The Geodise Toolboxes for Jython

Installing the Geodise XML, Compute and Database Toolboxes

Release: GeodiseLabPy v.1.0.2

Version: INSTALLPy.doc 1.1.1

Title: The Geodise Toolboxes – Installation Instructions

Authors: Dr Graeme Pound, g.e.pound@soton.ac.uk
Dr Jasmin Wason, j.l.wason@soton.ac.uk
Dr Marc Molinari, m.molinari@soton.ac.uk
Dr Hakki Eres, hakki.eres@soton.ac.uk
Dr Zhuoan Jiao, z.jiao@soton.ac.uk

PI: Prof Simon Cox, s.j.cox@soton.ac.uk

Web: http://www.geodise.org/

Acknowledgement:

The development of the Geodise toolboxes for public release has been supported by the managed programme of the Open Middleware Infrastructure Institute (http://www.omii.ac.uk/).

Contents

Prerequisites

The prerequisites required by the Geodise Toolboxes for Jython are:

- Java 2 Platform, Standard Edition, v 1.4.2 (J2SE)
- Jython 2.1

Installing Java 1.4.2

Java v 1.4.2 (JRE or SDK) can be downloaded from http://java.sun.com/j2se/1.4.2/. Please follow the installation instructions suitable for your platform.

Installing Jython 2.1

The Jython implementation of the Python language requires the Java platform to be present before it can be installed and run. The self-extracting installer *jython-21.class* should be downloaded from http://www.jython.org/.

To invoke the graphical Jython installer run the following command from the directory containing *jython-21.class*.

```
java jython-21
```

For detailed installation instructions, including the syntax for invoking the non-graphical installer, refer to http://www.jython.org/install.html.

Installing the GeodiseLab toolboxes

To install the GeodiseLab toolboxes for Jython it is necessary to add the several Java libraries to the Jython classpath, and to make the GeodiseLab modules available to the Jython path.

Extract the toolboxes

Unzip *GeodiseLabPy-1.0.2.zip* to a location <GEODISELABPY>.

Install the GeodiseLab toolboxes

Do either of the following:

(a) Run the installation script

Navigate to the directory <GEODISELABPY> and run the installation script. On Windows:

```
>> install.bat
```

On Linux:

```
>> ./install.sh
```

The user will be queried for the location of the directory containing the Jython executable, and the installation directory to which the GeodiseLab Toolboxes will be copied. An example of the installation of the GeodiseLab Toolboxes for Jython on Windows XP is shown below:

```
>> install.bat
```

```
Buildfile: build.xml

query_jythonhome:
    [input] Please enter the path to the directory
containing the Jython executable [default C:\Documents and
Settings\gep\Jython-2.1\]:
```

```
C:\Jython-2.1\
```

```
check_jythonhome:

query_installpath:
    [input] Please enter the path of the installation
directory [default C:\Documents and
Settings\gep\GeodiseLabPy\]:
```

C:\Documents and Settings\gep\GeodiseLabPy\

```
test_installpath:

check_installpath:

load_jythonregistry:

delete_installpath:

make_installpath:
    [mkdir] Created dir: C:\Documents and
Settings\gep\GeodiseLabPy

os_cmds:

pre-install:
     [copy] Copying 120 files to C:\Documents and
Settings\gep\GeodiseLabPy
     [copy] Copying 1 file to C:\Jython-2.1
     [copy] Copying 1 file to C:\Jython-2.1
    [touch] Creating C:\Documents and
Settings\gep\GeodiseLabPy\uninstall.bat

BUILD SUCCESSFUL
Total time: 5 seconds
Press any key to continue . . .
```

This installation script will edit the Jython executable (' jython.bat' on Windows or

'jython' on Linux) and the Jython 'registry' file. When the Jython executable is invoked the GeodiseLab modules will be available.

