



**DATAMINING**

**grid**

**Installation Manual for  
Grid Administrators and  
Application Developers**

**Version 1.0β**

# DATA MINING TOOLS AND SERVICES FOR GRID COMPUTING ENVIRONMENTS

## Installation Manual for Grid Administrators and Application Developers

### **Authors**

Vlado Stankovski, University of Ljubljana, Slovenia

Dennis Wegener, Fraunhofer AiS, Germany

Valentin Kravtsov, Technion Institute of Technology, Israel

Martin Swain, University of Ulster, United Kingdom

## Revision history

Deliverable administration and summary		
<b>Project acronym:</b> DataMiningGrid	<b>ID:</b> IST-2004-004475	
<b>Document identifier:</b>	DataMiningGrid-Manual-ver. 1.0β	
<b>Leading partner:</b> All Partners		
<b>Report version:</b> 10		
<b>Report preparation date:</b> 03.03.2007		
<b>Classification:</b> Public		
<b>Nature:</b> Supporting document describing the DataMiningGrid software ver. 1.0 beta		
<b>Author(s) and contributors:</b> Vlado Stankovski, Dennis Wegener, Valentin Kravtsov, Martin Swain		
<b>Status:</b>	X	Released

The DataMiningGrid © Consortium has addressed all comments received, making changes as necessary. Changes to this document shall be detailed in the change log table below.

Date	Edited by	Status	Changes made
3.3.2007	All Partners	Released	Released

Notice that other documents may supersede this document. A list of latest public DataMiningGrid deliverables can be found at the DataMiningGrid webpage at [www.DataMiningGrid.org/dissemination](http://www.DataMiningGrid.org/dissemination).

## Copyright

This report is © DataMiningGrid Consortium 2007. The document is donated free to the public in connection to the DataMiningGrid software version 1.0 beta.

## Citation

Vlado Stankovski, Dennis Wegener, Valentin Kravtsov, Martin Swain (2007). Installation Manual for System Administrators and Application Developers. DataMiningGrid Consortium, [www.DataMiningGrid.org](http://www.DataMiningGrid.org)

## Acknowledgements

The work presented in this document has been conducted in the context of the EU Framework Programme VI project IST 2004 004475 DataMiningGrid. DataMiningGrid is a 27-month project that started on September 1st, 2004 and is funded by the European Commission as well as by the industrial partners. Their support is appreciated.

The partners in the project are University of Ulster (UU), Fraunhofer Institute for Autonomous Intelligent Systems (FHG), DaimlerChrysler (DC), Israel Institute of Technology (TECH) and University of Ljubljana (LJU). The content of this document is the result of extensive discussions within the DataMiningGrid© Consortium as a whole.

## Additional information

Public DataMiningGrid reports and other information regarding DataMiningGrid are available through the public Web site:

[www.DataMiningGrid.org](http://www.DataMiningGrid.org).

## Summary

This Installation Manual covers the needs of grid administrators and developers of grid-based data mining applications. It contains an overview and installation instructions for the DataMiningGrid © 2007 software version 1.0 beta, which was developed jointly by the DataMiningGrid Partner organizations and is available under the Apache License V2.

The Installation Manual provides information to grid administrators in order to:

- Set up new DataMiningGrid machines (grid security infrastructure, certificates; Condor and Fork adaptors; WS-GRAM, MDS4, GridFTP and OGSA-DAI services); and
- Join the existing DataMiningGrid production environment or develop proprietary test beds and environments;
- Avoid error malfunctions, failures and other reported problems.

Developers of grid-based applications, who would like to further extend the functionality of their systems may benefit from an overview of the DataMiningGrid API.

By using the generic Data Mining Application Enabler many applications were already grid-enabled. An overview of this application is given at the end of the document. Workflow examples from our demonstrator applications are also provided.

This document also contains a short an end-user installation guide for all client side components and set up of grid security infrastructure.

The **version 1.0 beta** of the software is downloadable at:

<http://www.sourceforge.net/projects/datamininggrid>

CVS: <http://datamininggrid.cvs.sourceforge.net/datamininggrid/>

**IMPORTANT NOTICE:** The DataMiningGrid Consortium does not provide any technical support or maintenance for external users. Any support issues should be forwarded to the **DataMiningGrid-community mailing list**.

**You are welcome to join at:**

<https://lists.sourceforge.net/lists/listinfo/datamininggrid-community>

## Table of Contents

Summary.....	5
Table of Contents.....	6
1 Introduction .....	9
2 Development and Administration of a DataMiningGrid Production Environment .....	10
2.1 Setup of execution machines (Condor).....	10
2.1.1 Downloading and installing Condor .....	11
2.1.2 Condor Enhanced GT4 adaptors .....	11
2.2 Setup of GRAMs (Globus Toolkit 4) .....	11
2.2.1 Downloading Globus Toolkit 4 and installation .....	11
2.2.2 Customizing the GT4 setup.....	11
2.2.3 Configuring the topology of the DataMiningGrid production environment in MDS4 .....	13
2.3 Setup of head.....	15
2.3.1 DataMiningGrid Information Service.....	15
2.3.2 DataMiningGrid Resource Broker .....	17
2.4 DataMiningGrid security infrastructure .....	17
2.4.1 Introduction .....	17
2.4.2 Identification and Authentication .....	19
2.4.3 Obtaining host/users certificates .....	21
2.4.4 Deploying certificates on Unix/Windows/other machines .....	24
2.4.5 Generating grid-proxies .....	24
2.5 Errors, malfunctions, failures and other reported problems .....	24
2.5.1 Firewall issues .....	24

- 2.5.2 Well-known domain name ..... 24
- 3 DataMiningGrid API: client-side components ..... 24
  - 3.1 DataMiningGrid projects..... 24
  - 3.2 Downloading the projects..... 24
  - 3.3 Using the projects..... 24
  - 3.4 Java Doc..... 24
- 4 Installation of end-users software: Triana and the DataMiningGrid Units ... 24
  - 4.1 Downloading and installing Triana ..... 24
  - 4.2 Downloading and installing the DataMiningGrid units..... 24
  - 4.3 Security infrastructure ..... 24
- 5 Grid-enabling Existing Data Mining Applications..... 24
  - 5.1 Software Installation ..... 24
    - 5.1.1 Downloading and installing Apache Tomcat..... 24
    - 5.1.2 Downloading and deploying the DataMiningGrid Web applications 24
    - 5.1.3 Setting-up security..... 24
  - 5.2 Application as executable ..... 24
  - 5.3 Data Mining Application Description Schema ..... 24
    - 5.3.1 Application Description ..... 24
    - 5.3.2 Provenance ..... 24
  - 5.4 Inclusion of an application in the grid environment ..... 24
- 6 Example: Construction of a Simple Grid-enabled 'Hello World' Application . 24
  - 6.1 Building a simple application..... 24
  - 6.2 Description of a simple application..... 24
  - 6.3 Inclusion of a simple application into grid test bed ..... 24

7	Demonstrator-specific Software .....	24
7.1	Grid-enabled proprietary or open-source applications .....	24
7.2	DataMiningGrid Workflows.....	24
8	Conclusions and Future Work.....	24
	Appendix .....	24
	Example application description Hello World application.....	24

## 1 Introduction

The Data Mining Tools and Services for Grid Computing Environments (DataMiningGrid) Consortium is developing tools and services for deploying data mining applications on the grid. To demonstrate the developed technology, the project implemented a DataMiningGrid production environment (University of Ljubljana, University of Ulster and Fraunhofer AiS), which is now open for participation of external entities and a range of convincing demonstrator applications in e-science and e-business.

The developed technology is based on existing Globus Toolkit 4 and OGSA-DAI WSRF-compliant technology, GridBus resource broker and Triana Workflow Editor and Manager.

Here is a summary of the technology components developed by the DataMiningGrid project:

- Client-side software components for the Triana Workflow Editor and Manager, which facilitate the composition, execution and management of complex data mining workflows in grid environments;
- Data Mining Application Enabler ((a) a Web application and (b) a Triana unit), which grid-enables existing data mining programs;
- Grid-enabled data access and integration services that allow the users to identify (locate), access, integrate and interface distributed data sources and grid-enabled data mining programs in a flexible way. Please note that services related to access of relational databases are based on OGSA-DAI technology and have to be set up for any specific use-case; and
- A resource broker and information services which perform the execution of data mining programs in WSRF-compliant grid environments.

