the IMDI Metadata Search window, e.g.:



The following search options are available:

1. Select the category to be searched (section 10.2.1).
2. Enter the search item (section 10.2.2).
3. Add or delete a search query (section 10.2.3).

10.2.1 Select the category to be searched

The categories to be searched are predefined by the IMDI Browser and are displayed in form of pull-down menus.

The default categories, which are displayed every time you access the IMDI Metadata Search window, are Session and Name. These two options allow you to search for sessions with a particular name (as in the illustration above).

You can select other categories by clicking on the pull-down menus:

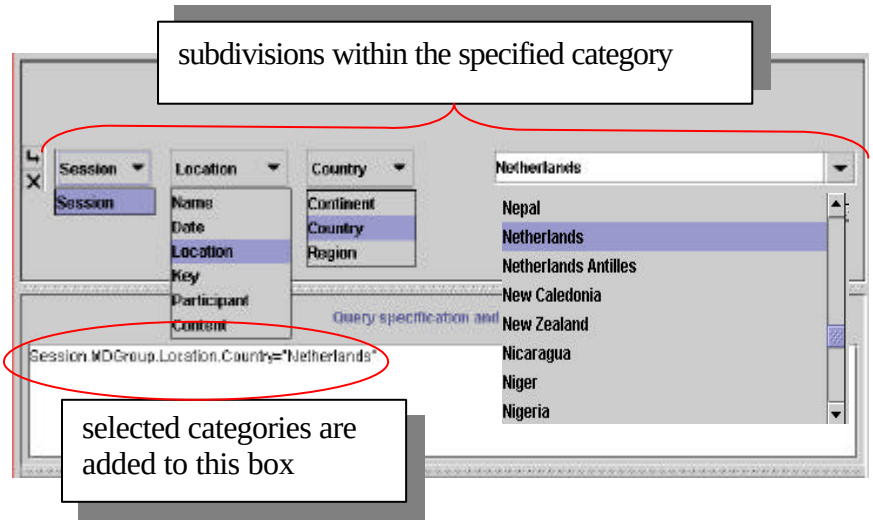
- The first category is always Session.
- The second category is chosen from the following list: Name, Date, Location, Key, Participant, or Content.
- Depending on your choice for the second category, further pull-down menus will appear to narrow down the category.

> **!** Note: If too many categories and pull-down menus are added, the IMDI Metadata Search window cannot display them all. To increase its size, click on the full screen icon (in the top right part of the window): 

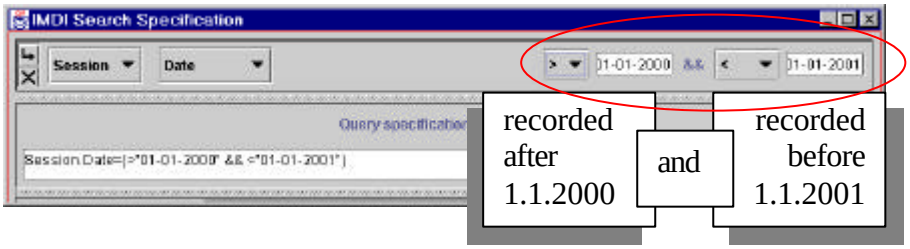
Some examples of metadata searches are given below.

(1) In the following illustration, Location is chosen as the second category. A third pull-down menu appears subdividing the category Location into Continent, Country, and Region. When Country is chosen, a fourth pull-down menu appears displaying the names of all countries.

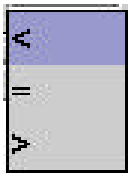
The selected category is automatically added into the box Query specification and results.



(2) In another example, Date is chosen as the second category (i.e., the date of the recording), e.g.:



In this case, further pull-down menus appear that allow you to specify the exact time period, e.g.:



search for a date before the specified date

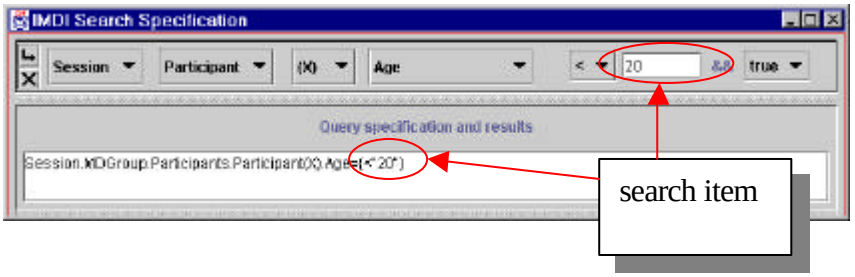
search for the exact date

search for a date after the specified date

For example, the specification "> *01-01-2000* && < *01-01-2001*" in the illustration above reads as "after 1.1.2000 and before 1.1.2001."

10.2.2 Enter the search item

After the search categories have been selected, the search item is entered into the box in the top right corner, e.g.:



Some search items have to be entered manually, others can be chosen from a pull-down menu.

! Note: If you enter the search item manually, you always need to press the key ENTER to commit the item.

! Note: You can type the search item directly in the box Query specification and results. Type it between the two quotation marks following the category specification, e.g.: *"20"*.

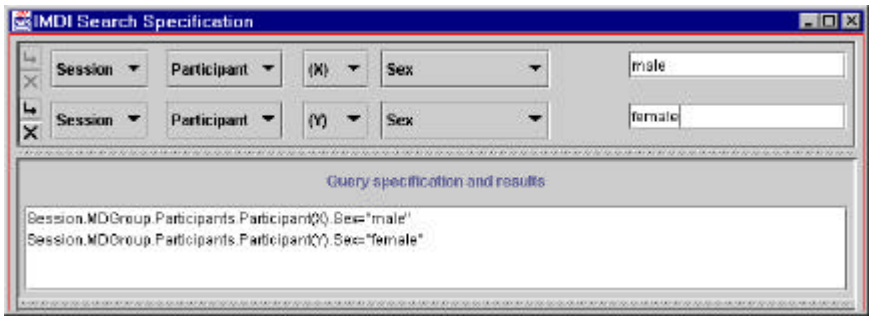
10.2.3 Add or delete a search query

You can add and delete queries as follows:

 use this button to add a query

 use this button to delete the query

When two (or more) queries are specified, the search process will find all sessions that contain query 1 *and* query 2 (*and* all subsequent queries). For example, in the following illustration, all sessions will be found that contain (1) a male participant X *and* (2) a female participant Y.



! Note: Variables such as X and Y are only of relevance if you specify more than one query. In this case, they allow you to specify that the participant of query 1 (X) should either be different from the participant of query 2 (Y) or that (s)he should be identical to the participant of query 2 (X). Please ignore these variables if you specify one query only.

10.3 Initiate and stop the search

Click the Search button at the bottom of the IMDI Metadata Search window to start the search. During the search the number of "hits" (matches) to your search criterion is shown, as well as the progress towards completion of the search.

! Note: Because of the large amount of data that is searched, it may take some time before the search is completed.

Once the search process has started, you can use the Stop button to stop the search.

10.4 Display the search results

Once the search process has started, the box Query specification and results in the IMDI Metadata Search window starts displaying the session nodes of each hit.

Double-click on a session node to open a second IMDI Browser window that contains the selected session, its metadata descriptions and its media and annotation files.

10.5 Save the search results

You can save the search results in form of a corpus node. The corpus node is displayed in the Meta Descriptions Tree panel of the IMDI Browser window.