(b) Manually install the toolboxes

Set the Jython classpath

Make the following Java libraries available to the Jython environment.

```
<GEODISELABPY>\lib\axis.jar
<GEODISELABPY>\lib\cog-jglobus.jar
<GEODISELABPY>\lib\cog-lib.jar
<GEODISELABPY>\lib\commons-discovery.jar
<GEODISELABPY>\lib\commons-logging.jar
<GEODISELABPY>\lib\cryptix-asn1.jar
<GEODISELABPY>\lib\cryptix.jar
<GEODISELABPY>\lib\cryptix32.jar
<GEODISELABPY>\lib\gdcompute.jar
<GEODISELABPY>\lib\gddatabase.jar
<GEODISELABPY>\lib\jakarta-regexp-1.3.jar
<GEODISELABPY>\lib\jaxrpc.jar
<GEODISELABPY>\lib\jce-jdk13-120.jar
<GEODISELABPY>\lib\jcert.jar
<GEODISELABPY>\lib\jdom.jar
<GEODISELABPY>\lib\jgss.jar
<GEODISELABPY>\lib\jnet.jar
<GEODISELABPY>\lib\jnumeric-0.1a3.jar
<GEODISELABPY>\lib\jsse.jar
<GEODISELABPY>\lib\jug.jar
<GEODISELABPY>\lib\junit.jar
<GEODISELABPY>\lib\log4j-1.2.8.jar
<GEODISELABPY>\lib\puretls.jar
<GEODISELABPY>\lib\saaj.jar
<GEODISELABPY>\lib\sunjce_provider.jar
<GEODISELABPY>\lib\wsdl4j.jar
<GEODISELABPY>\lib\xalan.jar
<GEODISELABPY>\lib\xercesImpl.jar
<GEODISELABPY>\lib\xml-apis.jar
<GEODISELABPY>\lib\xmlsec.jar
```

This may be done by (i) adding the libraries to the system classpath, or by (ii) editing the Jython startup script.

(i) Adding JAR files to the system environment variable CLASSPATH:

For example using Bash (where $GEODISELABPY is the location of the toolbox):

```
export
CLASSPATH=$CLASSPATH:$GEODISELABPY/lib/axis.jar:$GEO
DISELABPY/lib/cog-jglobus.jar:$GEODISELABPY/lib/cog-
lib.jar:$GEODISELABPY/lib/commons-discovery.jar:$GEO
DISELABPY/lib/commons-logging.jar:$GEODISELABPY/lib/
cryptix-asn1.jar:$GEODISELABPY/lib/cryptix.jar:$GEOD
ISELABPY/lib/cryptix32.jar:$GEODISELABPY/lib/gdcompu
te.jar:$GEODISELABPY/lib/gddatabase.jar:$GEODISELABP
Y/lib/jakarta-regexp-1.3.jar:$GEODISELABPY/lib/jaxrp
c.jar:$GEODISELABPY/lib/jce-jdk13-120.jar:$GEODISELA
BPY/lib/jcert.jar:$GEODISELABPY/lib/jdom.jar:$GEODIS
ELABPY/lib/jgss.jar:$GEODISELABPY/lib/jnet.jar:$GEOD
ISELABPY/lib/jnumeric-0.1a3.jar:$GEODISELABPY/lib/js
se.jar:$GEODISELABPY/lib/jug.jar:$GEODISELABPY/lib/j
unit.jar:$GEODISELABPY/lib/log4j-1.2.8.jar:$GEODISEL
ABPY/lib/puretls.jar:$GEODISELABPY/lib/saaj.jar:$GEO
DISELABPY/lib/sunjce_provider.jar:$GEODISELABPY/lib/
wsdl4j.jar:$GEODISELABPY/lib/xalan.jar:$GEODISELABPY
/lib/xercesImpl.jar:$GEODISELABPY/lib/xm-apis.jar:$G
EODISELABPY/lib/xmlsec.jar
```