More information about the system can be found in:

- the End-users Manual;
- the paper: Grid-enabling data mining applications with DataMiningGrid: An architectural perspective, V. Stankovski, M. Swain, Kravtsov, T. Niessen, Dennis Wegener and W. Dubitzky, recently submitted to the Distributed Computing journal (review underway);
- DataMiningGrid Digital Library and Website at [www.datamininggrid.org](http://www.datamininggrid.org)

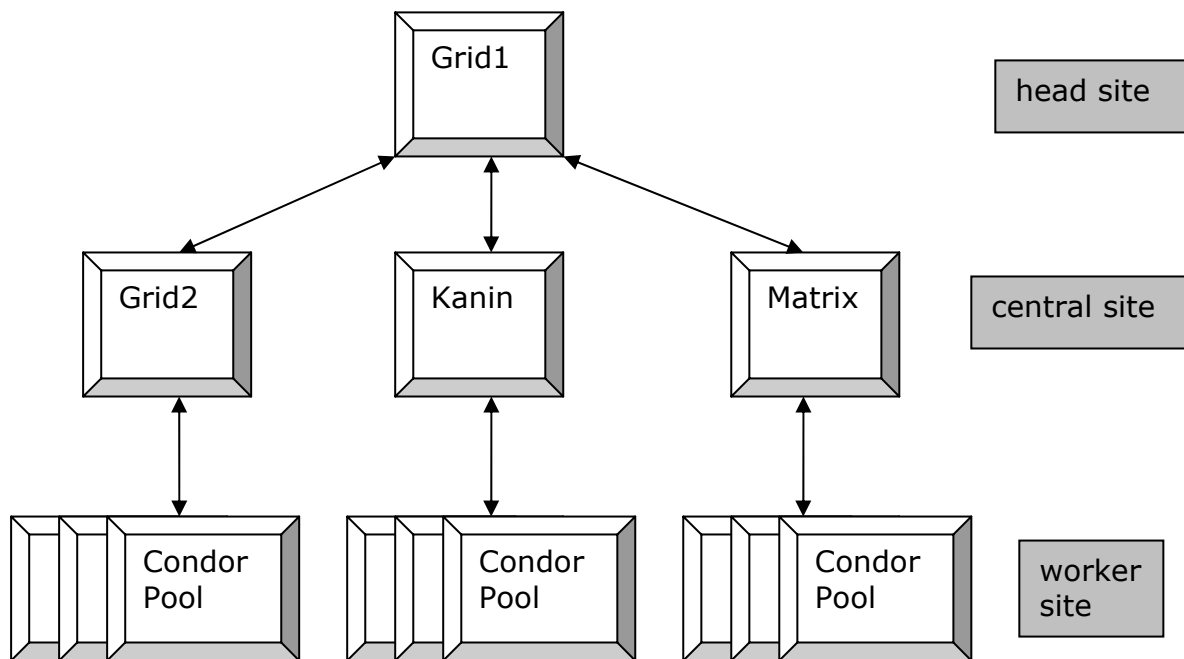
Please forward any support issues to the **DataMiningGrid-community** mailing list. You are welcome to join at:

<https://lists.sourceforge.net/lists/listinfo/datamininggrid-community>

This document covers the needs of grid administrators and enthusiastic developers of grid-based data mining applications. By following the manual you should be able to build your own DataMiningGrid environment or join our existing environment.

## 2 Development and Administration of a DataMiningGrid Production Environment

The setup of a production environment (also called a test bed) can be seen as an integration of the different resources (data, programs, storage and computers) into a Virtual Organization's grid. A DataMiningGrid production environment will usually be made of one 'head' site, which will run unique services (e.g. Resource Broker service, central MDS4 service, Information Integrator service), several 'central' sites (which will run GRAM services and manage the aggregated resources) and many worker machines, usually orchestrated by a local scheduler, e.g. Condor. See Fig. 1 below. The DataMiningGrid system is flexible and therefore allows also for other configurations.



**Figure 1: DataMiningGrid test bed topology showing the RB site, GRAM sites and worker machines.**

The following sections describe the setup of the Condor execution machines (Section 2.1), central site GRAMs (Section 2.2) and head site GRAM (Section 2.3).

### 2.1 Setup of execution machines (Condor)

The following sections describe the setup of the Condor execution machines.

### **2.1.1 Downloading and installing Condor**

Condor is a very powerful local scheduler, which orchestrates the jobs execution in a certain pool. The installation and configuration instructions can be found on the condor's Web site: <http://www.condorproject.org>

### **2.1.2 Condor Enhanced GT4 adaptors**

The Grid Resource Allocation and Management Service (WS-GRAM), which is responsible for submitting, monitoring, and cancelling jobs, manages the execution of applications (i.e. jobs) on particular computational resources through its adapter mechanism. These adapters may either execute jobs on the local machine, where GT is installed, using the C-like 'fork' command, or pass jobs to a local task scheduling system, which on his parts schedules them on all machines it controls. In our system, we use Condor as a local scheduler for managing clusters. However, the original GT4-Condor adapter lacks the capabilities of transferring complete directory structures and executing Java applications. The current Condor implementation (ver. 6.7) in general, and consequently the standard GT4 Condor adapter, restrict data movement to copying files only. While this problem has to be addressed by the Condor development team, we work around it by compressing the recursive directory structures into a single archive (i.e. single file), moving the archive to the execution machine and extracting its content before the actual execution of the data mining application starts. For executing Java applications, we extended the original GT4-Condor adaptor to handle parameters regarding the Java virtual machine and the class path. The original GT4 Fork adapter also lacked the ability to execute Java applications. Its modifications are very similar to the changes we made to the standard GT4-Condor adapter. See the following Section 2.2 for the customizing instructions.

## **2.2 Setup of GRAMs (Globus Toolkit 4)**

The following sections describe the setup of the central site GRAM machines.

### **2.2.1 Downloading Globus Toolkit 4 and installation**

Installation and configuration instructions can be found at the Globus Toolkit quickstart guide:

[www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html](http://www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html)

### **2.2.2 Customizing the GT4 setup**

The standard setup of the GRAMs causes some problems, which make it necessary to modify the default GRAM/Condor behaviour.

## Problems

The following problems make it necessary to modify the default GRAM/condor behaviour:

1. GRAM is not capable of executing Java jobs.
2. Condor is not capable to copy a complete directory structure to the execution machine (its file transfer mechanism can transfer single files only – recursive copy is not supported).
3. Fork execution does not support execution of java jobs

## Solution

The GRAM condor adapter has to be changed as follows in order to handle those problems:

1. If input is a directory, it is zipped into one file, which is transferred to the execution machine.
2. If the job is java job, it is wrapped as condor java job and executed in java universe.
3. The executable which is actually executed on target machine is not the original executable, but our script/java class which performs 2 parts:
  - a. unzip the file,
  - b. execute the original executable

## Setup instructions

For implementing the solution it is necessary to add few additional files which can be downloaded at SourceForge. In order to join the test bed, each GRAM must be configured as follows:

1. Please add the following files to you GT4 directory and add 'executable' permissions to it (`chmod a+x filenames`):
  - a. execution.LINUX.bat
  - b. execution.WINNT51.bat
  - c. unzip.exe
  - d. java.sh
2. Make sure that the Linux execution has Unix script representation:
  - a. Run: `dos2unix execution.LINUX.bat java.sh`
3. Please add the following files to your GT4 directory (make sure that they are readable by all: `chmod a+r filenames`):
  - a. RunJavaCode.class
  - b. RunJavaCode\$ClassPathHacker.class
  - c. RunJavaCode\$Unzip.class
  - d. RunJavaCode.java (optional)
4. Please backup your old condor.pm (file responsible for submission to condor) and put the new one instead of it:

- a. mv  
\$GLOBUS\_LOCATION/lib/perl/Globus/GRAM/JobManager/condor.pm  
\$GLOBUS\_LOCATION  
/lib/perl/Globus/GRAM/JobManager/condor.pm.original
  - b. put the attached condor.pm file into:  
\$GLOBUS\_LOCATION/lib/perl/Globus/GRAM/JobManager/
  - c. Run: dos2unix  
\$GLOBUS\_LOCATION/lib/perl/Globus/GRAM/JobManager/condor.pm
5. Please backup your old fork.pm (file responsible for submission to fork) and put the new one instead of it:
- a. mv  
\$GLOBUS\_LOCATION/lib/perl/Globus/GRAM/JobManager/fork.pm  
\$GLOBUS\_LOCATION  
/lib/perl/Globus/GRAM/JobManager/fork.pm.original
  - b. put the attached fork.pm file into:  
\$GLOBUS\_LOCATION/lib/perl/Globus/GRAM/JobManager/
  - c. Run: dos2unix  
\$GLOBUS\_LOCATION/lib/perl/Globus/GRAM/JobManager/fork.pm

**VERY IMPORTANT NOTICE:**

In the new condor.pm – please make sure it points to the correct condor\_submit and condor\_rm executables.

```
$condor_submit = `correct_path_to_condor_installation/condor_submit`
```

```
$condor_rm = `correct_path_to_condor_installation/condor_rm`
```

For the security setup please see Section 2.4.