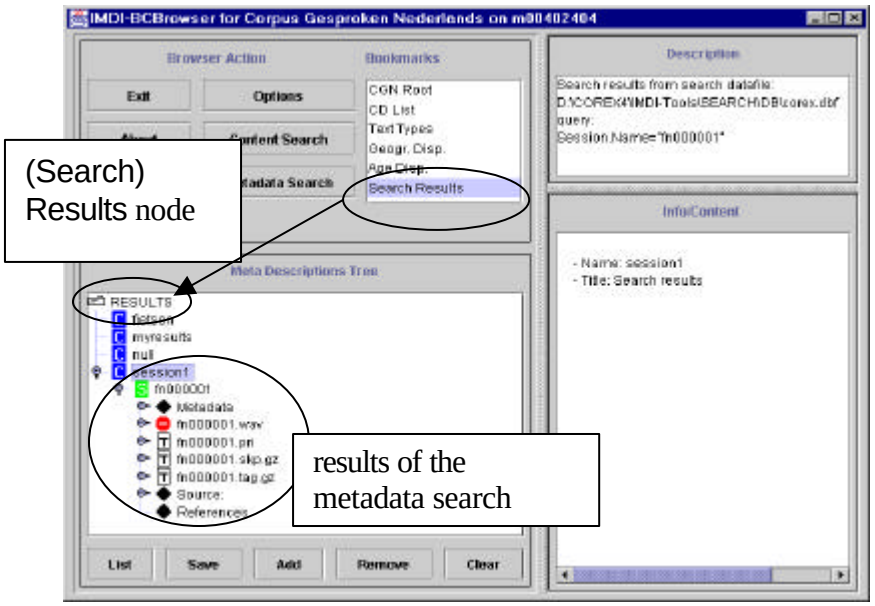
Do the following:

1. In the IMDI Metadata Search window, specify a name in the text box following the button Save results in:.
2. Click the button Save results in:. The results are saved under the specified name.

You can access the results through the IMDI Browser window. Do the following:

1. Go to the Bookmarks panel (see section 9.2).
2. Double-click on Search Results.

The Meta Descriptions Tree panel (see section 9.3) displays the new corpus under node under the node Results, e.g.:



The new corpus node is treated like any other corpus node: it contains sessions (those sessions where the search item was found), and the sessions contain metadata, media and annotation files.

You can remove the saved corpus node again. Do the following:

1. In the Meta Descriptions Tree panel, click on the node to select it. It will be highlighted in blue color.
2. Right-click on the selected node to open a pull-down menu.
3. Select Remove from the pull-down menu.

   The node will be removed.

> **!** Note: After selecting Remove, the icon of the removed node remains visible although it cannot be accessed anymore. The icon will be removed after exiting the node Results.

10.6 Exit the IMDI Metadata Search window

Click the Close button at the bottom of the IMDI Metadata Search window to exit the window.

Appendix 3 Miscelaneous documents MPI IMDITools

1. Local Tools for the IMDI-BCBrowser

Configuring Local Tool Use

The IMDI-BCBrowser has the possibility to start up locally installed tools when the right resources are available. The mapping between available local tools and required resources is specified in a configuration file "LocalTools.cfg". The basic format of the configuration file is XML. The IMDI-BCBrowser searches for this file in a directory that can be specified by the "Browser.cacheDir" parameter on the command line and default is the "IMDI-Tools" directory in the users home directory.

Local tools that are available can be started from the popup menu for the relevant nodes in the IMDI-BCBrowser. In the pop-up menu local tools have a "- "prefix.

Specifying the Node type

There are three classes of nodes where the local tool mechanism becomes relevant:

- **LanguageResourceNode(s): (TranscriptFileNode, SoundFileNode, VideoFileNode).** These nodes stand for different language resource types. The local tool that can be started works on the resource selected.
- **SessionNode(s)**: The Session node can bundle a variety of language resources. A tool here might want to show a transcription in combination with a sound file. The IMDI-BCBrowser knows (via the IMDI session metadata description file) what resources are available in a session and can start a tool working with (a subset) of these resources.
- **InfoFileNode(s)**: An InfoFileNode stands for an information file that is linked into the Browsable Corpus hierarchy. A local tool started from this node works on this information file. You may want to specify specific readers that can handle PDF and/or PS files.

Specifying a Platform

Local tools are limited to a specific platform: "Windows Media Player" only runs on the Windows platform and the XWaves signal processing software only runs on a number of UNIX platforms. So every local tool must have one or more platform specifications. The "WINDOWS" string indicates the Windows platform; a string "UNIX.xxx" indicates the UNIX platform where xxx can be a further specification of the platform, for instance SunOS for the Solaris platform.

Specifying a default action for a Node

It is possible to specify a local tool as the default action for a node. Whenever the user double-clicks that type of node and the necessary resources are available, that specific tool will be started.

Specifying the required resources

In the IMDI metadata description files, two metadata items identify the nature of the resources: “Type” and “format”. These two can be combined to a (semi-) mime-type “type/format”. This mime-type string is used in the configuration file to specify the required type of resource for a certain tool. It is possible to specify a “*” for the format, allowing all formats of a certain type. For example a local tool XXX works with all possible formats of audio, in this case the required mime-type is: “audio/*”.

Specifying the command for the local tools

In the configuration file the command for the local tool is specified by giving its path followed by any necessary flags and resource specifications. The resource specifications can be given in two flavors: (1) As a file specification in the form “$F1” or (2) as a URL specification in the form “$U1”. Of course the tool in question must support these formats. The numbers in the resource specification refer to the number in the required resource list. So for instance a tool XX that requires a transcription file with mime-type “text/chat” and a audio file with mime-type "audio/*” has for instance a command specification of:

XX –t $F1 –a $F2

The IMDI-BCBrowser substitutes the “$F1” and “$F2” tokens by the file specifications of the resources:

XX –t mychatfile.chat –a myaudiofile.wav

If a tool requires a file specification but in the IMDI metadata description there is only a remote “http” resource available, the tool is not made available in the popup menu.

In the command specification the variable $HOME may be used to specify the value of the “user.home” Java system property. This is useful when tools are stored in user dependent directories.

In the command specification the variable $CORPUSROOT may be used to specify the value of the Browser.corpusRoot parameter the user can specify on the command line.

The format of the LocalTools.cfg file

The outer XML element of the LocalTools.cfg file is **<LocalTools>.** This element has no attributes. The **<LocalTools>** element has one ore more **<Entry>** sub-elements; every **<Entry>** element specifies a tool for a specific node type. The **<Entry>** element has 3 attributes:

- *Name*= “xxx”: this is the name that appears in the pop-up menu.
- *Node*=”yyy”: yyy is the node type for which this tool may be shown. The following node types are supported:
  - SessionNode
  - InfoFileNode: for nodes representing human readable information, these files have usually mime-types of “text/txt”, “text/html”, “text/ps” etc.
  - SoundFileNode: A file containing an audio signal.
  - VideoFileNode: A file with video data.
- *Default*=”bbb”: bbb is true or false specifying if this tool is the default action for the node

Every **<Entry>** element can have one or more **<Resources>** elements. The **<Resources>** element specifies what types of resources are needed for the tool. There may be more than one resources specification because the same tool may work on different types of resources. The **<Resources>** element has as possible attributes:

- *MimeType1=*”xxx”: xxx is the mime-type of the first resource necessary for the tool
- *MimeType2=*”yyy”: yyy is the mime-type of the second resource necessary.
- *MimeType3=*”zzz”: zzz is the mime-type of the third resource necessary.

Every **<Resources>** element can have a number of **<Command>** sub elements, one for every platform that supports the local tool. The **<Command>** element has two attributes and has text content. The two attributes are:

- *Platform*=”xxx”: xxx is the platform specification. Possible is “WINDOWS” and “UNIX.vvvv” where vvv is a more narrow specification. For example Platform=”UNIX.SunOS” specifies the Solaris platform. The IMDI-BCBrowser is started with a command line parameter “Browser.OS=xxx”, the xxx is matched against the value of the “Platform” attribute to determine if a certain tool is relevant.
- *Type*=”yyy”: Indicates the type of command at the moment only the type “exec” is supported.

