INFORMATION SOCIETY TECHNOLOGIES
(IST)
PROGRAMME

OpenMolGRID

SPECIFICATION OF THE DATABASE ACCESS INTERFACE

Contract Reference:	**IST-2001-37238**
Document identifier:	**OpenMolGRID-4-D4.1c-0103-1-3-DBAT**
Date:	**07/10/03**
Work package:	**WP4: Grid Integration**
Partner	**FZJ, UU**
Lead Partner	**FZJ**
Document status:	**APPROVED**
Classification:	**INTERNAL**
Deliverable identifier:	**D4.1c**

Abstract: This document describes the server side tools needed to realise database access within OpenMolGRID

Delivery Slip

	Name	Partner	Date
From	Bernd Schuller	FZJ	18/09/2003
Verified by	Mathilde Romberg	FZJ	07/10/2003
Approved by	G.H.F.Diercksen (TC) R.Ferenczi (QE)	OMC CGX	15/11/2003 07/04/2004

Document Log

Issue	Date	Comment	Author
1-0	1/06/2003	Submitted to EC	M. Romberg, B. Schuller
1-1	16/09/2003	Submitted for internal review	M. Romberg, B. Schuller
1-2	07/10/2003	none	M. Romberg
1-3	07/04/2004		M. Romberg

Document Change Record

Issue	Item	Reason for Change
1-1	Improved document structure, increased level of technical detail, added discussion of the design choices made. Interface description reworked. Adopted new document template.	External review Brussels, 17/06/2003
1-2	typing errors corrected	Internal Review Process
1-3	project information	

Files

Files in this section relate to actual storage locations on the BSCW server located at https://hermes.chem.ut.ee/bscw/bscw.cgi. The URL below describes the location on BSCW from the root OpenMolGRID directory

Software Products	User files / URL
Word 2000/XP	OpenMolGRID/Workpackage n/Deliverables/ OpenMolGRID-4-D4.1c-0103-1-3-DBAT

Project information

Project acronym:	OpenMolGRID
Project full title:	Open Computing GRID for Molecular Science and Engineering
Proposal/Contract no.:	IST-2001-37238
European Commission:	
Project Officer:	Annalisa BOGLIOLO
Address:	European Commission - DG Information Society F2 - Grids for Complex Problem Solving B-1049 Brussels Belgium
Office	BU31 4/79
Phone:	+32 2 295 8131
Fax:	+32 2 299 1749
E-mail	annalisa.bogliolo@cec.eu.int
Project Coordinator:	Mathilde ROMBERG
Address:	Forschungszentrum Jülich GmbH ZAM D-52425 Jülich Germany
Phone:	+49 2461 61 3703
Fax:	+49 2461 61 6656
E-mail	m.romberg@fz-juelich.de

Contents

1. Introduction

1.1. Purpose and Scope

This document describes the server side tools that are needed to provide database access within OpenMolGRID, based on the underlying Grid infrastructure, UNICORE. The interfaces of these database access tools to UNICORE and to the underlying database are detailed.

1.2. Document Overview

Workpackage four (WP4) of the OpenMolGRID project deals with the Grid integration of components being developed in WP1 – WP3.

From WP1 the data warehouse has to be integrated through an access module and a corresponding user interface based on the underlying Grid infrastructure UNICORE. UNICORE is described in detail in [7].

Access to databases is currently not provided by UNICORE, but it is essential for the data warehousing part of the project (WP1). As the data warehouse can be seen as 'just another database', WP4.1 will focus on a solution for database access in a general, flexible and extensible way. In this task the open interfaces of the UNICORE infrastructure will be used to develop the necessary additions to the system. The Client components used to access databases are described in [6].

On the server side of the Grid system the queries and upload requests have to be transformed into requests for the target database, executed, and the resulting output has to be made available in a format suitable for further processing.

The basic architecture can be summarised as follows:

Figure 1: Summary of basic architecture

We see several advantages with this concept. By introducing a "middle tier" database access tool (DBAT), we achieve our aims of flexibility and extensibility, while maintaining a consistent, seamless user view on databases. The DBAT can be extended and modified to fit the needs of the evolving system, its interface with the Grid infrastructure UNICORE can be adapted as UNICORE moves towards the Open Grid Services Architecture. The XML output format is well suited for further automated processing of results.