### **2.2.3 Configuring the topology of the DataMiningGrid production environment in MDS4**

The Monitoring and Discovery System (MDS) is the Globus Toolkit's information services component. It provides information about the status of grid resources. Major components include the Index service, Trigger service, and Aggregator service. MDS facilitates the discovery and characterization of resources, and monitors services and computations. Using a sample service, we will also demonstrate how to set up, configure, and use MDS to register and query for a service.

#### ***The Community Index Service setting***

In order for the system to be able to propagate information about registered resources throughout the test bed any local IndexService needs to share his information with index services deployed on other machines. For this reason the local host upstreams the information contained in his IndexService to other

service(s) in the system. To set the upstream value, which determines in which direction the information should go, use a text editor to edit the following file:

```
$GLOBUS_LOCATION/etc/globus_wsrfd_index/hierarchy.xml
```

Add the bold text below:

```

<config>

<!--
<upstream> elements specify remote index services that the local
index will be registered to.
Set an upstream entry for each VO index that you wish to
participate in.
-->

<upstream>https://GT4-
source: 8443/wsrfd/services/DefaultIndexService</upstream>

</config>
  
```

### *Query the IndexService*

To verify registration, use the GT4 command-line tools to query the IndexService as follows for a test service, which is deployed in one container only. Make sure to put the entire command on one line:

```

$GLOBUS_LOCATION/bin/wsrfd-query -s
  http://GT4-source: 8443/wsrfd/services/DefaultIndexService '/'
... Lots of output ...
<ns1: MemberServiceEPR>
  <ns8: Address
xml ns: ns8=' http://schemas.xmlsoap.org/ws/2004/03/addressing' >
    http://129.7.248.40:8080/wsrfd/services/AuctionService
  </ns8: Address>
  
```

So for the DataMiningGrid test bed you have to set an upstream to the head GRAM machine (see also Section 2.3).

## **2.3 Setup of head**

The head GRAM needs a similar GT4 setup as the central cite GRAMs except the customizations (Section 2.2.2) and the topology settings (Section 2.2.3). The head machine is not used for execution so there is no need for changing the settings according to Section 2.2.2. The other GT4 machines forward there information about the services according to the setup of Section 2.2.3 so no additional settings of the head machine are necessary.

For the head GRAM you have to set up some additional services as described in the following sections.

For the security setup please see Section 2.4.

### **2.3.1 DataMiningGrid Information Service**

The InformationIntegrator Service provides information about the grid-enabled applications. Through the InformationIntegrator service you are able to access the application descriptions of the available applications. This registry is queried by the client side components like the Triana Unit Application Explorer or the Generic Website.

The service contains the script `concatDescription.sh`. The service executes this script in order to concatenate the Application Descriptions to one xml document.

#### **2.3.1.1 Downloading the service**

The service (*org\_dataminiggrid\_informationIntegrator.gar*) can be downloaded at SourceForge. Additional to the gar file you need a script called *concatDescriptions.sh*.

#### **2.3.1.2 Deploying the service**

The InformationIntegrator service can be deployed by using the following command:

```
globus-deploy-gar org_dataminiggrid_informationIntegrator.gar
```

This will create a few new folders under `$GLOBUS_LOCATION` and copy files to the dedicated locations.

#### **2.3.1.3 Set-up instructions**

##### **Customize Index Service**

The MDS Index service has to be customized as follows:

In the file `$GLOBUS_LOCATION/etc/globus_wsrf_mds_index/jndi-config.xml` the following entry has to be included (or just customized):

```
<parameter>
  <name>executableMappings</name>
  <value>concatDescription=concatDescription.sh</value>
</parameter>
```

If there is more than one entry in the `<value>` tag it has to look like this:

```
<value>xyz=xyz.sh, concatDescription=concatDescription.sh</value>
```

### **Customize InformationIntegrator Service**

In the new created folder `$GLOBUS_LOCATION/etc/org_datamininggrid_services_informationIntegrator/` some files have to be customized. Inside the file `informationIntegrator-aggregator-registration.xml` you have to set the IPs to the local machines IP.

The name of the service is set inside the file `jndi-config.xml`.

You have to copy the script `concatDescription.sh` the service executes to the folder `$GLOBUS_LOCATION/libexec/aggrexec`. Make sure that the script is executable by using `chmod 777`.

### **Customize InformationIntegrator Script**

If necessary in the script `concatDescription.sh` you have to customize the path and the name of the xml application description files to find. The original script looks like this:

```
#!/bin/bash
allfiles=`find /tmp/upload -name 'ApplicationDescription.xml'`
echo '<ApplicationDescriptions>'
for fileName in $allfiles
do
  awk 'NR>=2' $fileName
done
echo '</ApplicationDescriptions>'
```

## **2.3.2 DataMiningGrid Resource Broker**

### **2.3.2.1 Downloading the service**

The Resource Broker service (*org\_datamininggrid\_executionsystem.gar*) can be downloaded at SourceForge. Additionally you need a file called *stageinscript*.

### 2.3.2.2 Deploying the service

Resource Broker is deployed as a standard WSRF compliant service into the GT4 or tomcat container. For deploying the RB to GT4 container, just type:

```
globus-deploy-gar org_datamininggrid_executionsystem.gar
```

### 2.3.2.3 Set-up instructions

Please also make sure that the file named 'stageinscript' is placed into the \$GLOBUS\_LOCATION directory.

## 2.4 DataMiningGrid security infrastructure

### 2.4.1 Introduction

In general, grid utilizes public key or asymmetric cryptography for authentication of users, resources and service. According to the basics of public-key cryptography, each resource on the Grid has a key pair, a public and a private key. The public key is made public while the private key must be kept secret. Encryption is performed using the public key while decryption and digital signature is performed with the private key.

It is important to notice that generating a key pair does not automatically provide you access to the Grid resources. A trusted authority of the Grid, called the Certificate Authority (CA), needs to sign your key pair this way confirming your identity. This signing procedure of the CA is often referred as issuing a certificate. For the DataMiningGrid test bed there shall be at list one trusted CA, namely the DataMiningGrid Certificate Authority.

See also Globus Toolkit 4 quickstart guide:

<http://www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html>

#### 2.4.1.1 DataMiningGrid Certification Authority

DataMiningGrid CA is managed by the University of Ljubljana (contact email: [vlado@stankovski.net](mailto:vlado@stankovski.net)), and accepts certificate requests from external entities, evaluates them and signs them on a case-by-case basis. Universities, companies and physical persons are welcome to apply for certificates or negotiate mutual acceptance of Certificate Authorities.

The DataMiningGrid CA provides X.509 certificates for identification and authentication purposes in the scope of the DataMiningGrid project. The minimum key length for all certificates is 1024 bits. **Validity period is one year.**

#### 2.4.1.2 DataMiningGrid Termination

The DataMiningGrid CA may be terminated without prior notice, although there are no such plans at the moment. At time of termination, DataMiningGrid CA will stop issuing certificates and all DataMiningGrid CA certificates will expire in one year after that.