The content of the <Command> element is the command string with the resource file names replaced by $F1 … $Fn and resource URL’s replaces by $U1 … $Un

An example of a LocalTools.cfg file

This is an example of a local tools configuration file as used at the MPI:

```
<LocalTools>
<Entry Name="Med" Node="SessionBCnode" Default="true">
  <Resources MimeType1="text/chat" MimeType2="audio/*">
    <Command Platform="UNIX" Type="exec">med $U1 $U2</Command>
    <Command Platform="WINDOWS" Type="exec">med $U1 $U2</Command>
  </Resources>
  <Resources MimeType1="text/cha" MimeType2="audio/*">
    <Command Platform="UNIX" Type="exec">med $U1 $U2</Command>
    <Command Platform="WINDOWS" Type="exec">med $U1 $U2</Command>
  </Resources>
  <Resources MimeType1="text/chat" Default="true">
    <Command Platform="UNIX" Type="exec">med $U1</Command>
    <Command Platform="WINDOWS" Type="exec">med $U1</Command>
  </Resources>
  <Resources MimeType1="text/cha" Default="true">
    <Command Platform="UNIX" Type="exec">med $U1</Command>
    <Command Platform="WINDOWS" Type="exec">med $U1</Command>
  </Resources>
</Entry>

<Entry Name="Med" Node="TranscriptFileNode" Default="true">
  <Resources MimeType1="text/chat">
    <Command Platform="UNIX" Type="exec">med $U1</Command>
    <Command Platform="WINDOWS" Type="exec">med $U1</Command>
  </Resources>
</Entry>

<Entry Name="CLAN" Node="TranscriptFileNode" Default="true">
  <Resources MimeType1="text/chat">
    <Command Platform="WINDOWS" Type="exec">C:\CHILDES\CLAN\CLAN.EXE
$F1</Command>
  </Resources>
</Entry>
```

```
<Entry Name="Windows Media Player" Node="VideoFileNode" Default="true">
  <Resources MimeType1="video/*">
    <Command Platform="WINDOWS" Type="exec">"C:\Program Files\Windows Media
Player\mplayer2.exe" $F1</Command>
  </Resources>

</Entry>
<Entry Name="SMTV" Node="VideoFileNode" Default="true">
  <Resources MimeType1="video/mpg">
    <Command Platform="UNIX.SunOS" Type="exec">/opt/SUNWsmtv/bin/showmetv $F1
</Command>
  </Resources>
</Entry>

<Entry Name="XWaves" Node="SoundFileNode" >
  <Resources MimeType1="audio/wav">
    <Command Platform="UNIX" Type="exec">xwaves $F1 </Command>
  </Resources>
  <Resources MimeType1="audio/x-esps">
    <Command Platform="UNIX" Type="exec">xwaves $F1 </Command>
  </Resources>
  <Resources MimeType1="audio/esps">
    <Command Platform="UNIX" Type="exec">xwaves $F1 </Command>
  </Resources>
</Entry>

<Entry Name="Signal" Node="SoundFileNode" Default="true">
  <Resources MimeType1="audio/*">
    <Command Platform="UNIX" Type="exec">signal $U1 </Command>
    <Command Platform="WINDOWS" Type="exec">signal $U1 </Command>
  </Resources>
</Entry>

<Entry Name="Print" Node="InfoFileNode">
  <Resources MimeType1="text/*">
    <Command Platform="UNIX" Type="exec">lp $F1 </Command>
  </Resources>
</Entry>
</LocalTools>
```

2. The CorpusName.dbf file format.

The “service file” CorpusName.dbf file is (part of) an implementation of IMDI metadata search architecture. The file contains all the metadata of the sessions files in a (sub) corpus in a flattened format. This file is searched by a Perl script to answer IMDI metadata search queries. This is done for performance reasons; it avoids opening all XML files and profits from the optimised Perl regular expression search.

An example of this flattened format for the metadata of one session is given here:

```
$LOCAL_CORPUSROOT="/data/corpora/esf_conv/"
$REMOTE_CORPUSROOT="http://www.mpi.nl/world/data/esf_archive/"
Session_000000.URL="./xml/ladfc11a.1.imdi"
Session_000000.Name="ladfc11a.1"
Session_000000.Date="1982-11-09"
Session_000000.MDGroup.Location.Continent="europe"
Session_000000.MDGroup.Location.Country="netherlands"
Session_000000.MDGroup.Key__000={"sound_linking","??"}
Session_000000.MDGroup.Project.Contact.Name="esf"
Session_000000.MDGroup.Content.Languages.Language__000.Name="dutch"
Session_000000.MDGroup.Content.Languages.Language__001.Name="arabic"
Session_000000.MDGroup.Content.Key__000={"keywords","free-conversation, cultural-differences."}
Session_000000.MDGroup.Participants.Participant__000.Type="investigator"
Session_000000.MDGroup.Participants.Participant__000.Name="investigator 1"
Session_000000.MDGroup.Participants.Participant__000.FullName="josee"
Session_000000.MDGroup.Participants.Participant__000.Code="j= jc"
Session_000000.MDGroup.Participants.Participant__000.Anonymous="false"
Session_000000.MDGroup.Participants.Participant__001.Type="investigator"
Session_000000.MDGroup.Participants.Participant__001.Name="investigator 2"
Session_000000.MDGroup.Participants.Participant__001.FullName="rachid"
Session_000000.MDGroup.Participants.Participant__001.Code="z= rz"
Session_000000.MDGroup.Participants.Participant__001.Anonymous="false"
Session_000000.MDGroup.Participants.Participant__002.Type="investigator"
Session_000000.MDGroup.Participants.Participant__002.Name="investigator 3"
Session_000000.MDGroup.Participants.Participant__002.FullName="kemal, husband of fatima"
Session_000000.MDGroup.Participants.Participant__002.Code="w= op"
Session_000000.MDGroup.Participants.Participant__002.Anonymous="false"
Session_000000.MDGroup.Participants.Participant__003.Type="subject"
Session_000000.MDGroup.Participants.Participant__003.Name="subject"
Session_000000.MDGroup.Participants.Participant__003.FullName="fatima"
Session_000000.MDGroup.Participants.Participant__003.Code="f=fc"
Session_000000.MDGroup.Participants.Participant__003.Languages.Language__000.Name="arabic"
Session_000000.MDGroup.Participants.Participant__003.Languages.Language__001.Name="dutch"
Session_000000.MDGroup.Participants.Participant__003.Age="26"
Session_000000.MDGroup.Participants.Participant__003.Sex="female"
Session_000000.MDGroup.Participants.Participant__003.Anonymous="false"
Session_000000.MDGroup.Participants.Participant__003.Key__000={"religion","islam"}
Session_000000.MDGroup.Participants.Participant__003.Key__001={"birthplace","kenitra - marocco"}
Session_000000.MDGroup.Participants.Participant__003.Key__002={"year_of_birth","1956"}
Session_000000.MDGroup.Participants.Participant__003.Key__003={"civil_status","married"}
Session_000000.MDGroup.Participants.Participant__003.Key__004={"schooling_level","primary schol"}
Session_000000.MDGroup.Participants.Participant__003.Key__005={"language_spoken_at_home","moroccan
-arabic"}
Session_000000.Resources.MediaFile__000.ResourceLink="&datadir;/ladfc11a.1.wav"
Session_000000.Resources.MediaFile__000.Type="audio"
Session_000000.Resources.MediaFile__000.Format="esps"
Session_000000.Resources.AnnotationUnit__000.ResourceLink="&datadir;/ladfc11a.1.cha"
Session_000000.Resources.AnnotationUnit__000.Type="text"
Session_000000.Resources.AnnotationUnit__000.Format="chat"
Session_000000.Resources.AnnotationUnit__000.Anonymous="false"
Session_000000.Resources.AnnotationUnit__001.ResourceLink="&datadir;/ladfc11a.1tr"
Session_000000.Resources.AnnotationUnit__001.Type="text"
Session_000000.Resources.AnnotationUnit__001.Format="esf"
Session_000000.Resources.AnnotationUnit__001.Anonymous="false"
Session_000000.Resources.Source__000.Id="cassette: fc/m1, 02"
```

Session_000000.Resources.Source__000.Format="cc"
Session_000000.Resources.Source__001.Id="ladfc11"
Session_000000.Resources.Source__001.Format="dat"

To distinguish between sessions a unique number is added to the "Session" element. Just as to distinguish between repeatable elements such as "Participant a unique number is added to the "Participant" element.