1.3. Document Structure

The document is structured as follows: section 2 gives a brief overview about the databases that have to be integrated into the OpenMolGRID system, such as their query formats and access restrictions. Section 3 outlines the requirements on the database access architecture of OpenMolGRID. The architecture chosen is described in detail in section 4.

2. Database Descriptions

This section deals with particular databases to be considered in the OpenMolGRID project. The main subject of these databases is toxicity. Subject to analysis are relevant databases with their access methods and interfaces, licensing, and security issues. The relevant data sources can be grouped into the following classes:

- Web-accessible with no access restrictions (examples: ECOTOX, NTP)
  - Data selection by queries
  - Download of complete database
- Web-accessible with user ID and password
- Licensed databases available on specific media (CD) (examples: TOMESCPSTM Integrated Index™ Query; IUCLIC)

The data available on CD only are not designed to be accessed over the internet and cannot be included into the data warehouse for licensing reasons. Therefore it has been decided to not analyse them any further.

The following subsections briefly describe the examples for web-accessible databases and relational databases; the OpenMolGRID data warehouse will be of the latter type.

2.1. Web-accessible Databases

The public databases to be integrated into the OpenMolGRID system are described in detail in [2], [3] (ECOTOX), and [4] (NTP). For the purpose of this document they can be characterised in the following way:

- They are accessed via the HTTP protocol
- They can not be queried via a standard query language (e.g. SQL)

2.2. Relational Databases

Integrating and accessing relational database management systems (RDBMS) like Oracle or PostgreSQL ([8]) will be very important within OpenMolGRID, since the OpenMolGRID data warehouse (MOLDW) itself will be implemented as a relational database (PostgreSQL has been selected for this purpose [1]).

Relational databases support complex queries using some dialect of SQL.

3. Requirements for Database Access

3.1. User Requirements

The OpenMolGRID users want to access the data warehouse and other databases with related content such as ECOTOX or NTP in the same seamless way.

The output from the data source should be available in different formats for viewing or further processing, independent of the original format at the data source. The output from data access has to be made available in human readable format for the users to be able to examine the result and to trim it for further processing steps.

3.2. Security Requirements

It is foreseen that some data sources to be accessed within OpenMolGRID will have restricted access. While this is not an issue for the databases considered so far, the database access architecture must take access restrictions into account.

3.3. General Considerations

Independent of the original output format of the database considered, the output data has to be available in a format that is suitable for further automated processing within the OpenMolGRID system.

It should be possible to easily integrate new data sources.

4. Specification of the Adapter for Database Access

4.1. Principles, Design Choices Made

The basic mechanism to access databases via UNICORE is realised in the following way.

For each database to be integrated a program called Database Access Tool (DBAT) will be installed on a UNICORE Vsite, which can be executed by the UNICORE target system interface (TSI) on behalf of a user. The DBAT contains parts that are specific for the given database. From the UNICORE point of view, the DBAT is a resource of type "application".

Information about this resource (and therefore the given database) can be provided in an application specific metadata file and queried via standard UNICORE mechanisms. This information includes database specifics such as table names and field names, as described in more detail in section 4.4.3. This allows a truly seamless integration of new databases into the system, provided they contain the same class of information, such as "toxicity information about chemical substances" or "structure of chemicals".

Furthermore, this approach fits in very well with the overall approach taken within OpenMolGRID (see [7]), where UNICORE applications are used to "wrap" software packages, and a flexible, extensible control layer is provided. In this respect, database access is treated on the same basis as, for example, descriptor calculation, and can be easily integrated into the complex workflows considered within OpenMolGRID.