In the meantime, it is assumed that certificates issued by other CAs will be accepted by some partners forming the DataMiningGrid production environment, which is based on bilateral agreements. A list of trusted CAs shall be defined by the each Partner in the DataMiningGrid project before termination of the DataMiningGrid CA.

#### 2.4.1.3 End Entities

The DataMiningGrid CA issues **certificates for services, hosts and people** from the DataMiningGrid Consortium as well as other entities.

#### 2.4.1.4 Applicability

Person certificates can be used to **authenticate** a person to reach **dedicated sites** that have **agreed to accept certificates from the** DataMiningGrid CA and may require the signing of Globus proxy certificates.

A list of hosts which accept certificates from the DataMiningGrid CA is presented below:

- grid1.dk-grid.ais.fraunhofer.de
- grid2.kd-grid.ais.fraunhofer.de
- kanin.fgg.uni-lj.si
- matrix.scic.ulst.ac.uk

**Both host and service certificates** can be used to identify a unique DN used in the authorization processes and for encryption of communication (GSI/SSL/TLS).

#### 2.4.1.5 User Restrictions

The ownership of a DataMiningGrid CA certificate does not imply automatic access to any kind of computing resources.

#### 2.4.1.6 Subscriber Obligation

Each subscriber requesting a certificate from the DataMiningGrid CA **MUST**:

- Read and adhere to the procedures described in this document;
- Generate a key pair using a trustworthy method;

- Take reasonable precautions to prevent any loss, disclosure or unauthorized use of the private key associated with the certificate, including:
  - For person certificates:
    - Never sharing the private key with other users;
    - Selecting a pass phrase of a minimum recommended 8 characters;
    - Protecting the pass phrase from others;
    - Always using the pass phrase to encrypt the stored private key.
  - For Service/Host certificates:
    - Storing them encrypted whenever possible;
    - They may be kept unencrypted on the host that they represent;
- Provide correct personal information and optionally authorize the publication of the certificate;
- Notify the issuer, i.e. DataMiningGrid CA immediately in case of private key loss or compromise; and
- Use the certificates for the permitted uses only.

## 2.4.2 Identification and Authentication

### 2.4.2.1 Initial Registration

The Subject Name is an X.500 name type, a Distinguished Name. It has one of the following forms:

- **Person:** MUST include the full name of the subject;
- **Host:** MUST include the domain name of the host; and
- **Service:** MUST include the fully qualified domain name of the host, and optionally the named service.

The Subject Name in a certificate MUST have a reasonable association with the authenticated name of the subscriber.

The X.500 Distinguished Name (DN) MUST be unique for each subject name certified by the DataMiningGrid CA.

Certificates MUST apply to unique individuals or resources.

Example DNs for the DataMiningGrid CA, user and host are presented below:

- **DataMiningGrid DN**  
'/O=Grid/OU=DataMiningGridTestbed/OU=DataMiningGridCA-kanin.fgg.uni-lj.si/CN=DataMiningGrid CA'
- **User DN** '/O=Grid/OU=DataMiningGridTestbed/OU=DataMiningGridCA-kanin.fgg.uni-lj.si/OU=fgg.uni-lj.si/CN=Vlado Stankovski'

- **Host DN** '/O=Grid/OU=DataMiningGridTestbed/OU=DataMiningGridCA-kanin.fgg.uni-lj.si/CN=host/kanin.fgg.uni-lj.si'

#### 2.4.2.2 Authentication of Individual Identity

The DataMiningGrid CA verifies the identity of a person by checking:

- A certificate request (or renewal or revocation) **MUST** be sent to **vlado@stankovski.net** with an email subject containing 'Certificate Request' (or 'Certificate Revocation') and originate from a valid email address from a **known person**;
- **Known person** is a person involved directly in the DataMiningGrid project or other persons as verified as such by the issuer;
- A **contact list of known people**, as a flat text file, will be maintained to which only a known person has authorized access;
- **Each known person on the contact list MUST** be characterized by the following attributes:
  - First and Second Name
  - Institution
  - Valid email address
- A request **will be accepted if** the person is known and characterized by all aforementioned attributes.

#### 2.4.2.3 Certificate Revocation

A certificate will be revoked when the information it contains is suspected to be incorrect or compromised. This includes situations where:

- The subscriber's private key is lost or suspected to be compromised;
- The information in the subscriber's certificate is suspected to be inaccurate;
- The subject has failed to comply with the rules in this policy;
- The system to which the certificate has been issued has been retired;
- The subscriber no longer needs the certificate to access a relying parties' resources;
- The subscriber violated his/her obligations.

A certificate revocation can be requested by the holder of the certificate to be revoked or by any other entity presenting proof of knowledge of a circumstance of revocation.

#### 2.4.2.4 Liability issues

The DataMiningGrid CA and its agent issue person certificates according to the practices described in this document to validate identity. **No liability, implicit or explicit, is accepted.**

The DataMiningGrid CA and its agents make **no guarantee** about the security or suitability of a service that is identified by a DataMiningGrid CA certificate. The certification service is run with a reasonable level of security, but it is provided on a **best-effort basis**. It does not warrant its procedures and it will take no responsibility for problems arising from its operation, or for the use made of the certificates it provides.

DataMiningGrid CA **denies** any financial or any other kind of responsibility for damages or impairments resulting from its operation.

DataMiningGrid CA assumes **no financial responsibility** with respect to use or management of any issued certificate, or any services which are running in the DataMiningGrid production environment.

#### 2.4.2.5 Contact Details

The contact person for questions related to this document or the DataMiningGrid CA in general is:

**Vlado Stankovski** <vlado@stankovski.net>

#### 2.4.3 Obtaining host/users certificates

The following Section describes the use of SimpleCA for multiple machines in the DataMiningGrid test bed.

Below the source GT4 machine is the machine with the issuing authority, the target GT4 machine is the one which obtains the host certificate.

On the target machine CA package has to be installed. Copy CA package from source-GT4 to the target-GT4 machine which can be found under /home/globus/.globus/simpleCA//globus\_simple\_ca\_hash\_setup-HASH.tar.gz

Run the following commands as root:

```
$GLOBUS_LOCATION/sbin/gpt-build globus_simple_ca_HASH_setup  
HASH.tar.gz gcc32dbg  
$GLOBUS_LOCATION/sbin/gpt-postinstall
```

Run the following as root (or, if no root privileges are available, add the **-nonroot** option to the command line):

```
$GLOBUS_LOCATION/setup/globus_simple_ca_CA_Hash_setup/setup-gsi -
default
```

### 2.4.3.1 Host certificates

#### Request a host certificate

As root, run:

```
grid-cert-request -host 'hostname'
```

This creates the following files:

- /etc/grid-security/hostkey.pem
- /etc/grid-security/hostcert\_request.pem
- (an empty) /etc/grid-security/hostcert.pem

*Note:* If you are using your own CA, follow their instructions about creating a hostcert (one which has a commonName (CN) of your hostname), then place the cert and key in the /etc/grid-security/ location.

#### Sign the host certificate

1. Send or mail the certificate request (/etc/grid-security/hostcert\_request.pem) to the GT4 machine where issuing authority is located (source-GT4 machine which CA package is being used).
2. Now at the source GT-4 side who has received the certificate request (/etc/grid-security/hostcert\_request.pem) from the target GT4-machine.

As globus, run:

```
grid-ca-sign -in hostcert_request.pem -out hostsigned.pem
```

3. A signed host certificate, named hostsigned.pem, is written to the current directory.
4. When prompted for a passphrase enter the one you specified for the private key of the CA certificate at source-GT4 machine.
5. Rename hostsigned.pem to hostcert.pem
6. send hostcert.pem to target GT4
7. As root at target GT4 machine move the signed host certificate i.e hostcert.pem (which you received from source-GT4) to /etc/grid-security/hostcert.pem.

The certificate should be owned by root and be read-only for other users.

The key should be read-only by root.

### Make the host credentials accessible by the container

At target-GT4, the host key (`/etc/grid-security/hostkey.pem`) is only readable to root. The container (hosting environment) will be running as a non-root user (probably the `globus` user) and in order to have a set of host credentials which are readable by the container, we need to copy the host certificate and key and change the ownership to the container user.