Special formatting is required for the user configurable Key/Value pairs such as:
Session_000000.MDGroup.Participants.Participant__003.Key__000={"religion","islam"}
Both the element name and its value are user definable and should be treated as values and are therefore R-values.

Note that every session is identified with a URL: Session_000000.URL="./xml/ladfc11a.1.imdi"
that is returned if the session conforms to the query constraints.

The path URL's can be relative. To allow the an application to compute the absolute URL of the corpus and session metadata description files the CorpusName.dbf file can contain lines with format:

$LOCAL_CORPUSROOT="file:/rootdir
$REMOTE_CORPUSROOT="http://ourhost/remrootdir"

This allows an application to use one of these definitions dependent on the usage scenario (local or remote access) to make the relative path names absolute. The definitions are valid for the Session metadata info that follows them until they are superseded by a new definition.

The CreateCorpusStructures.pl script creates the .dbf files.

3. The CorpusName.hry file format.

The "service file" CorpusName.hry file is (part of) an implementation of the "CorpusStructureService" used by the IMDI-BCBrowser to obtain information on what "Sessions" are contained in a (sub-)corpus.
The file contains lines that either specifies that a metadata description file as a corpus together with its child subcorpus metadata description files. Or the line specifies that a metadata description file is a session. It is also possible to refer authority for a sub-corpus to another .hry file by putting a special statement in the top .hry file. An example of this .hry format is:

C CFURL1 CFURL11 CFURL12 CFURL13
C CFURL11 CFURL111 CFURL112 CFURL113
C CFURL12 CFURL121 CFURL122 CFURL123
CS CFURL13 cfurl13.hry
S CURL111
S CURL112
S CURL113
S CURL121

This represents a corpus with topnode CFURL1 with three subcorpus nodes further culminating in a number of Sessions: CURL111 ... CURL113. The structure for subcorpus "CFURL13" is delegated to the "cfurl.hry" file.

The path URL's can be relative. To allow the an application to compute the absolte URL of the corpus and session metadata description files the CorpusName.hry file can contain lines with format:

$LOCAL_CORPUSROOT="file:/rootdir
$REMOTE_CORPUSROOT="http://ourhost/remrootdir"

This allows an application to use one of these definitions dependend on the usage scenario (local or remote access) to make the relative path names absolute. The definitions are authorative for the corpus structure definition lines that follow them until they are superseeded by a new definition.

The .hry file can also specify the associated SearchService or .dbf file. The statement for this has format:
$SEARCHSERVICE=".dbf file path"
Every .hry file should only have one SEARCHSERVICE directive only.

The .hry files are created by the CreateCorpusStructures.pl script.

4. The IMDI-BCBrowser Tool

Installation and configuration guide

Updated july 2002

Introduction.

The IMDI-BCBrowser is a tool that is used to browse linked IMDI metadata descriptions that describe linguistic corpora sub-corpora and resources. The format of these metadata descriptions and the semantics of the metadata vocabulary used can be found in [1].
The Browser has also additional functions build in such as a complete version of the IMDI-BCEditor [2] that can be used to modify or generate new IMDI metadata descriptions. Also some viewer/editors for language resources of a specific format are included in the configuration file. The users can start-up local available tools on language resources directly from the Browser. How to set this up is described in [3].

IMDI metadata infrastructure

The IMDI-BCBrowser follows the links between the IMDI metadata description files and displays them as a tree structure. To support metadata search and sub-corpus selection there can be two kinds of additional information files or services present. The first is the "CorpusStructureService" or CorpusName.hry file. This provides information about the structure of the corpus hierarchy: which Sessions are contained in which sub-corpora. The Browser could also find this information by following the links between the corpus nodes but that is too slow. The default CorpusStructureService can be specified as a parameter during startup. This default can be overruled when a corpus metadata discription file specifically points to another CorpusStructure Service. That new setting is then valid (authorative) for all its descendants unless one of them defines a new one. If there is no CorpusStructure service available for a subcorpus, the Browser can not make selections out that subcorpus. The second information service is the SearchService or CorpusName.dbf file.This service provides metadata about (part of) a corpus. Again the Browser could obtain this information by following the links to sub-corpora and Sessions but this process is too slow. The default SearchService can be specified as a parameter during startup. This default can be overruled when a corpus metadata discription file specifically points to another SearchService. That new setting is then valid (authorative) for all the corpus node descendants unless one of them defines a new one. If there is no Searchservice available for a subcorpus, the Browser can not search metadata descriptions in that corpus.

Installation.

The IMDI-BCBrowser is a Java programme and needs an installed version of at least Java1.3. If the tool is distributed on a CD/DVD, the installation procedure also installs a Java version. This is only the case for the Windows platform, UNIX users are expected to provide their own Java installation.
The Browser (and connected tools) is contained in a "jar" file imdi.jar. Required other jars are xerces.jar and jmf.jar (on the CD/DVD these are also provided). The start-up command for the Browser is:

path_to_java\java –Dparam_1="value_1" –classpath xerces.jar;jmf.jar;imdi.jar mpi.bc.Browser "start_node"

There are a large number of parameters that can be provided on the command-line, these will be explained in the configuration section. Usually the whole command is inserted in a ".bat" file or desktop icon.
The IMDI tools such as IMDI-BCBrowser and IMDI-BCEditor rely partly on information stored in configuration files. Normally these files are stored in a "configuration" directory named "IMDI-Tools"; this is a sub directory of the directory specified by the "user.home" java property. For UNIX platforms this is for instance "/home/username/IMDI-Tools" and on Windows 2000/NT it is equivalent to the variable %USERPROFILE%. This value can be influenced by specifying parameters with the start-up command (see the configuration section). The directory also functions as a cache where configuration files are stored after they are first copied from remote sites.
The first time the Browser is started it will create the configuration directory and try to fill it with files downloaded from the IMDI site http://www.mpi.nl/IMDI/Config/mpi. If there is no network connectivity or the remote site is down the Browser will use internal default files and copy these to the configuration directory.

If you want to force the Browser to download new configuration files, delete those present in "cacheDir"

Configuration

The following (user) configuration parameters can be provided on the command line:

- **OS:** Specifies the platform for WINDOWS use "WINDOWS" for unix use "UNIX.xxx" where xxx is a further specification of the unix brand. For instance on SUN OS.="UNIX.SunOS". It is the output of the UNIX command "uname –s". Normally the default is correct.
- **cacheDir**: Specifies the directory where the IMDI programmes store configuration files. This must be a writable directory. **Default** is the sub directory named IMDI-Tools in the directory specified by the java property "user.home". So for UNIX this should be "/home/username/IMDI-Tools".
- **configURL**: Points to a local or remote directory that contains configuration files. Default is "http://www.mpi.nl/IMDI/Config/mpi". If you have site-specific configuration use this parameter to have all Browsers at your site download that specific configuration file. If this parameter is not present or an emty string the Browser will not try to download any configuration files but uses any defaults present in the Jar file.
- **ltConfigFileName**: The name of the configuration file that specifies the available local tools. Default is "LocalTools.cfg" in the directory pointed to by the "cacheDir" parameter. So in the default (UNIX) case the Browser first looks for a file "/home/username/IMDI-Tools/LocalTools.cfg". If it cannot find this it tries to download it from the URL pointed to by the "configURL" parameter. In the default case this would be: http://www.mpi.nl/IMDI/Config/mpi/LocalTools.cfg and if that resource also is not available the Browser uses an internal default file. This internal default file is copied to the "cacheDir" directory. If you want to replace it by one defined at URL configURL simply remove the version in the cacheDir directory.
- **mdSearchMode**: Using "mdSearchMode=true" enables the MD Search button in the browser. This mode is also automatically enabled when using some other modes such as "mpiMode=true".
- **corpusStructureURL:** The URL of the default CorpusName.hry file that is used to compute the sessions belonging to a (sub-) corpus. This URL can also be named by a Corpus metadata description file in which case that URL overrules the default settings by defCorpusStructureFile for that Corpus Node and its descendants.
- **mdDefSearchFile**: The specification of the metadata search file. or SearchService. This file or remote service is used when specifying a metadata search. The user may change this file runtime by using the "Browse" button in the MD Search panel. Default is empty string. If the file specification is a remote URL (http protocol) the browser uses the remote HTTP server to start IMDI metadata search remotely

if this has been configured at the remote site. The SearchService can also be named by a Corpus metadata description file. This setting then overules the default setting by "mdDefSearchFile" for that CorpusNode and its descendants.