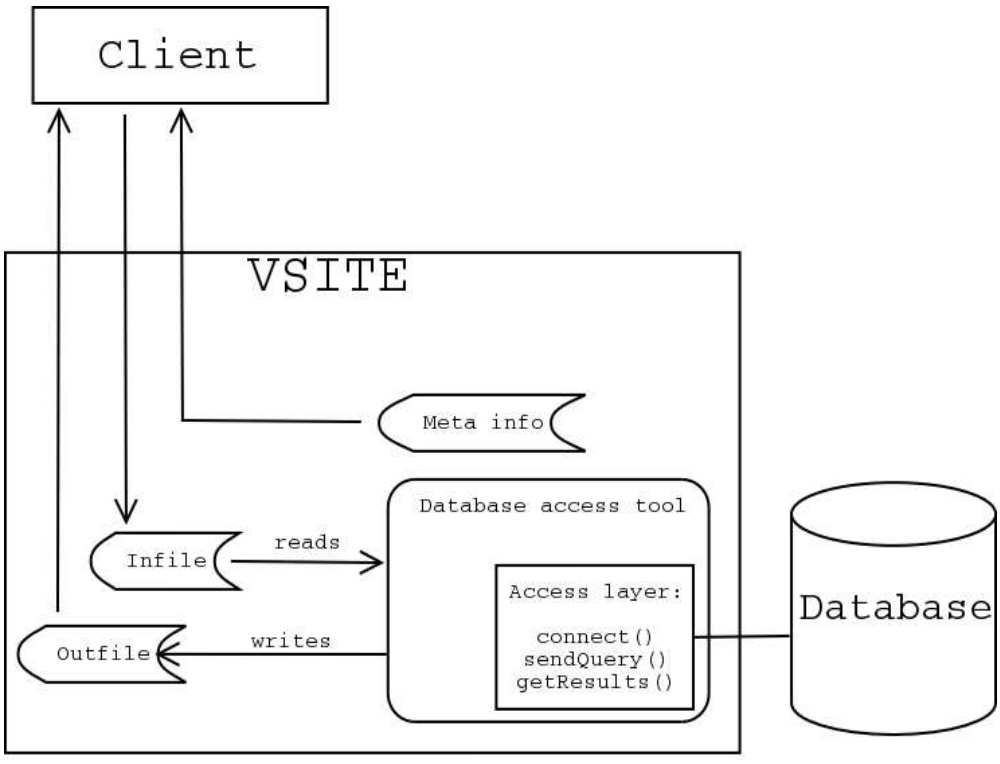

Figure 2 Architecture for accessing databases via UNICORE

Figure 2 explains the database access concept in detail: The Client reads the metadata from the Meta info file to be able to prepare the query. Using the user's selections it generates the query and sends it in file `infile` to the Vsite, together with the request for the application DBAT. The UNICORE TSI calls DBAT specifying input and output files:

```
dbat infile outfile
```

The database access tool performs the following tasks:

- the input file is opened, parsed, and the query is extracted,
- the query is translated into the database specific format, if needed,
- a connection to the database is established and the query is sent,
- results are generated from the database response, and
- the result file containing results and administrative data is generated and written to the local file system as `outfile`.

Finally the result file is transferred back to the client via the standard UNICORE transfer mechanism.

4.2. Discussion of the Approach Chosen

In the original workplan, it was planned to modify the UNICORE target system interface (TSI) to provide access to a database.

The approach taken here, using applications to access databases, is superior to the approach that involves modifying the TSI, for the following reasons:

- The basic UNICORE components (such as NJS and TSI) and their interfaces are not modified, and so dependencies on these are minimal. This gives more flexibility in the light of the migration to an OGSA-compliant UNICORE, since at this point it is not yet clear exactly how an OGSA-compliant UNICORE will look like.
- UNICORE applications offer a metadata mechanism which allows to solve several problems related to a seamless user view on databases.
- Database access is treated on the same basis as other software packages (e.g. descriptor calculation software). This allows an easy integration of database access into complex, automated workflows.

4.3. Interface Description: DBAT – Database

The interface between the DBAT and the corresponding database is defined by the communication protocols and the authorization scheme.

The communication between DBAT and the database can be broken down to three basic functions:

- `connect()`: handles authorization issues and open a connection to the database;
- `sendQuery()`: generates a native query from the SQL query contained in the input file, and sends it to the database;
- `getResults()`: gets the result of the query.

4.3.1. Communication Mechanisms

The database access tool handles the communication with the database, which of course depends on the database under consideration. For example, in the case of the NTP Web database a query must be sent as an HTTP request to the Web server hosting the database.

Since DBAT is written in Java, the JDBC application program interface (Java database connectivity, see [9]) can be used to handle the communication to relational databases. An example DBAT implementation for MySQL has been written, which can be used as a blueprint for accessing JDBC compliant databases, ideally only changing the JDBC driver name and possibly the logon procedure.