As root, run:

```
root# cd /etc/grid-security
root# cp hostkey.pem containerkey.pem
root# cp hostcert.pem containercert.pem
root# chown globus.globus containerkey.pem containercert.pem
```

At this point the certificates in `/etc/grid-security` should look something like:

```
root# ls -l *.pem
-rw-r--r-- 1 globus globus 1785 Oct 14 14:47 containercert.pem
-r----- 1 globus globus 887 Oct 14 14:47 containerkey.pem
-rw-r--r-- 1 root  root  1785 Oct 14 14:42 hostcert.pem
-r----- 1 root  root   887 Sep 29 09:59 hostkey.pem
```

Start the container by following command:

```
globus-start-container
```

If it runs with out any problem then it shows that your target-GT4 is using your source-GT4 credentials & both can use and register (MDS) each other services. From now on user of both GT4 installations will have the same privileges to use each other services.

### 2.4.3.2 User certificates

Users also must request user certificates, which you will sign using the *globus* user.

#### Request a user certificate

As your normal user account (*not globus*), run:

```
grid-cert-request
```

After you enter a passphrase, this creates

- `~$USER/.globus/usercert.pem` (empty)
- `~$USER/.globus/userkey.pem`
- `~$USER/.globus/usercert_request.pem`

Email the `usercert_request.pem` file to the SimpleCA maintainer (*Source-GT4*).

#### Sign the user certificate

1. As the SimpleCA owner *globus*(*Source-GT4*), run:

```
grid-ca-sign -in usercert_request.pem -out signed.pem
```

2. When prompted for a password enter the one you specified for the private key of the CA certificate.
3. Now send the signed copy (`signed.pem`) back to the user who requested the certificate.
4. As your normal user account (*not globus*), copy the signed user certificate into `>~/globus/` and rename it as `usercert.pem`, thus replacing the empty file.

The certificate should be owned by the user and be read-only for other users.

The key should be read-only by the owner.

#### Add authorization

Add authorizations for users:

Create `/etc/grid-security/grid-mapfile` as root.

You need two pieces of information:

- The subject name of a user
- The account name it should map to.

The syntax is one line per user, with the certificate subject followed by the user account name.

Run **grid-cert-info** to get your subject name, and **whoami** to get the account name:

```
bacon$ grid-cert-info -subject
/O=Grid/OU=GlobusTest/OU=simpleCA-
mayed.mcs.anl.gov/OU=mcs.anl.gov/CN=Charles Bacon
bacon$ whoami
bacon
```

You may add the line by running the following as root:

```
root# $GLOBUS_LOCATION/sbin/grid-mapfile-add-entry -dn \
'/O=Grid/OU=GlobusTest/OU=simpleCA-
mayed.mcs.anl.gov/OU=mcs.anl.gov/CN=Charles Bacon' \
-ln bacon
```

The corresponding line in the `grid-mapfile` should look like:

```
'/O=Grid/OU=GlobusTest/OU=simpleCA-
mayed.mcs.anl.gov/OU=mcs.anl.gov/CN=Charles Bacon' bacon
```

### Verify Basic Security

Now that you have installed a trusted CA, acquired a hostcert and acquired a usercert, you may verify that your security setup is complete (Please see Section 2.4.5)

Important:

*IMPORTANT NOTICE: If the user of source-GT4 wants to use the services of target-GT4 then he/she must have a valid entry in both grid-mapfiles of source-GT4 and target-GT4 machine.*

### 2.4.4 Deploying certificates on Unix/Windows/other machines

End user certificates have to be deployed in the home directory of the user in a subdirectory called '.globus'. The file 8d39a21a.0 (belonging to the CA) has to be located in '.globus/certificates'.

### 2.4.5 Generating grid-proxies

If the user wants to use the services then he/she must have a valid credential. For generating such a proxy certificate, run the following command as your user account:

```
bacon$ grid-proxy-init -verify -debug

User Cert File: /home/bacon/.globus/usercert.pem
User Key File: /home/bacon/.globus/userkey.pem

Trusted CA Cert Dir: /etc/grid-security/certificates

Output File: /tmp/x509up_u506
Your identity: /DC=org/DC=doegrids/OU=People/CN=Charles Bacon 332900
Enter GRID pass phrase for this identity:
Creating proxy ...+++++++
.....+++++++
Done
Proxy Verify OK
Your proxy is valid until: Fri Jan 28 23:13:22 2005
```

There are a few things you can notice from this command. Your usercert and key are located in \$HOME/.globus/. The proxy certificate is created in /tmp/. The 'up' stands for 'user proxy', and the \_u506 will be your UNIX userid. It also prints out your distinguished name (DN), and the proxy is valid for 12 hours.

## 2.5 Errors, malfunctions, failures and other reported problems

### 2.5.1 Firewall issues

Usually the GRAMs which participate in the test bed have a kind of security configuration which can cause transport problems. If there are firewalls running on the GRAMs you maybe have to customize the firewall scripts. These scripts should allow communication between the machines which are participating in at accessing the test bed, e.g. the scripts should include the IP range of all other GRAMs, the timeserver used for synchronizing the system time and the clients which are accessing the test bed.

### 2.5.2 Well-known domain name

If the GRAMs of the test bed do not have a fully qualified domain name you have to edit the client machines host file and add for instance the following line:

```
193.175.164.2 grid1.kd-grid.ais.fraunhofer.de
```

On Linux the hosts file is located at */etc/hosts* and on Windows at */windows/system32/drivers/etc/hosts*

## 3 DataMiningGrid API: client-side components

### 3.1 DataMiningGrid projects

The DataMiningGrid software system was developed as 5 main projects which also can be used as an API:

- dm-g-Commons: The Commons project contains common constants and external libraries
- dm-g-DataServices: The DataServices project contains the clients for the Data Services
- dm-g-ExecutionSystem: The ExecutionSystem project contains the API and clients for the execution and the deployable WSRF-compliant ResourceBroker service
- dm-g-InformationIntegrator: The InformationIntegrator project contains the deployable InformationIntegrator service and the client side components. Additionally, it contains the DM ApplicationDescription Schema and the Application Descriptions belonging to the demonstrator applications.
- dm-g-TrianaToolbox: The TrianaToolbox project contains the Triana Units and the related Triana Types.

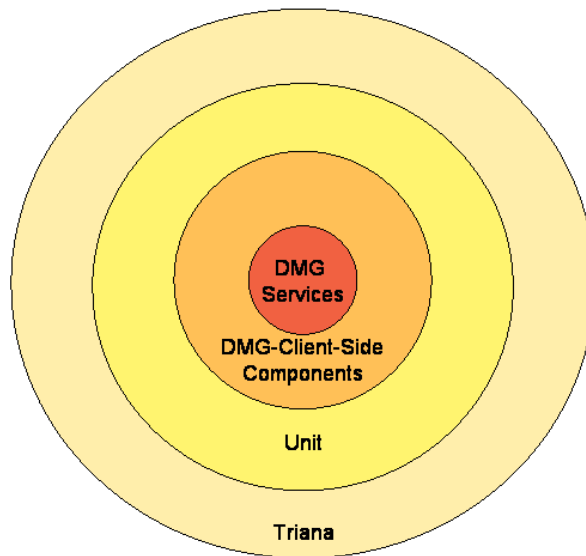
### 3.2 Downloading the projects

The projects can be freely downloaded from SourceForge under the Open Source Apache License V2.

### 3.3 Using the projects

Figure 2 illustrates the encapsulation of the respective client-side components by the Triana units, which in turn call the required DMG and GT4 services. This encapsulation enables application developers to integrate DMG components into their applications without extra labour.

This encapsulation only works for services developed by the Partners, since they have decided to provide client-side components for each service used in this project to relieve users of the burden of discovering and binding to services in the right manner.



**Figure 2: The four different layers for integrating and accessing DMG services**

### 3.4 Java Doc

JavaDoc is currently not provided.

## 4 Installation of end-users software: Triana and the DataMiningGrid Units

### 4.1 Downloading and installing Triana

Triana is a powerful Workflow Editor and Manager. The installation and configuration instructions can be found at the Triana web site:

<http://www.trianacode.org>

### 4.2 Downloading and installing the DataMiningGrid units

The DataMiningGrid Units can be downloaded from SourceForge.