- **globalBookmarkFileName**: The name of the global bookmark file. This file is parsed at the start and any bookmark definitions found there are put in the bookmark list before definitions found in the private bookmark file. The Browser first looks at the (remote) url "configURL/globalBookmarkFileName" if this file can not be downloaded it tries if a file with the name "globalBookmarkFileName" is available in the Jar file. Default is the name "globalBookmarks.xml".
- **defaultCharSet**: The character set that is assumed by the Browser for files when it cannot determine the character set. For instance when accessing a file not through a HTTP server. Default is "ISO-8859-1"
- **mpiMode**: Using mpiMode=true makes some default resources available via the MPI (Max Planck Institute for Psycholinguistics) intranet. If using. Default is false.
- **imdiMode**: Using imdiMode=true makes metadata hierarchies available that were created for the IMDI show case. Default is false.
- **dobesMode**: Using dobesMode=true makes DoBes project metadata hierarchies available. Default is false.
- **corexMode**: Using corexMode=true sets the Browser in the mode as an exploitation tool of the "Spoken Dutch Corpus". Do not use this if you do not have the Spoken Dutch Corpus since it assumes special available data. Default is false.
- **CorpusRoot**: This sets the default value oi the rootURL of all corpora. It is used whenever a relative resource path needs to be resolved. The default can be overruled by "parameter" entries in the "CorpusName.hry" file. Those parameter definitions "$LOCAL_CORPUSROOT=my dir" wil change the value of LorbusRoot for the with the CorpusSiructure associated Corpus Node and its descendants.
- **cdRoot**: corex specific.Sets the name of the directory where the COREX viewer searches for audio files.
- **useCDnames**: corex specific. If true adds the name of the release directory to cdRoot a when searching for audio files.

Examples

To start the IMDI-BCBrowser on the "sample"childes corpus hierarchy do:

Java_home\bin\java
–DcacheDir="D:\mydata\IMDI-Tools"
-DdefSearchFile="D:\mydata\ChildesDemo\childes.dbf"
-DcorpusRoot="D:\mydata\ChildesDemo"
–classpath xerces.jar;jmf.jar;imdi.jar mpi.bc.Browser
D:\mydata\Corpora\CHILDES\imdi\childes.imdi

To start the IMDI-BCBrowser as an exploitation tool for the Spoken Dutch Corpus do:

Java_home\bin\java
-DcorpusRoot="C:\COREX3a\Corpora"
–DcdRoot="E:"
–DcacheDir="C:\COREX3a\IMDI-Tools"
-classpath xerces.jar;jmf.jar;imdi.jar
C:\COREX3a\Corpora\metadata\xml\cgn.xml

To start the IMDI-BCBrowser to use the "sample" IMDI showcase data available from the MPI Website, do:

Java_home\bin\java
-DimdiMode=true
-classpath xerces.jar;jmf.jar;imdi.jar
http://www.mpi.nl/IMDI/data/showcase.imdi

To start the IMDI-BCBrowser with site-specific configuration files use:

Java_home\bin\java
–DcacheDir=”D:\mydata\IMDI-Tools”
–DconfigURL=”http://www.mysite.edu/IMDI/config”
–classpath xerces.jar;jmf.jar;imdi.jar mpi.bc.Browser
D:\Corpora\mycorpus\start.imdi

If you want to force the Browser to download new configuration files, delete those present in “cacheDir”

References:

[1] IMDI metadata vocabulary, ISLE D10-2
[2] IMDI metadata editor manual, Appendix 1
[3] Local Tools for the IMDI-BCBrowser “LocalTools.doc” Appendix 3, chapter 1

Appendix 4 : Database search procedures and examples

1 Text of primary conversion scripts

1.1 Main conversion script : *mpi2db*

```
#!/usr/bin/sh

# Convert the MPI / IMDI text files to tab separated
# entries for a database

# Argument : name of the file

/usr/local/bin/sed -n -f ./SessionId.sed $1 > SessionId.table
/usr/local/bin/sed -n -f ./SessionKeys.sed $1 > SessionKeys.table
/usr/local/bin/sed -n -f ./ContentKeys.sed $1 > ContentKeys.table
/usr/local/bin/sed -n -f ./ParticipantId.sed $1 > ParticipantId.table
/usr/local/bin/sed -n -f ./ParticipantKeys.sed $1 > ParticipantKeys.table
/usr/local/bin/sed -n -f ./ParticipantLanguages.sed $1 > ParticipantLanguages.table
/usr/local/bin/sed -n -f ./Mediafile.sed $1 > Mediafile.table
/usr/local/bin/sed -n -f ./AnnotationUnit.sed $1 > AnnotationUnit.table
/usr/local/bin/sed -n -f ./Source.sed $1 > Source.table
/usr/local/bin/sed -n -f ./ProjectContentLanguages.sed $1 > ProjectContentLanguages.table

./oneline ./SessionId.table
./oneline ./SessionKeys.table
./oneline ./ContentKeys.table
./oneline ./ParticipantId.table
./oneline ./ParticipantKeys.table
./oneline ./ParticipantLanguages.table
./oneline ./Mediafile.table
./oneline ./AnnotationUnit.table
./oneline ./Source.table
./oneline ./ProjectContentLanguages.table
```

1.2 Sample ‘sed’ script: *SessionId.sed*

```
s/Session_\(.....\)\.URL=/----\
\1\
url\
/p
s/Session_\(.....\)\.Name=/----\
\1\
name\
/p
s/Session_\(.....\)\.Date=/----\
\1\
date\
/p
s/Session_\(.....\)\.MDGroup\.Location\.Continent=/----\
\1\
location_continent\
/p
```

```
s/Session_\(......\)\.MDGroup\.Location\.Country=/----\
\1\
location_country\
/p
s/Session_\(......\)\.MDGroup\.Project\.Contact\.Name=/----\
\1\
project_contact_name\
/p
```

1.3 Another sample ‘sed’ script : *ParticipantKeys.sed*

```
s/Session_\(......\).*Participant__\(...\)\.Key__\(...\)={\(.*\),\(.*\)}/----\
\1\
\2\
\3\
\4\
\5/p
```

1.4 The *oneline* conversion script

```
#!/usr/bin/sh

# sets on one line, separated by vertical bars, the entries between the --- marks

tr '\n' '|' < $1 | sed 's/----/\
/g' | sed 's/^|\(.*\)|$/\1/' | tail +2 | tr -d "" > $1.temp

echo " >> $1.temp
mv $1.temp $1
```

2 Text of Postgres commands for database creation

2.1 The main file: *main.pg*

```
\i aliment-database.pg

\i load-participant-id.pg
drop table participant_id_temp;

\i load-session-id.pg
drop table session_id_temp;

\i load-mediafile.pg
drop table mediafile_temp;

\i load-source.pg
drop table source_temp;

\i load-annotation-unit.pg
drop table annotation_unit_temp;
```