For example in Windows XP (where %GEODISELABPY% is the location of the toolbox):

```
Start>Control Panel>System>Advanced>Environment
Variables>System Variables>CLASSPATH>Edit:

Variable value =
.;%GEODISELABPY%\lib\axis.jar;%GEODISELABPY%\lib\cog
```

```
-jglobus.jar;%GEODISELABPY%\lib\cog-lib.jar;%GEODISE
LABPY%\lib\commons-discovery.jar;%GEODISELABPY%\lib\
commons-logging.jar;%GEODISELABPY%\lib\cryptix-asn1.
jar;%GEODISELABPY%\lib\cryptix.jar;%GEODISELABPY%\li
b\cryptix32.jar;%GEODISELABPY%\lib\gdcompute.jar;%GE
ODISELABPY%\lib\gddatabase.jar;%GEODISELABPY%\lib\ja
karta-regexp-1.3.jar;%GEODISELABPY%\lib\jaxrpc.jar;%
GEODISELABPY%\lib\jce-jdk13-120.jar;%GEODISELABPY%\l
ib\jcert.jar;%GEODISELABPY%\lib\jdom.jar;%GEODISELAB
PY%\lib\jgss.jar;%GEODISELABPY%\lib\jnet.jar;%GEODIS
ELABPY%\lib\jnumeric-0.1a3.jar;%GEODISELABPY%\lib\js
se.jar;%GEODISELABPY%\lib\jug.jar;%GEODISELABPY%\lib
\junit.jar;%GEODISELABPY%\lib\log4j-1.2.8.jar;%GEODI
SELABPY%\lib\puretls.jar;%GEODISELABPY%\lib\saaj.jar
;%GEODISELABPY%\lib\sunjce_provider.jar;%GEODISELABP
Y%\lib\wsdl4j.jar;%GEODISELABPY%\lib\xalan.jar;%GEOD
ISELABPY%\lib\xercesImpl.jar;%GEODISELABPY%\lib\xml-
apis.jar;%GEODISELABPY%\lib\xmlsec.jar
```

(ii) Editing the Jython startup script

Alter the value of the '-classpath' argument in the Jython startup script (*jython.bat* under Windows or *jython.sh* under Unix) to include the location of the required Jar files.

For example the edited line of *jython.bat* may resemble:

```
set GEODISELABPY=C:\GeodiseLabPy\

"java.exe" "-Dpython.home=C:\jython-2.1"  classpath
"C:\jython-2.1\jython.jar;%GEODISELABPY%\lib\axis.ja
r;%GEODISELABPY%\lib\cog-jglobus.jar;%GEODISELABPY%\
lib\cog-lib.jar;%GEODISELABPY%\lib\commons-discovery
.jar;%GEODISELABPY%\lib\commons-logging.jar;%GEODISE
LABPY%\lib\cryptix-asn1.jar;%GEODISELABPY%\lib\crypt
ix.jar;%GEODISELABPY%\lib\cryptix32.jar;%GEODISELABP
Y%\lib\gdcompute.jar;%GEODISELABPY%\lib\gddatabase.j
ar;%GEODISELABPY%\lib\jakarta-regexp-1.3.jar;%GEODIS
```

```
ELABPY%\lib\jaxrpc.jar;%GEODISELABPY%\lib\jce-jdk13-
120.jar;%GEODISELABPY%\lib\jcert.jar;%GEODISELABPY%\
lib\jdom.jar;%GEODISELABPY%\lib\jgss.jar;%GEODISELAB
PY%\lib\jnet.jar;%GEODISELABPY%\lib\jnumeric-0.1a3.j
ar;%GEODISELABPY%\lib\jsse.jar;%GEODISELABPY%\lib\ju
g.jar;%GEODISELABPY%\lib\junit.jar;%GEODISELABPY%\li
b\log4j-1.2.8.jar;%GEODISELABPY%\lib\puretls.jar;%GE
ODISELABPY%\lib\saaj.jar;%GEODISELABPY%\lib\sunjce_p
rovider.jar;%GEODISELABPY%\lib\wsdl4j.jar;%GEODISELA
BPY%\lib\xalan.jar;%GEODISELABPY%\lib\xercesImpl.jar
;%GEODISELABPY%\lib\xml-apis.jar;%GEODISELABPY%\lib\
xmlsec.jar;%CLASSPATH%"       org.python.util.jython
%ARGS%
```

Set the Python path

Add the directory containing the GeodiseLab modules to the search path used by Jython. This may be done by (i) editing the Jython registry, or by (ii) appending the path at runtime.

(i) Edit the file called *registry* located in the root directory of the Jython installation (e.g. *c:\jython-2.1*). Append the python.path property with the location <GEODISELABPY>\bin (double slashes will be required for Windows paths). For example:

```
python.path = C:\\GeodiseLabPy\\bin\\
```

By default the python.path property is not set, there are examples at the top of the *registry* file.