The DataMiningGrid.zip file containing the DataMiningGrid Units has to be unzipped in the home directory of the Triana installation (\$TRIANA\_HOME). After (re-) starting Triana a new folder will appear in the tree-view, which now contains the DataMiningGrid units. The DataMiningGrid units are ordered by their packages.

### 4.3 Security infrastructure

End-users should also obtain and install a certificate (see section 2.4 for details on the security setup). The certificate files (usercert.pem and userkey.pem) are usually installed in a .globus directory in the users' home directory,

e.g. C:\Documents and Settings\Your Name\.globus

It is also necessary to register the DataMiningGrid CA (or any other authority used in your case). This is done by copying the files:

8d39a21a.signing\_policy and 8d39a21a.0 in

C:\Documents and Settings\Your Name\.globus\certificates

At application run-time, the end-users proxy is automatically generated by Triana in a default directory. Please check in the temporary directory, e.g.

C:\Documents and Settings\Your Name\Local Settings\Temp

For an automatically generated proxy file called: "x509up\_u\_ your name".

## **5 Grid-enabling Existing Data Mining Applications**

In this section we discuss the process of writing a description of an arbitrary application/algorithm by utilizing the Data Mining Application Description Schema. We also explain how to include an application in the grid test bed environment.

Section 5.1 describes the required software installation. Section 5.2 characterizes the properties of an application w.r.t. to the grid middleware. It also specifies the required control and execution flow. The Data Mining Application Description Schema provides an easy way to describe an application to be included into the grid environment. Section 5.3 introduces the Data Mining Application Description Schema and gives an overview of the main elements. Section 5.4 explains how to describe an application according to the Data Mining Application Description Schema and the necessary steps to include an application into the grid environment.

### **5.1 Software Installation**

#### **5.1.1 Downloading and installing Apache Tomcat**

To install the DataMiningGrid Web applications you will need a running servlet container like Apache Tomcat. Please refer to <http://tomcat.apache.org/> on how to install a servlet container.

For this version of the DataMiningGrid websites, at least Tomcat Version 5.0 is required.

#### **5.1.2 Downloading and deploying the DataMiningGrid Web applications**

The following Web applications were developed by the DataMiningGrid consortium:

- Data Mining Application Enabler (DMApplicationEnabler.war)
- Generic DataMiningGrid website (generic\_website.war)
- Web digital library (website) (Web\_digital\_library.war)

The DataMiningGrid website consists of a single WAR (Web Application Archive) file dmgrid\_generic\_website.war. This file can be downloaded from SourceForge. In order to deploy the website you have to copy dmgrid\_generic\_website.war into the webapps directory of Tomcat and restart it. The servlet container automatically unpacks the war file into a directory named dmgrid\_generic\_website in the webapps directory.

Alternatively you can use Tomcat Manager to deploy the war file. Please refer to

<http://tomcat.apache.org/tomcat-5.0-doc/manager-howto.html>

or

<http://tomcat.apache.org/tomcat-5.0-doc/appdev/deployment.html>

for detailed instructions.

### **5.1.3 Setting-up security**

The user who runs the Tomcat servlet container has to hold a user certificate as described in Section 2.4. You have to make sure that the credential of this user is valid for the time the Web application is supposed to run.

## **5.2 Application as executable**

A DataMiningGrid application is an application which is executed from a DataMiningGrid client side component (e.g. the workflow editor or a Web application).

Seen from the grid middleware viewpoint, a DataMiningGrid application or algorithm is an executable with associated input data, output data and parameter settings. The executable can be a Java, C, Python or BashShell program.

A data mining application that is to be grid-enabled has to have the following properties. One should be able to invoke the application from the command-line i.e., it has to be implemented to run without a graphical user interface. It should be possible to control input data, output data, and parameter settings via the command-line, using the format of a flag followed by the associated value ('flag <space> value', e.g. '-number 100').

Additionally an application may have some system requirements like minimum free disk space, minimum memory or required operating system which have to be complied in order to run.

To describe all these properties the partners developed the Data Mining Application Description Schema which is introduced in the following Section.

## **5.3 Data Mining Application Description Schema**

The Data Mining Application Description Schema developed by the partners is an XML Schema for describing data mining applications in order to make them operable in the grid environment. The schema provides an easy way to describe the application and all its input data, output data, parameter settings etc. It consists of a generic part and DM specific part.

Additionally there is a part of the schema which is used for describing provenance information. The following Sections describe the elements of the xml schema. For the detailed schema see Appendix.

### **5.3.1 Application Description**

The Data Mining Application Description Schema is used to describe any data mining algorithm. It consists of (A) a 'common' part, which can be used also for describing any application whatsoever (i.e. not necessarily data mining) and (B) data mining specific part with some additional information relating to data mining applications.

The schema which describes an application consists of several elements which describe all different properties of the application. The main element of the schema is *application*. It contains the elements *generalInformation* (A), *applicationType* (B), *execution* (A) and *applicationInformation* (A).

First we give the description of the common part of the schema:

#### (A) Common part of the schema

1) The element *generalInformation* contains general information about the application, it is collected for the application identification purpose and is used to find the application inside the grid registry. In detail it contains the following information:

- Longer textual description of the application;
- Comprehensive textual description of the algorithm(s), the values it takes, parameters, input and output data;
- Additional comments;
- Global unique ID (this is set automatically when using the Data Mining Application Enabler);
- Vendor - or producer of the Open Source application;
- Version of the software, e.g. 3.2;
- Version of the source code, e.g. 2.4.15;
- Version of the build, e.g. 5423;
- Short description of the software; and
- Date of uploading to the grid (this is set automatically when using the Data Mining Application Enabler)

2) The element *execution* contains information about the applications execution. This is in detail the following:

- Type of execution: Java, C, Bash Shell, Python;
- Required libraries along with the executable, e.g. in the ecological modelling application some libraries are used which are not available with

OS distributions (they have to be uploaded by the application developer, when he is using the Data Mining Application Enabler application);

- Other execution specific information, e.g. if the executable is going to run in a sub-directory or not and which sub-directory; and
- Commands and arguments to start the interpreter, e.g. 'python'.

3) The element *applicationInformation* contains the elements *options*, *dataInputs*, *dataOutputs*, *hostEnvironmentVariables*, *requirements*, *parameterLoops* and *parameterLists*. These elements store the following information under the tags:

3.1) The element *options* describes the options of the application which are defined by the DataMiningGrid application developer. The developer describes one-by-one each option. This element contains the flag and the parameter value(s) which are compiled into the command line call. In order to make it for the user easily possible to specify the value the option has a data type. Additional information, like status of the parameter (optional/hidden), default values and/or possible values can be supplied. If the parameter is if the parameter is hidden the user cannot later specify a value at the GUI and if it is set to optional the user does not have to set a value.

In detail the element contains the following information:

- Name of option to be displayed to the user on the client;
- Data type of this option;
- Flag to be printed to command line at start-up by the system;
- Optional status, defining whether the option is obligatory or not, e.g. parameterA 0 is not necessary option, meaning the algorithm can also run without a value for this option);
- Hidden status, defining if the option is hidden from the user. If set to true, defaultValues MUST be set (by the developer) and the algorithm can be run, but, the end-user will have no chance to see or change this option of the application; and
- Two short sentences at most for a tooltip specified by the application developer.

3.2) The elements *dataInputs* and *dataOutputs* are used to describe the application's input and output data, i.e. data types (file or directory), other data, transfer protocols possible for the particular data, physical location of data, other descriptors (e.g. is the data input obligatory or not) etc.

If the algorithm does not know where the data is you can use *appendToCommandLine* to append a flag-value pair to the command line to set the input/output. *stageIn* specifies if the input data will be moved to the GRAM, *stageOut* specifies if the output data will be moved from the GRAM. You also can set the input optional. The type of the input/output is either a file or a directory. The value of the output consists of the location of the output file or directory.