2.2 Excerpt from the intermediary table creation file: *aliment-database.pg*

```
drop table session_id_temp ;

create table session_id_temp(
```

```
   session_nb smallint,
   infotype text,
   infovalue text);

copy session_id_temp
from '/home/andrei/ISLE/IMDI/database_tests/SessionId.table'
using delimiters '|' ;

create index index_session_id_temp on session_id_temp(session_nb);

-----------------------------------------

drop table session_keys ;

create table session_keys(
   session_nb smallint,
   key_nb smallint,
   key_name text,
   key_value text);

copy session_keys
from '/home/andrei/ISLE/IMDI/database_tests/SessionKeys.table'
using delimiters '|' ;

create unique index index_session_keys on session_keys(session_nb,key_nb);

-----------------------------------------

drop table participant_id_temp;

create table participant_id_temp (
   session_nb smallint,
   participant_nb smallint,
   infotype text,
   infovalue text);

copy participant_id_temp
from '/home/andrei/ISLE/IMDI/database_tests/ParticipantId.table'
using delimiters '|' ;

create index index_participant_id_temp
on participant_id_temp(session_nb,participant_nb);

-------------------------------------------------

drop table participant_keys ;

create table participant_keys(
   session_nb smallint,
   participant_nb smallint,
   participant_key_nb smallint,
   participant_key_name text,
   participant_key_value text);

copy participant_keys
```

```
from '/home/andrei/ISLE/IMDI/database_tests/ParticipantKeys.table'
using delimiters '|' ;

create unique index index_participant_keys
on participant_keys(session_nb,participant_nb,participant_key_nb);

[...]
```

2.3 Sample final table creation file: *load-participant-id.pg*

```
drop table participant_id;

create table participant_id (
   session_nb smallint,
   participant_nb smallint,
   type text,
   name text,
   fullname text,
   code text,
   role text,
   age text,
   sex text,
   anonymous text);
---------------------------------
insert into participant_id (session_nb, participant_nb)
select distinct session_nb, participant_nb
from participant_id_temp;

update participant_id
set type = (select infovalue from participant_id_temp
          where participant_id_temp.session_nb=participant_id.session_nb
            and participant_id_temp.participant_nb=participant_id.participant_nb
            and participant_id_temp.infotype='type');

update participant_id
set anonymous = (select infovalue from participant_id_temp
          where participant_id_temp.session_nb=participant_id.session_nb
            and participant_id_temp.participant_nb=participant_id.participant_nb
            and participant_id_temp.infotype='anonymous');

update participant_id
set name = (select infovalue from participant_id_temp
          where participant_id_temp.session_nb=participant_id.session_nb
            and participant_id_temp.participant_nb=participant_id.participant_nb
            and participant_id_temp.infotype='name');

update participant_id
set fullname = (select infovalue from participant_id_temp
          where participant_id_temp.session_nb=participant_id.session_nb
            and participant_id_temp.participant_nb=participant_id.participant_nb
            and participant_id_temp.infotype='fullname');

update participant_id
set code = (select infovalue from participant_id_temp
          where participant_id_temp.session_nb=participant_id.session_nb
            and participant_id_temp.participant_nb=participant_id.participant_nb
```

```
            and participant_id_temp.infotype='code');

update participant_id
set role = (select infovalue from participant_id_temp
           where participant_id_temp.session_nb=participant_id.session_nb
             and participant_id_temp.participant_nb=participant_id.participant_nb
             and participant_id_temp.infotype='role');

update participant_id
set age = (select infovalue from participant_id_temp
          where participant_id_temp.session_nb=participant_id.session_nb
            and participant_id_temp.participant_nb=participant_id.participant_nb
            and participant_id_temp.infotype='age');

update participant_id
set sex = (select infovalue from participant_id_temp
          where participant_id_temp.session_nb=participant_id.session_nb
            and participant_id_temp.participant_nb=participant_id.participant_nb
            and participant_id_temp.infotype='sex');
---------------------------------------

create unique index index_participant_id
on participant_id(session_nb,participant_nb);
```