(ii) Append the search path at runtime with the following code. This may be the done at the Jython command line or inside a python script. For example:

```
import sys
sys.path.append('c:\\GeodiseLabPy\\bin\\')
```

Uninstalling the GeodiseLab toolboxes

The uninstall script may be invoked to uninstall the GeodiseLab toolboxes only if the installation script has previously be used to install the toolbox. Uninstall scripts are available in both the installation and <GEODISELABPY> directories. On Windows:

```
>> uninstall.bat
```

On Linux:

```
>> ./uninstall.sh
```

XML Toolbox

The Geodise XML Toolbox for Jython provides functionality for converting and storing Python data structures in XML and for loading and parsing XML data into Python data structures.

If you have followed the installation instructions of the Geodise Toolboxes for Jython, the file gdxml.py should be in the python path and thus be available to the Jython environment. You should be able to use the functionality provided by importing the package using:

```
>>> import gdxml
```

at the Jython command prompt.

Test Installation

To test the XML Toolbox and see a demonstration of its capabilities, run the Jython startup script, and enter the following command at the Jython command prompt:

```
>>> import demo_xml
```

A short script will be run demonstrating some of the basic functionality of the Geodise XML Toolbox.

Compute Toolbox

This version of the Geodise Compute Toolbox for Jython provides Globus GT2 client functionality to the Python scripting language. The Geodise Compute Toolbox for Jython uses the Java CoG v1.2 (http://www.globus.org/cog/)[1].

These installation instructions will describe how to configure the Java CoG.

Setting up the Java CoG

This creates a *cog.properties* file containing the default settings for the Java CoG, in a *.globus* directory in the user's home directory. The user should have a valid X509 certificate from a Certificate Authority (CA) acceptable to the managers of the Globus resources that the user wishes to access. To access Globus resources the user should submit the subject line of their certificate to the resource managers to allow them to map it to a user account.

a) Create a *.globus* directory in the user's home directory. Note that in Windows this is best done at the command line with the command 'mkdir .globus'.

b) Copy the CA certificates for the CAs that you wish to trust into the *.globus* directory. These must include the CA certificates for the CA which signed your user certificate, as well as the CAs for any Globus resources that you wish to access. The CA certificate for the UK eScience CA (*01621954.0*) is included with the Geodise Compute Toolbox distribution.

c) Copy the example *cog.properties* file distributed with the Geodise Compute Toolbox into the *.globus* directory, and edit the values of the properties 'usercert', 'userkey', 'proxy' and 'cacert' to the correct values. The example *cog.properties* file contains example configurations for Windows and Unix platforms.

The properties ‘usercert’ and ‘userkey’ refer to locations of the PEM encoded user certificate and corresponding private key. The file ‘cacert’ contains the certificate of the CA which signed the user’s X.509 certificate (in PEM

[1] This product includes software developed by and/or derived from the Globus project (http://www.globus.org/).

format). Where ‘proxy’ will be the location of the user’s proxy certificate once it has been generated by `gd_createproxy`. The properties ‘proxy.strength’ and ‘proxy.lifetime’ contain default settings for the cryptographic strength and lifetime of the proxy certificate. Note that the file separator on a Windows PC must be defined with double backslashes, “\\”.

d) OPTIONAL: For performance the user may wish to change the random seed algorithm, see the section II.3 of the Java CoG FAQ

Test Installation

To test the Compute Toolbox run the Jython startup script, and enter the following command at the Jython command prompt:

```
>>> import demo_compute
```

A short script will be run demonstrating some of the basic functionality of the Geodise Compute Toolbox.

Database Toolbox

The Geodise Database Toolbox for Jython provides client side functions for archiving, querying, retrieving, grouping, and sharing data in an archive. Metadata and Jython variables are stored in an Oracle database accessed though Web services and files are stored on a Globus enabled server.

Requirements

1) The XML Toolbox and Compute Toolbox must be installed and configured before the Database Toolbox can be used.

2) Add certificate subject to database.
The default database for this distribution is hosted by the UK National Grid Service (NGS) on the CCLRC-RAL Data Cluster.

a) Read the terms of use for the UK National Grid Service:
http://www.ngs.ac.uk/NGS-tacu.shtml

b) Join the NGS-ANNOUNCE mailing list to keep informed of any scheduled downtime or other issues related to the NGS RAL Oracle database.
If you have registered with the NGS you should already receive these emails. Otherwise you can join the list at:
http://www.jiscmail.ac.uk/cgi-bin/webadmin?SUBED1=ngs-announce&A=1

c) Contact the database administrator (j.l.wason@soton.ac.uk for the NGS RAL Geodise database) to register your certificate subject so that you are authorised to store and access data in the repository. You can find this in your usercert.pem file, or by viewing the result string returned by calling gd_certinfo from the Compute Toolbox. Here is an example:

/C=UK/O=eScience/OU=Southampton/L=SeSC/CN=joe bloggs

3) Get an account on the file store host.
You need an account on the host you will be storing files on. This can be any Globus enabled server, e.g. grid-data.rl.ac.uk or pacifica.iridis.soton.ac.uk. The administrator will need to add your certificate subject and username to the gridmap file on that machine.