In detail the elements contain the following information:

- Files containing input data for the application. This could be also a list of files, e.g. *modellingDataSet*, *simulationSet1*, *simulationSet2*, ...;
- Remote File Directory Type (protocol, port, host, localPath, etc.);
- *ioType*, *label*, *flag*, *toolTip*, optional input/output specification;
- Information on whether or not to stage in (physically copy) this input to the execution machine;
- Whether to pass the input/output to the application via command line;
- Protocols for file transfers allowed in the DataMiningGrid system. This could be enhanced in the future with protocols such as ftp, http, https. Currently we have only gsiftp and http; and
- Input/output types possible are: Datafile, Directory (e.g. in text-mining demonstrators), Parameterfile (e.g. in ecological demonstrator).

3.3) The element *hostEnvironmentVariables* contains environment variables to be set at the execution machine before execution (variable – value pairs).

3.4) The element *requirements* contains the system requirements for the resource broker. This can be the minimum memory, the minimum disk space, the GRAM job manager type (Fork or Condor), the IP of the execution machine, the operating system or the architecture. This all is information about the application that is used by the Resource Broker to find a best possible match with computational resources in the test bed.

In detail, the element contains the following information:

- Minimum memory needed to run the application, minimal disk space;
- GRAM requirements. No entry means that the application may run as 'fork' and as 'condor' job. This could be further extended to other local schedulers;
- Operating system needed for execution (it can be ALL, Windows or Linux);
- Processor architecture (executables have been compiled only for specific architecture, e.g. intel, pc, itanium, etc.); and
- List of IPs of machines with GT4 installations the application is allowed to run on, e.g. in some demonstrators we can not run the algorithms on any machine so the users can specify the exact pool of machines where to run the applications.

3.5) The elements *parameterLists* and *parameterLoops* are used to define loops and lists - this part of the schema is used in the ParameterControl unit. The list element is used for a sweep over a list of values. It need not be provided in the description of an application explicitly but it is set automatically if an iterated execution of the application for a list of different values is required. The loop element is used for a sweep over a user-defined interval with a fixed step size.

This element need not be provided in the description of an application but it is set automatically if the user wants to perform a sweep.

Basically, a structure is provided for creating multiple options, both for numeric and alphanumeric cases.

E.g.:

```
algorithm.exe -learningParameterA 3  
  
algorithm.exe -learningParameterA 2  
  
algorithm.exe -learningParameterA 1
```

This structure can store the list {1,2,3} or in case of alphanumeric values e.g. {yellow, green, blue}. Then, at execution stage, the Resource Broker uses these lists to generate as many as necessary jobs for execution.

More detailed:

- Structure for creating multiple values for a single option in a loop-like manner. This applies to numeric options only. The parameter name (e.g. \$X) may be referenced by option values (e.g. -number \$X), input, and output file/dir names. The loop is evaluated automatically by the Execution sub-system and each iteration results in a separate execution of the application. It is filled by the end-user or automatically by the system. E.g. {1,2,3}
- Structure for defining multiple values for a single option as a list. The parameter name (e.g. \$X) may be referenced by option values (e.g. -name \$X), input, and output file/dir names. It applies to any option. The list is evaluated automatically by the Execution sub-system and each value results in a separate execution of the application. The structure can be filled-in by the end-user or the system! e.g. {yellow, green, blue}

#### (B) Part specific to data mining:

This is to enable faster, easier discovery of relevant algorithms. (In the future, there could be 100s, even 1000s of data mining applications running in the DataMiningGrid test bed or in a grid environment based on the DataMiningGrid system).

The element *applicationType* contains this information, which is in detail:

- Application domain / textual description;
- Name of the (large-scale) application this algorithm belongs to (e.g. Weka);
- Name of the atomic application/algorithm. It should be unique within an application group, e.g. J48;
- The type of problem the algorithm solves (e.g. regression);

- The technique used by the actual algorithm (e.g. regression tree, classification, clustering, attribute importance, association, other)
- CRISP-DM phase: See also [www-ai.cs.uni-dortmund.de/kdnet/kd\\_standards\\_final.ppt?self=\\$docwtpjt&part=data](http://www-ai.cs.uni-dortmund.de/kdnet/kd_standards_final.ppt?self=$docwtpjt&part=data);
- Data exploration techniques (e.g. univariate data plots), data quality issues, attribute interactions, etc.;
- Data Understanding, All pre-processing algorithms, e.g. data integration, selection, sampling, transformation, cleaning, etc.;
- All DM algorithms, e.g. classifiers, differential equation solvers, etc.;
- All validating techniques; and
- E.g. monitoring and evaluation effectiveness after deployment.

### 5.3.2 Provenance

The ProvenanceType contains information about the job execution. It has the following elements:

- *applicationDescription*: This element contains the whole information about algorithm execution. It is used in order to preserve the information about which algorithm and which parameters were used in order to achieve the results. This element contains all information described in Section 3.2.1.
- *submissionTime*: This element contains the date and time of the submission of all the jobs to Resource Broker service.
- *completionTime*: This element contains the date and time of the completion of the last job (out of all the jobs submitted to Resource Broker service).
- *schedulerStatus*: This element contains the status of the execution (success/failure).
- *resultsLocation*: This element contains the URI to the results of the job(s) execution.
- *jobsStatus*: This element contains some additional information regarding each one of the jobs. For each job it contains the start & end time of the actual execution, job's status (success/failure), error description (in case of the failed job), and the data regarding variables instantiation.

The provenance information is later shown inside a provenance Triana unit. Inclusion of an application in the grid environment

## 5.4 Inclusion of an application in the grid environment

In order to include an application in the grid environment it is necessary to create an application description according to the Data Mining Application Description Schema. Within this description the developer must specify which options, data inputs, data outputs etc. the application has. As result is the application developer gets an instance of the Data Mining Application Description Schema which describes the applications necessary details. The application

description is later read by the Triana Unit ParameterControl or the generic website. The GUI of these clients automatically adapts to the application description so that the users can easily specify the values for all parameters.

There are two ways of setting up an application description. The first way is to manually create and upload an application description and the corresponding executable into the grid environment. The second way is using a website which has the functionality to guide through the process of describing an application and including it in the grid environment. At the head machine of the grid there is a globus aggregator service *InformationIntegrator*. This service registers the application descriptions at the registry and keeps the application descriptions inside the registry up to date. In detail it is as follows: Within the xml files *informationIntegrator-aggregator-registration.xml* and *jndi-config.xml* located in `$GLOBUS_LOCATION/etc/org_datamininggrid_informationIntegrator/` (which are created automatically when deploying the service) there is information about which script the aggregator has to execute, how often the script is executed, which name the information will have inside the registry etc. The script itself (*concatDescriptions.sh* located in folder `$GLOBUS_LOCATION/libexec/aggrexec/`) will be called each time as specified inside the xml file. Within the script there is a specified path to a folder which will contain the application descriptions. The script searches this folder (including subfolders) for xml files named *ApplicationDescription.xml* and adds them to the MDS.

**Writing an instance of the Data Mining Application Description Schema manually:** In order to include an application into the grid environment manually you have to create a new folder which will contain the new created application description named *ApplicationDescription.xml*. This folder has to be a subfolder of the folder specified in the script belonging to the *InformationIntegrator* service. Additionally you have to upload the executable to the location specified inside the application description.

**Using the Data Mining Application Enabler Web interface as a user-friendly way of writing instances of the Data Mining Application Description Schema:** The Data Mining Application Enabler is a tool, which supports the creation of the description and uploads the executable as well as other necessary files for the application.

An example of a data mining application description is given in Section 6.2 where a simple Hello World application is described.

## 6 Example: Construction of a Simple Grid-enabled 'Hello World' Application

This Section discusses the construction of a simple application and gives a Simple Workflow example. We will use a Hello World application and guide through the steps of creating the application, describing the application according to the application description schema, the inclusion of the application into the grid environment and the construction of a workflow to execute the application on the grid.

### 6.1 Building a simple application

The first step is to create an application which we will include into the grid environment. In this example we will use a java command line application. The parameters of the application can be set via command line with flag-value pairs. The application consists of an executable which prints 'Hello World' into a file. The executable is a jar file `myTestExecutable.jar` which contains the main class `MyTestClass`. The application has one option for the count of the printout and a data output which is the file where the 'Hello World' is printed.