3 Text of queries and excerpt of results

3.1 Full text of the test query file, including comments

```
\o mpi-queries-output.txt

-- ================================
-- Example Queries for IMDI Search

-- Andrei Popescu-Belis, Feb. 2002
-- ================================

-- ---------------------------------------------------------------

-- Give me the session with name "lsfpa12a.1"
-- **** session1.q ****
-- Session.Name="lsfpa12a.1"

-- COMMENT: : no such name in L2.dbf

select session_nb, name
from session_id
where name='d-dk1m1-hun' ;

-- ---------------------------------------------------------------

-- Give me all sessions from June 1983
-- **** session2.q ****
-- Session.Date="1983-06.+"
```

```
-- COMMENT:  : and order them by date...

select session_nb, name, date
from session_id
where (date >= '06-01-1986'
  and  date <= '06-30-1986')
order by date;

-- ---------------------------------------------------------------------

-- Give me all the sessions collected by "colin lector"
-- **** col1.q ****
-- Session.MDGroup.Collector.Name="colin lector"

-- COMMENT:  : There is no "Collector" in L2.dbf
-- COMMENT:  Variant: All session numbers where the investigator is hayo

select session_nb, type, name
from participant_id
where type = 'investigator'
  and name = 'hayo' ;

-- More interesting: Give me all investigator names with
-- the number of sessions in which each was involved

select name, count(session_nb) as number_of_sessions
from participant_id
where type = 'investigator'
group by name ;

-- ---------------------------------------------------------------------

-- Give me all participants born in 1965 (using key "YEAR_OF_BIRTH")
-- **** key1.q ****
-- Session.MDGroup.Participants.Participant(a).Key={"YEAR_OF_BIRTH","1965"}

-- COMMENT: : No such key in L2.dbf
-- COMMENT: : Using instead {"levelofproficiency" , "*good*"}

select session_nb, participant_nb, participant_key_value
from participant_keys
where (participant_key_name = 'levelofproficiency'
  and  participant_key_value ~~ '%good%') ;

-- try also participant_key_value ~* '.*good.*'

-- ---------------------------------------------------------------------

-- Give me all married participants born in 1965 (using keys YEAR_OF_BIRTH, CIVIL_STATUS)
-- **** key2.q ****
-- Session.MDGroup.Participants.Participant(a).Key={"YEAR_OF_BIRTH","1965"}
-- Session.MDGroup.Participants.Participant(a).Key={"CIVIL_STATUS","married"}

-- No such keys in L2.dbf
```

```
-- Give me all the participant_key names in L2.dbf

select distinct participant_key_name from participant_keys ;

-- ---------------------------------------------------------------------

-- Give me all sessions located in Europe
-- **** loc1.q ****
-- Session.MDGroup.Location.Continent="Europe"

-- Give me all sessions located in France (without continent specified)
-- **** loc2.q ****
-- Session.MDGroup.Location.Country="France"

-- Give me all sessions located in France (with contintent specified)
-- **** loc3.q ****
-- Session.MDGroup.Location.Continent="Europe"
-- Session.MDGroup.Location.Country="France"

-- COMMENT: : Unfortunately, there are no locations in L2.dbf
-- The only ones are in Content.Keys : {"location","prague"}

-- ---------------------------------------------------------------------

-- Give me all participants with full name "Clive"
-- **** part1.q ****
-- Session.MDGroup.Participants.Participant(a).FullName="Clive"

-- COMMENT: : There are no 'fullname' in L2.dbf, but there are 'name'-s.
-- Give me all participant names:

select distinct name from participant_id ;

-- Give all information about participant named sonja:

select *
from participant_id
where name = 'sonja' ;

-- ---------------------------------------------------------------------

-- Give me all male participants
-- **** part2.q ****
-- Session.MDGroup.Participants.Participant(a).Sex="Male"

select *
from participant_id
where sex = 'male' ;

-- ---------------------------------------------------------------------

-- Give me all female participants
-- **** part3.q ****
-- Session.MDGroup.Participants.Participant(a).Sex="Female"
```

```
select *
from participant_id
where sex = 'female' ;

-- ---------------------------------------------------------------------

-- Give me all combinations of one male and one female participant in a session
-- **** part4.q ****
-- Session.MDGroup.Participants.Participant(a).Sex="Male"
-- Session.MDGroup.Participants.Participant(b).Sex="Female"

select p1.session_nb, p1.sex, p1.participant_nb, p2.sex, p2.participant_nb
from participant_id p1, participant_id p2
where p1.session_nb = p2.session_nb
  and p1.sex = 'male'
  and p2.sex = 'female' ;

-- ---------------------------------------------------------------------

-- Give me all combinations of one female and another female in sessions where
-- the content language is "French"
-- **** part5.q ****
-- Session.MDGroup.Participants.Participant(a).Sex="Female"
-- Session.MDGroup.Participants.Participant(b).Sex="Female"
-- Session.MDGroup.Content.Languages.Language(o).Name="French"

-- COMMENT: : There is no French in L2.dbf
-- Show me all Content.Languages with their count :

select language_name, count(session_nb)
from project_content_languages
group by language_name ;

-- COMMENT: : Show me the languages of the sessions in which there are two females

select p1.session_nb, p1.participant_nb, p2.participant_nb, pcl.language_name
from participant_id p1, participant_id p2, project_content_languages pcl
where p1.session_nb = p2.session_nb
  and p1.sex = 'female'
  and p2.sex = 'female'
  and p1.participant_nb < p2.participant_nb
  and pcl.session_nb = p2.session_nb ;

-- COMMENT: : Show me the number of sessions per language with two female participants

select  pcl.language_name, count(p1.session_nb)
from participant_id p1, participant_id p2, project_content_languages pcl
where p1.session_nb = p2.session_nb
  and p1.sex = 'female'
  and p2.sex = 'female'
  and p1.participant_nb < p2.participant_nb
  and pcl.session_nb = p2.session_nb
group by pcl.language_name ;
```

```
-- Give me all combinations of one female and another female in sessions where
-- the content language is "czech sign language"

select p1.session_nb, pcl.language_name, p1.name, p1.sex, p2.name, p2.sex
from participant_id p1, participant_id p2, project_content_languages pcl
where p1.session_nb = p2.session_nb
  and p1.sex = 'female'
  and p2.sex = 'female'
  and p1.participant_nb < p2.participant_nb
  and pcl.session_nb = p2.session_nb
  and pcl.language_name = 'czech sign language' ;

-- ---------------------------------------------------------------------

-- Give me all combinations of three female participants where the content
-- language is "French"
-- **** part6.q ****
-- Session.MDGroup.Participants.Participant(a).Sex="Female"
-- Session.MDGroup.Participants.Participant(b).Sex="Female"
-- Session.MDGroup.Participants.Participant(c).Sex="Female"
-- Session.MDGroup.Content.Languages.Language(o).Name="French"

-- COMMENT:  : There is no language "French" in L2.dbf
-- and there are no three females in a session (but f+f+m)

select p1.session_nb, pcl.language_name, p1.name, p2.name, p3.name
from participant_id p1, participant_id p2, participant_id p3, project_content_languages pcl
where p1.session_nb = p2.session_nb
  and p2.session_nb = p3.session_nb
  and p1.sex = 'female'
  and p2.sex = 'female'
  and p3.sex = 'female'
  and p1.participant_nb < p2.participant_nb
  and p2.participant_nb < p3.participant_nb
  and pcl.language_name = 'czech'
  and pcl.session_nb = p1.session_nb ;

-- no solution !

-- ---------------------------------------------------------------------

-- Give me all female participants with the age of 30 to 39
-- **** part7.q ****
-- Session.MDGroup.Participants.Participant(a).Age="3[0-9]"
-- Session.MDGroup.Participants.Participant(a).Sex="Female"

select distinct name, age
from participant_id
where sex = 'female'
  and age >= 30
  and age <= 39 ;

-- ---------------------------------------------------------------------
\o
```

3.2 Excerpts of the results file

```
==============================
Example Queries for IMDI Search

==============================

select session_nb, name
from session_id
where name='d-dk1m1-hun' ;

 session_nb |    name
------------+-------------
         13 | d-dk1m1-hun
(1 row)


---------------------------------------------------------------------

Give me all sessions from June 1983
**** session2.q ****
Session.Date="1983-06.+"

COMMENT:   and order them by date...

select session_nb, name, date
from session_id
where (date >= 06-01-1986
and date <= 06-30-1986 )
order by date;

 session_nb |    name     |    date
------------+-------------+------------
        289 | p-db1k2-001 | 1986-06-02
        291 | p-db1m2-eid | 1986-06-02
        290 | p-db1k2-002 | 1986-06-02
        288 | p-db1i2-ein | 1986-06-02
        287 | p-db1e2-mig | 1986-06-02
        170 | p-dj7i1-pkt | 1986-06-13
        171 | p-dj7k1-001 | 1986-06-13
        172 | p-dj7k1-002 | 1986-06-13
        501 | p-ds7e1-gsi | 1986-06-13
        595 | p-dm8i1-lmp | 1986-06-13
        502 | p-ds7i1-pkt | 1986-06-13
        503 | p-ds7k1-001 | 1986-06-13
        504 | p-ds7k1-002 | 1986-06-13
        505 | p-ds7k1-003 | 1986-06-13
        506 | p-ds7m1-otr | 1986-06-13
        507 | p-ds7p1-fab | 1986-06-13
         78 | i-df7p1-hab | 1986-06-18
         77 | i-df7m1-dar | 1986-06-18
         76 | i-df7k1-001 | 1986-06-18
         75 | i-df7i1-kfm | 1986-06-18
         74 | i-df7e1-ccp | 1986-06-18
        596 | p-dm8k1-001 | 1986-06-27
```

```
      597 | p-dm8k1-002 | 1986-06-27
      598 | p-dm8k1-003 | 1986-06-27
      599 | p-dm8m1-vge | 1986-06-27
      600 | p-dm8p1-vsr | 1986-06-27
(26 rows)
```

Give me all the sessions collected by "colin lector"
COMMENT: There is no "Collector" in L2.dbf
COMMENT: Variant: All session numbers where the investigator is hayo

```
select session_nb, type, name
from participant_id
where type = investigator
and name = hayo ;
```

```
 session_nb |     type     | name
------------+--------------+------
      385 | investigator | hayo
      386 | investigator | hayo
      387 | investigator | hayo
      388 | investigator | hayo
      557 | investigator | hayo
      558 | investigator | hayo
(6 rows)
```

More interesting: Give me all investigator names with
the number of sessions in which each was involved

```
select name, count(session_nb) as number_of_sessions
from participant_id
where type = investigator
group by name ;
```

```
      name        | number_of_sessions
------------------+--------------------
                  |                 13
 aleandar         |                  1
 aleksandar       |                121
 aleksandar()     |                  1
 astrid           |                151
 b                |                  1
 b.               |                  3
 barbara          |                875
 bernt            |                  2
 christine dimroth |                 40
 e                |                  2
 g                |                  1
 giesela          |                  2
 gisela           |                  2
 h                |                  2
 hayo             |                  6
```

```
 heike          |              2
 heiner         |            229
 karin          |             12
 lena           |            218
 lenathomas     |              1
 norbert        |             61
 norbert,       |              1
 roman          |             25
 tomasz         |              1
(25 rows)
```

Give me all participants born in 1965 (using key "YEAR_OF_BIRTH")
**** key1.q ****
Session.MDGroup.Participants.Participant(a).Key={"YEAR_OF_BIRTH","1965"}