For non-JDBC databases, the SQL query must first be translated to the native form, for example in the case of the NTP database.

4.3.2. Security and Authorization

Besides public Web-based databases such as ECOTOX and NTP where access is not restricted, there are data sources which require authorization information like userid and password.

In this case, the additional security parameters will be requested from the user and provided to the DBAT. The input file specifications (see below) contain provisions for it. The DBAT will then read userid and password (or the certificate) and use it to access the database.

The current example implementation of a DBAT for JDBC compliant relational databases provides the following authorization schemes

public	no authentication needed, logon to the database is anonymously.
automatic	the users' Unix username is mapped to a database username. The DBAT logs on using this username and no password. The user need not specify anything, since his Unix logon name is handled by the UNICORE infrastructure (NJS and UUDB). This allows the database administrator to assign roles and grant privileges to certain users.
private	the user has to specify his username and password in the client GUI, which will transmit the authentication information as part of the input to the DBAT. The DBAT will try to logon to the database using this information.

From the UNICORE point of view, the "automatic" authorization scheme is the most consistent, but of course the database administrator or owner might wish to enforce "private" authorization.

4.4. Interface Description: UNICORE Client Plugin – DBAT

This interface contains of three parts:

input file format	defines which information is contained in the file sent from client to DBAT, and in what format, this is covered in section 4.4.1
output file format	the DBAT output format is described in section 4.4.2, while transformations to other formats are described in section 4.4.3
application metadata	sections 4.4.3 and 4.4.5

4.4.1. Input File Format

Input to DBAT will be an XML file containing the query and user authentication. A sample file looks as follows:

```
<?xml version="1.0"?>
<dbat_input>
    <query>SELECT * FROM toxicity WHERE cas=12345
    </query>
    <user>SampleUserName</user>
    <password>SomePassword</password>
</dbat_input>
```

There are some important things to note about this format:

- Username and password have to be specified only if the data source is explicitly marked as "private" in the metadata (see section 4.4.5).
- The query is in SQL. An obvious possible extension could be to specify the query language in the metadata as well (as for example done in the OGSA-DAI effort [11]). This, however would enormously complicate the client component that builds the query from the user selections in the GUI. The decision between SQL and an XML based query language, e.g., XQuery, was decided by the fact that the data warehouse (MOLDW) will be implemented as an SQL database.

4.4.2. Output File Format

As output format, we have chosen a custom XML format, since XML is easy to process and convert into other (not necessarily XML) formats using XSLT [12].

The results received from a database can be visualised as a table. This will be written row by row to the output file

The following example shows the XML output produced using the DTD. Consider the following result table:

Id	name	weight
1	Hydrogen	1
12	Magnesium	24
33	Arsenic	75

Given that ‘id’ and ‘weight’ are integer and ‘name’ is of type string, the following result file would be produced:

```
<?xml version=”1.0”?>
<dbat_output>
    <status>0
    </status>
    <info>request successful
    </info>
    <results>
         <column_info>
              <no_of_columns>3<no_of_columns>
              <column>
                   <no>1</no>
                   <label>id</label>
                   <typeSQL>LONG</typeSQL>
                   <typeJDBC>4</typeJDBC>
              </column>
              <column>.....</column>
         </column_info>
         <row>
              <value>1</value>
              <value>Hydrogen</value>
              <value>1</value>
         </row>
         <row>
              <value>12</value>
              <value>Magnesium</value>
              <value>24</value>
         </row>
         <row>
              <value>33</value>
              <value>Arsenic</value>
              <value>75</value>
         </row>
    </results>
</dbat_output>
```

As can be seen from the example, all information needed to reconstruct the result table from the XML file is included, such as column type information. In addition to providing the appropriate information to reconstruct the result table, information will also be available relating to the success or failure of the query.