Configuration

Setup

To configure the Database Toolbox run the Jython startup script, and enter the following commands at the Jython command prompt:

```
>>> from gddatabase import *
>>> gd_dbsetup()
```

`gd_dbsetup()` creates a *.geodise* directory in the user's home directory if one does not exist then copies the necessary configuration files into it. Example locations for the *.geodise* directory are:

Windows: *C:\Documents and Settings\your_username\.geodise*
Linux: *$HOME/.geodise*

It will then copy the contents of *<GEODISELABPY>/.geodise* into your *<home_dir>/.geodise* directory.

File store

`gd_dbsetup` will prompt you for the following configuration properties which determine the file store host:

hostname - a Globus enabled server.
hostdir - a directory on that server where files can be stored.

`gd_dbsetup` saves these properties in *<home_dir>/.geodise/ClientConfig.xml*

e.g.

```
<fileserver>
      <hostname>grid-data.rl.ac.uk</hostname>
      <hostdir>/home/<ngs-username>/geodise-data</hostdir>
</fileserver>
```

Test Installation

To test the Database Toolbox run the Jython startup script, and enter the following command at the Jython command prompt:

```
>>> import demo_db
```

A short script will be run demonstrating some of the basic functionality of the Geodise Database Toolbox.

Advanced configuration

A remote database is used to store Jython variables and metadata, and all database interaction is done through web services. The location of these services can also be configured in ClientConfig.xml under the `<webservices>` tag, although it is unlikely you will want to change the default settings for these.

```
<webservices>
   <metadataWS>https://portal.e-science.soton.ac.uk/
GeodiseDB_1_0_NGS/services/MetadataService</metadataWS>
 ...
</webservices>
```

Troubleshooting

(a) Security Exception when using Java 1.4.2 (versions 1.4.2_05 and above)
Java 1.4.2 (versions 1.4.2_05 and above) contains an old version of Xalan which causes an exception in the Apache XML Security software used by the Database Toolbox. Copy xalan.jar from <INSTALL_DIR>/lib to <JAVA_HOME>/jre/lib/endorsed to correct this. Create the endorsed directory in <JAVA_HOME>/jre/lib if it does not exist.

(b) Why does my query cause an 'OutOfMemoryError'?
This error can happen when there are a large number (thousands) of results returned from a query. If the result message from the database service is too large then the Java client code (which sits behind the Jython functions) runs out of memory when trying to parse it.

Here are two simple ways to solve this problem:

i) Reduce the size of your query results (see the next section).

ii) Decrease the value of <query_results><chunk_bytesize> in the <home_dir>/.geodise/ClientConfig.xml file. The database toolbox parses large

query results a 'chunk' at a time, and the default chunk size is 2000000 (about 2MB). Decreasing this number means your machine deals with the query results in smaller chunks and is less likely to run out of memory.

(c) How can I reduce the size of my query results?
Reducing the query result size can speed up the query execution time. There are two general approaches which can be used together:

i) Be more selective by making the query search conditions more specific. For example,

```
>> gd_query('standard.userID = me & standard.archiveDate
>= 2004-10-19 & params < 5')
```

is more selective than

```
>> gd_query('standard.userID = me')
```

and will reduce the size of the query results.

ii) Reduce the number of fields returned in each result structure. The 3rd argument to gd_query can be set to a comma separated list of fields you want returned. The default, '*', will return all the fields available. By only requesting a subset of these you reduce the size of each result structure. The following example only returns the unique identifier(standard.ID) and params field for each matching result.

```
>> q = 'standard.archiveDate >= 2004-10-19 &
standard.archiveDate <= 2004-10-21'
>> gd_query (q, 'file', 'standard.ID, params')
```

Combining approaches i and ii together will reduce the query result size significantly.