COMMENT: : No such key in L2.dbf
COMMENT: : Using instead {"levelofproficiency" , "*good*" }

```
select session_nb, participant_nb, participant_key_value
from participant_keys
where (participant_key_name = levelofproficiency
and participant_key_value ~~ %good% ) ;
```

```
 session_nb | participant_nb |  participant_key_value
------------+----------------+-------------------------
       852 |              1 | nativespeaker/verygood
       853 |              1 | nativespeaker/verygood
       854 |              1 | nativespeaker/verygood
       855 |              1 | nativespeaker/verygood
(…etc…)
      1634 |              0 | nativespeaker/verygood
      1635 |              0 | nativespeaker/verygood
      1636 |              0 | nativespeaker/verygood
      1637 |              0 | nativespeaker/verygood
      1638 |              0 | nativespeaker/verygood
      1639 |              0 | nativespeaker/verygood
(309 rows)
```

try also participant_key_value ~* .*good.*

Give me all married participants born in 1965 (using keys YEAR_OF_BIRTH, CIVIL_STATUS)
**** key2.q ****
Session.MDGroup.Participants.Participant(a).Key={"YEAR_OF_BIRTH","1965"}
Session.MDGroup.Participants.Participant(a).Key={"CIVIL_STATUS","married"}

No such keys in L2.dbf
Give me all the participant_key names in L2.dbf

```
select distinct participant_key_name from participant_keys ;
```

```
 participant_key_name
----------------------
 aufenthaltsdauer
 duration_of_stay
 levelofproficiency
(3 rows)
```

Give me all participants with full name "Clive"
**** part1.q ****
Session.MDGroup.Participants.Participant(a).FullName="Clive"

COMMENT: : There are no fullname in L2.dbf, but there are name -s.
Give me all participant names:

select distinct name from participant_id ;

```
                    aleandar            b.                  zmal
     name           aleksandar          barbara             (149 rows)
--------------      aleksandar()        [...]
                    antek
 adel               astrid              wjung
 ahaj               b                   xenia
```

Give me all male participants
**** part2.q ****
Session.MDGroup.Participants.Participant(a).Sex="Male"

select *
from participant_id
where sex = male ;

```
 session_nb | participant_nb |   type    |  name   | fullname |     code     |            role             | age | sex
| anonymous
------------+----------------+-----------+---------+----------+--------------+-----------------------------+----
+------+-----------
etc.etc.etc.etc.etc.

      1630 |              0 | subject   | necti   |          | sub70_cz     |                             | 18  | male | false
      1631 |              0 | subject   | necti   |          | sub70_cz     |                             | 18  | male | false
      1632 |              0 | subject   | necti   |          | sub70_cz     |                             | 18  | male | false
      1633 |              0 | subject   | necti   |          | sub70_cz     |                             | 18  | male | false
      1634 |              0 | subject   | necti   |          | sub70_cz     |                             | 18  | male | false
      1635 |              0 | subject   | necti   |          | sub70_cz     |                             | 18  | male | false
      1636 |              0 | subject   | necti   |          | sub70_cz     |                             | 18  | male | false
      1637 |              0 | subject   | necti   |          | sub70_cz     |                             | 18  | male | false
      1638 |              0 | subject   | necti   |          | sub70_cz     |                             | 18  | male | false
      1639 |              0 | subject   | necti   |          | sub70_cz     |                             | 18  | male | false
(903 rows)
```

Give me all combinations of one male and one female participant in a session
**** part4.q ****
Session.MDGroup.Participants.Participant(a).Sex="Male"
Session.MDGroup.Participants.Participant(b).Sex="Female"

```
select p1.session_nb, p1.sex, p1.participant_nb, p2.sex, p2.participant_nb
from participant_id p1, participant_id p2
where p1.session_nb = p2.session_nb
and p1.sex = male
and p2.sex = female ;
```

```
 session_nb | sex  | participant_nb |  sex   | participant_nb
------------+------+----------------+--------+----------------
        753 | male |              0 | female |              1
        754 | male |              0 | female |              1
        755 | male |              0 | female |              1
        756 | male |              0 | female |              1
        757 | male |              0 | female |              1
etc.etc.etc.etc.
       1636 | male |              0 | female |              1
       1637 | male |              0 | female |              1
       1638 | male |              0 | female |              1
       1639 | male |              0 | female |              1
(609 rows)
```

Give me all combinations of one female and another female in sessions where
the content language is "French"
**** part5.q ****
Session.MDGroup.Participants.Participant(a).Sex="Female"
Session.MDGroup.Participants.Participant(b).Sex="Female"
Session.MDGroup.Content.Languages.Language(o).Name="French"

COMMENT: : There is no French in L2.dbf
Show me all Content.Languages with their count

```
select language_name, count(session_nb)
from project_content_languages
group by language_name ;
```

```
    language_name    | count
---------------------+-------
 croatian            |     3
 czech               |   887
 czech sign language |    62
 english             |   298
 german              |   989
```

```
 italian            |    65
 polish             |   604
 russian            |    31
 turkish            |     6
 unknown            |     2
 vietnamese         |    37
(11 rows)
```

COMMENT: : Show me the languages of the sessions in which there are two females

```
select p1.session_nb, p1.participant_nb, p2.participant_nb, pcl.language_name
from participant_id p1, participant_id p2, project_content_languages pcl
where p1.session_nb = p2.session_nb
and p1.sex = female
and p2.sex = female
and p1.participant_nb < p2.participant_nb
and pcl.session_nb = p2.session_nb ;
```

```
 session_nb | participant_nb | participant_nb |    language_name
------------+----------------+----------------+---------------------
        764 |              0 |              1 | czech
        765 |              0 |              1 | czech
        766 |              0 |              1 | czech
        767 |              0 |              1 | czech
        768 |              0 |              1 | czech
etc.etc.etc.etc.etc.
       1613 |              0 |              1 | czech
       1614 |              0 |              1 | german
       1614 |              0 |              1 | czech
(559 rows)
```

COMMENT: : Show me the number of sessions per language with two female participants

```
select pcl.language_name, count(p1.session_nb)
from participant_id p1, participant_id p2, project_content_languages pcl
where p1.session_nb = p2.session_nb
and p1.sex = female
and p2.sex = female
and p1.participant_nb < p2.participant_nb
and pcl.session_nb = p2.session_nb
group by pcl.language_name ;
```

```
    language_name    | count
---------------------+-------
 czech               |   347
 czech sign language |    33
 english             |    64
 german              |   103
 vietnamese          |    12
(5 rows)
```

Give me all combinations of one female and another female in sessions where the content language is "czech sign language"

```
select p1.session_nb, pcl.language_name, p1.name, p1.sex, p2.name, p2.sex
from participant_id p1, participant_id p2, project_content_languages pcl
where p1.session_nb = p2.session_nb
and p1.sex = female
and p2.sex = female
and p1.participant_nb < p2.participant_nb
and pcl.session_nb = p2.session_nb
and pcl.language_name = czech sign language ;
```

```
 session_nb |    language_name    | name   | sex   | name   | sex
------------+---------------------+---------+--------+---------+--------
       1243 | czech sign language | klaud   | female | barbara | female
       1244 | czech sign language | klaud   | female | barbara | female
etc.etc.etc.etc.
       1292 | czech sign language | petlach | female | barbara | female
       1293 | czech sign language | petlach | female | barbara | female
(33 rows)
```

Give me all female participants with the age of 30 to 39
**** part7.q ****
Session.MDGroup.Participants.Participant(a).Age="3[0-9]"
Session.MDGroup.Participants.Participant(a).Sex="Female"

```
select distinct name, age
from participant_id
where sex = female
and age >= 30
and age <= 39 ;
```

```
  name   | age
---------+-----
 barbres | 33
 muck    | 36
 papst   | 36
 polit   | 34
 rike    | 34
 sil     | 30
(6 rows)
```