The output file format can be succinctly specified in the form of an XML document type definition:

```
<!--
```

```
  dbat_output.dtd
  Specification of DBAT output format

  Version: 1.0, April 1, 2003
  OpenMolGRID, http://www.openmolgrid.org
-->
<!ELEMENT info (#PCDATA)>
<!ELEMENT status (#PCDATA)>
<!ELEMENT no_of_columns (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT label (#PCDATA)>
<!ELEMENT typeSQL (#PCDATA)>
<!ELEMENT typeJDBC (#PCDATA)>
<!ELEMENT typeOMG (#PCDATA)>
<!ELEMENT column (no,label,typeOMG?,typeSQL?,typeJDBC?)>
<!ELEMENT column_info (no_of_columns,column*)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT row (value+)>
<!ELEMENT results (column_info,row*)>
<!ELEMENT dbat_output (info?,status?,results)>
```

Given the advantages of XML as an output format, namely flexibility and easy automatic processing, there is one major disadvantage relating to the fact that the XML format chosen introduces a rather large overhead, which can become noticeable once large result sets have to be transmitted. The overhead is worst in case of result sets having a lot of very small elements, say a single column of integers, where every integer "n" is encoded as `<row><value>n </value></row>` in the XML result file. Since UNICORE packs files (using the ZIP algorithm) before transmitting them, the network and bandwidth overhead introduced by using XML is considered not so important.

4.4.3. Converting the Output: XSLT

Users may need the output from a database query for direct use in programs such as Matlab or Excel. Therefore, the XML output from the database access tool needs to be transformed into other formats (for example comma-separated values, or tab-delimited). For this purpose, XSLT [12] is used. Though these transformations are not actually part of the database access tool, normally UNICORE sites offering database access will also offer output transformations. Since XSLT transformations can be computationally demanding, they are done server-side. On the client side, the database access plugin (GUI) offers the possible output transformations to the user.

4.4.4. Querying Database Capabilities: Table Names and Fields

Seamless access requires a consistent user interface on the client side. All databases under consideration contain toxicity information. These should be queryable using the same user interface. In order to achieve seamlessness, some sort of abstraction is needed.

A solution for this is to use abstract table names and abstract fields in the user interface. The user input is compiled into an SQL query by the user interface. If needed, this "abstract query" is translated by the DBAT into the real query for the database (e.g., HTTP request).

When users select the database query plugin in their UNICORE client and select a database on a given Vsite for querying, the plugin requests application specific information from that Vsite. This information will include table names and fields that the plugin needs to build a valid query. The UNICORE incarnation database (IDB) provides the application specific information as part of the software resource as described below.

In the UNICORE IDB, an Application is described as

```
   APPLICATION description name version meta_data_file_name
```

where the content of the file called *meta_data_file_name* is sent to the client as part of the Application resource. So, for example,

```
APPLICATION "ECOTOX Database" dbat_ecotox_web 1.0 ecotox_meta.xml
```

The metadata is used by the UNICORE Client plugin for different purposes:

- Information about the selected data source can be shown to the user;
- The graphical user interface of the plugin is modified and adapted to the selected data source. This includes enabling/disabling certain GUI elements;
- Generation of a query that is valid for the data source selected.

4.4.5. Contents of the Metadata File

The metadata file contains the following information encoded in XML:

- Database name
- Access restrictions
- Information about the database intended for the user
- Table names
- Field names and field types.

An XML document type definition (DTD) is defined (dbat_meta.dtd) and is given in section 4.4.6. It defines the various XML tags used in the metadata file. A brief description of the tags used is as follows.

`<dbat_meta>`	root element, specifies the XML document type
`<database>`	contains all information that refers to this database; it has two attributes: • "name": specifies the name of the database. • "access": indicates whether the user needs a user name and password (or some other authentication) to access the database.
`<description>`	sub-tag of `<database>`; description of the database intended for the user. At the present time, it is intended that this is a paragraph of html code, that can be included in a Java GUI element.
`<table>`	sub-tag of `<database>`; has the name of the table as an attribute and contains sub-tags describing the table and the fields in it. Multiple tables can be present.
`<field>`	sub-tag of `<table>` with the following attributes: • "name": the name of the field • "type": this is the "high-level" data type of the field, for example "2D structure", or "CAS number" or "Chemical formula". A draft for a complete list of these data types is available in Deliverable D1.3 ([5]) of the OpenMolGRID project. This type information is very important, since it is needed to process the results, or to visualize them. • "description": this can be used in the GUI as well, to present a description of output data to the user. • "query": This attribute is used to specify whether fields can be queried. For example, a database might include chemical structure information, but may not allow to query it.