So, a command line call for printing 100 times 'Hello World' into a text file would look like this:

```
java -jar myTestExecutable count 100 result textFile.txt
```

### 6.2 Description of a simple application

The second step is to create an application description of the java application according to the Data Mining Application Description Schema (see Section 5.3). In our case we have two parameters, the option *count* and the data output *result*. In order to describe the application you have to create an instance of the Data Mining Application Description Schema.

Inside the elements *generalInformation* and *applicationType* you have to include some meta data about the application which will later be used to find the application inside the registry (e.g. the id 'HelloWorld'). Within the element *execution* we have to specify what kind of application our application is. In our case it is a java application, so we have to specify some information about the application run file (where it is located, name of the file, main class to execute etc.).

The options and data output have to be described inside the *applicationInformation* element. For the option *count* we create an *option* element of type integer. The flag of the option is 'count'. For the result file we create a *dataOutput* element with the flag 'result'.

Additionally we can pre-set some requirements for the application, for example the minimum memory, minimum disk space etc.

See Appendix for the description of the Hello World application. Please also see the End Users Manual.

### **6.3 Inclusion of a simple application into grid test bed**

To include the Hello World application into the grid environment you have to follow the instructions given in Section 5.4. Furthermore you also can use the data mining Application Enabler website.

## 7 Demonstrator-specific Software

### 7.1 Grid-enabled proprietary or open-source applications

The DataMiningGrid consortium developed demonstrator-specific software. The following table shows a list of applications that were grid enabled:

**Table 1. Grid-enabled applications.**

Name	Group	Functional Area	Vendor	Licence
Id	CrispDM Phase	Technique		
HarvestOneLib	No group	OTHER	University of Ljubljana	Apache License V2
HarvestOneLib	DATA_PREPARATION	OAI-PMH		
EMInduction	FGG-EcologicalModeling	OTHER	Jozef Stefan Institute, Faculty of Civil and Geodetic Engineering	Other Open Source
fgg-lagramge-induction	MODELLING	Induction		
EMSimulation	FGG-EcologicalModeling	OTHER	Jozef Stefan Institute, Faculty of Civil and Geodetic Engineering	Other Open Source
fgg-lagramge-simulation	EVALUATION	Simulation		
PreprocessCorpus	FHG-Textmining	OTHER	Fraunhofer Institute Autonomous intelligent Systems Department Knowledge Discovery (FhG-AIS.KD)	proprietary
fhg1	DATA_PREPARATION	XML Parsing		
Cross-validation	FHG-Textmining	CLASSIFICATION	Fraunhofer Institute Autonomous intelligent Systems Department Knowledge Discovery (FhG-AIS.KD)	modified BSD
fhg2	EVALUATION	SVM		
SVM Classifier training	FHG-Textmining	CLASSIFICATION	Fraunhofer Institute Autonomous intelligent Systems Department Knowledge Discovery (FhG-AIS.KD)	modified BSD
fhg3	MODELLING	SVM		
Crossvalidation	FHG-Textmining	CLASSIFICATION	Fraunhofer Institute Autonomous intelligent Systems Department Knowledge Discovery (FhG-AIS.KD)	modified BSD
fhg5	EVALUATION	Graph Mining		
Ontology Learning	FHG-Textmining	CLUSTERING	Fraunhofer Institute Autonomous intelligent Systems Department Knowledge Discovery (FhG-AIS.KD)	proprietary
fhg6	EVALUATION	Probabilistic Latent Semantic Indexing		
Ontology Recall	FHG-Textmining	CLUSTERING	Fraunhofer Institute Autonomous intelligent Systems Department Knowledge Discovery	proprietary
fhg7	EVALUATION	Probabilistic Latent Semantic Indexing		

(FhG-AIS.KD)				
EAController uu-grn1	Gene modelling MODELLING	OTHER Evolutionary algorithm	Weihenstephan University of Applied Sciences and University of Ulster	LGPL
EAController uu-grn3	Gene modelling MODELLING	OTHER Evolutionary algorithm	Weihenstephan University of Applied Sciences and University of Ulster	LGPL
EAController uu-grn2	Gene modelling MODELLING	OTHER Simple data processing	Weihenstephan University of Applied Sciences and University of Ulster	LGPL
CVStoCAREN uu-pf-pp3	P-Found DATA_PREPARATION	OTHER Formatting	University of Ulster	Proprietary
CAREN uu-pf1	P-Found data warehouse MODELLING	OTHER Apriori algorithm	pja@di.umindo.pt	Proprietary
DiscretizeSimulation uu-pf-pp2	P-Found DATA_PREPARATION	OTHER Discretization	University of Ulster	Open Source
WindowSimulation uu-pf-pp1	P-Found DATA_PREPARATION	OTHER Windowing	University of Ulster	Open Source
J48 weka_j48	Weka MODELLING	CLASSIFICATION Decision Tree	University of Waikato New Zealand	GNU GPL
My Hello Grid World TestApplication Hello World	TestApplications EVALUATION	OTHER none	Fraunhofer Institute Autonomous intelligent Systems Department Knowledge Discovery (FhG-AIS.KD)	modified BSD

## 7.2 DataMiningGrid Workflows

Some developed workflows used for execution of the grid-enables demonstrator applications are also available for download from SourceForge for those system users, who want to get familiar with the system and do some additional exercises.

## 8 Conclusions and Future Work

This document gives a comprehensive overview over the software developed by the DataMiningGrid Consortium.

Following are the most important links to the project resources:

<http://www.datamininggrid.org>

<http://sourceforge.net/projects/datamininggrid/>

## Appendix

### Example application description Hello World application

```
<?xml version='1.0' encoding='UTF-8'?>
<app:application xmlns:app='http://www.datamininggrid.org/applicationDescription'
  xmlns:dm='http://www.datamininggrid.org/datamining'
  xmlns:com='http://www.datamininggrid.org/common'
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:schemaLocation='http://www.datamininggrid.org/applicationDescription
  ../../schema/datamininggrid/information/application_description/dmg_application_description.xsd
  http://www.datamininggrid.org/common dmg_common.xsd
  http://www.datamininggrid.org/datamining dmg_data_mining.xsd '>

  <generalInformation build='1' codeVersion='1.0'
    id='Hello World'
    shortDescription='Run simple program'
    swVersion='1.0'
    uploadDate='2006-08-01'
    vendor='me'>
    <longDescription>Run simple java code which prints out lines.</longDescription>
    <comment></comment>
  </generalInformation>
  <applicationType applicationName='My Hello Grid World TestApplication'
    applicationGroup='TestApplications'
    applicationDomain='data mining'
    crispDMPhase='Evaluation'
    functionalArea='Other'
    technique='none' />
  <execution>
    <javaExecution interpreterCommand=' java' mainClass='myTestPackage.MyTestClass'>
      <interpreterArguments></interpreterArguments>
      <applicationRunFile description='The main jar containing the main class.'
        host='grid2.kd-grid.ais.fraunhofer.de'
        localPath='/gridapps/test'
        fileDirParameterName='myTestExecutable.jar'
        protocol='gsiftp' port='2811' />
    </javaExecution>
  </execution>
  <applicationInformation>
    <options dataType='int'
      defaultValues='5'
      flag='count'
      hidden='false'
      label='Count'
      optional='false'
      tooltip='number of printed lines' />
  <!-- All possible data inputs. All printings to SDTOUT & STDERR by the application are covered by the
  system automatically and thus have not to be specified here. -->
  <!-- All possible data outputs -->
    <dataOutputs flag='result'
      ioType='Datafile'
      label='Output file'
      tooltip='Name of output file.'
      optional='false'
      hidden='false'
      stageOut='true'
      appendToCmdLine='true' />
  <!-- Requirements for execution machines -->
    <requirements>
      <minMemory label='Min memory' value='10' unit='MB' tooltip='Minimum memory of execution
      machine must be larger than minimum memory required for JVM.' />
      <minDiskSpace label='Min free disk space' value='1' unit='GB'
      tooltip='Minimum required disk space. Actual required disk space depends on the size of the
      input data and should be chosen with care.' />
      <operatingSystem label='Operating system' value='ALL' />
      <architecture label='Architecture'>
        <value>ALL</value>
      </architecture>
    </requirements>
  </applicationInformation>
</app:application>
```