In this way, the client receives a list of tables, their descriptions and the fields contained in those tables.

Example

Let us imagine a simple Web database, where toxicity information is queried by specifying a CAS number or a chemical name. The result is plain text. The XML file giving the required information looks like this:

```
<?xml version="1.0"?>
<dbat_meta>
  <database name="TXT_WEB"
    <description>Web toxicity database</description>
    <table name="TOXICITY">
      <description>Toxicity information</description>
      <field name="OMG_CAS" query="yes"/>
      <field name="OMG_CHEMNAME" query="yes"/>
  <field name="INFOTEXT" type="text/plain" query="no"/>
    </table>
        ....
    </database>
</dbat_meta>
```

This information allows the client plugin to build a valid SQL query like

```
    SELECT * FROM TOXICITY WHERE OMG_CHEMNAME="Benzene"
```

Furthermore, the plugin knows that only CAS number or chemical name can be used in queries and that there will be a plain text column in the result.

The fields and tables specified in the metadata file are *abstract*, since they do not necessarily correspond to actual tables and fields on the target database. The translation to the real database information will be done by the database access tool, which, together with its metadata file, is the only part of the architecture that has to be modified whenever a new database is included.

4.4.6. Metadata Specification

The metadata file format specification (XML DTD) is given below:

```
<!--
  dbat_meta.dtd
  Specification of DBAT metadata format
  Version: 1.0, April 14, 2003
  OpenMolGRID, http://www.openmolgrid.org
-->
<!ELEMENT field EMPTY>
<!ATTLIST field name CDATA #REQUIRED
                description CDATA
                type CDATA #REQUIRED
         query CDATA #REQUIRED
  >
<!ELEMENT description (#PCDATA)>
<!ELEMENT table (description,field+)>
<!ATTLIST table name CDATA #REQUIRED>
<!ELEMENT database (description,table+)>
<!ATTLIST database name CDATA #REQUIRED
                   access CDATA #REQUIRED
  >
<!ELEMENT dbat_meta (database)>
```

4.5. Implementation Details

The database access tools will be implemented in Java, mainly for reasons of operating system independence, since even though most servers in OpenMolGRID will run Linux, one has to consider Solaris or even Windows based servers as future possibilities. Apart from this, a future inclusion of DBAT into a Web service context will be simplified using Java as a programming language.

So far, two database access tools have been implemented. The first (called DBAT_NTP) is used to access the NTP database (see [4]), and as such is highly customised and not reusable for other purposes without major modification.

The second implementation, DBAT_MYSQL deals with the popular MySQL database [10]. It is highly portable and reusable, since it uses JDBC to handle the communication to the database and implements the full security options detailed in section 4.3.2. This implementation will be used as the basis for DBATs for relational databases, especially the OpenMolGRID data warehouse.

4.6. Accessing Web Databases via UNICORE

This section will discuss how the proposed architecture can be used to access databases that are web-based. Two examples are described here, NTP and ECOTOX.

4.6.1. The NTP Database

The NTP database uses HTML query forms that call a CGI-script on their web server. The original NTP query form (as taken from their web page, see [4]) is the following:

```
<HTML><HEAD><TITLE>NTP Chemistry Health and Safety Search</TITLE>
<link rel=”stylesheet”
href=”http://ntp-server.niehs.nih.gov/stylesheets/pre1.css”>
</HEAD>
<BODY BACKGROUND=http://ntp-server.niehs.nih.gov/Icons/cn.jpg>
<FORM method=POST ACTION=”/cgi/iH_Indexes/Chem_H&S/Chem_H&S.cgi”
TARGET=”main”>
<h4>Chemistry H&S Search:</h4> <input name=”KEYWORDS” size=20 align=right>
<input type=submit value=”Go”><p>
Enter a CAS No., chemical name or synonym.
<input type=”hidden” name=”SUBSTRING” value=”true”>
<input type=”hidden” name=”HEADER” value=”IH_Chem_H&S_Header”>
<input type=”hidden” name=”FOOTER” value=”IH_Chem_H&S_Footer”>
<input type=”hidden” name=”INDEX” value=”Chem_H&S.idx”>
<input type=”hidden” name=”LANGUAGE” value=”nglish_fr”>
<input type=”hidden” NAME=”MAXHITS” VALUE=”99”></FORM>
</body></html>
```

The CAS or name of the chemical to search for go into the KEYWORDS field. Additionally, the fields of type “hidden” have to be provided as well, since the DB server uses them internally.

To make NTP accessible via UNICORE, a specific database access tool, called DBAT_NTP, has been developed which connects to the NTP Web server via HTTP, sends a valid query and fetches the output. The classes in the standard Java package `java.net` can be used to take care of HTTP communication. For example, the HTTP connection can be established in a few lines of Java code:

```
URL url = null;
HttpURLConnection conn = null;
String DB_URL=
 “http://ntpdb.niehs.nih.gov/cgi/iH_Indexes/Chem_H&S/Chem_H&S.cgi”;

public boolean connect() {
  try{
    url = new URL(DB_URL);
    conn = (HttpURLConnection)url.openConnection();
    return true;
```

```
    } catch(Exception e){ }
    return false;
   }
```

The second task is to provide a facility translating the SQL query into a valid NTP query. In the case of NTP, the structure of possible queries is very simple. It is sufficient to get the needed keywords via a text search.

For example,

```
    SELECT * FROM TOXICITY WHERE OMG_CHEMNAME=Benzene
```

will be translated into the URL-encoded form

```
KEYWORDS=Benzene&+&INDEX=Chem_H%26S.idx&SUBSTRING=true&LANGUAGE=English_fr&MAXHITS
=9999
```

using the chemical name(s) given in the WHERE clause of the SQL query as `KEYWORDS`. Upon posting it to the database server, NTP returns a HTML page containing links to text files with toxicity information about the specified chemical. These are evaluated and followed to get the texts themselves. The text files are then included in the XML result file.

4.6.2. The ECOTOX Database

ECOTOX has two different user interfaces as described in [2] and [3].The one more useful for direct access via DBAT is the “simple query” available at http://www.epa.gov/ecotox/quicksearch.html. The principle is the same as for NTP, but more options are available and the queries can be more complex.

5. References

[1] Deliverable D1.1a,
Specification of software components for UNICORE-compliant data input/output format, complementary data warehousing methods, and protocols and a user interface defining process control parameters, process logic, and resource assignment

[2] Deliverable D1.1b,
ECOTOX – Terretox data specification

[3] Deliverable D1.1c,
ECOTOX – Aquire data specification

[4] Deliverable D1.1d,
NTP data specification

[5] Deliverable D1.3,
Properties and Priorities of the Data for Pharmaceutical and Phytopharmaceutical Compounds

[6] Deliverable D4.1a,
Specification of the generic user interface for database access

[7] Deliverable D4.5a,
Description of the OpenMolGRID Grid architecture, security architecture, and infrastructure and the deployment of the project's testbed

[8] SQL: "Practical PostgreSQL. A Hardened, Robust, Open Source Database"
edited by John C. Worsley, Joshua D. Drake, Jonathan Gennick

[9] Java Data Base Connectivity, http://java.sun.com/products/jdbc

[10] MySQL, http://www.mysql.com

[11] OGSA database access and integration, http://www.ogsadai.org.uk

[12] Extensible Stylesheet Language for Transformations (XSLT), http://w3c.org /Style/XSL

6. Terminology / Glossary

AJO	Abstract Job Object
CAS	Chemical Abstracts Service
CD	Compact Disc
CGI	Common Gateway Interface
DB	DataBase
DBAT	Database Access Tool
DTD	Data Type Definition
FZJ	Forschungszentrum Jülich
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDB	Incarnation DataBase
JDBC	Java DataBase Connectivity
JRE	Java Runtime Environment
MOLDW	OpenMolGRID Data Warehouse
MySQL	Open Source Relational Database
NJS	Network Job Supervisor
NTP	National Toxicity Program
OGSA	Open Grid Services Architecture
OGSA-DAI	Open Grid Services Architecture – Data Access and Integration
RDBMS	Relational DataBase Management System
SQL	Structured Query Language
TSI	Target System Interface
UNICORE	Uniform Interface to Computer Resources
URL	Unified Resource Locator
UU	University of Ulster
UUDB	UNICORE User DataBase
Vsite	Virtual Site
WP	Work Package
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language for Transformations