



USER MANUAL
GRID MONITORING AND DATA ANALYSIS TOOL

WP3.2

Document Filename:	CG3.2-v1.0-UAB-GMDATUserManual.doc
Work package:	WP3.2 – GMDAT
Partner(s):	ICM
Lead Partner:	UAB
Config ID:	CG3.2-v1.0-UAB-GMDATUserManual
Document classification:	PUBLIC

Document Log

Version	Date	Summary of changes	Author
0.1	17/08/2004	Draft version	Piotr Nowakowski
0.2	22/08/2004	Divided chapter on usage into subsections, removed installation info (to be delivered as a separate document), provided examples basing on WP3 deliverables	Piotr Nowakowski
0.3	29/09/2004	Inclusion of TB remarks	Ariel Garcia, Piotr Nowakowski
1.0	29/11/2004	GMDAT User Manual - Draft	K.Nawrocki, A. Padee, K. Wawrzyniak
	26/01/2005	Verified by the QE	Robert Pajak

CONTENTS

COPYRIGHT NOTICE	4
1. INTRODUCTION.....	5
1.1. ABBREVIATIONS AND ACRONYMS	5
1.2. REFERENCES AND SOURCE CODE	5
2. PRODUCT USAGE	6
2.1. RUNNING THE PRODUCT	6
2.1.1. <i>Operating Requirements</i>	6
2.1.2. <i>Step-by-Step User Setup</i>	7
2.2. BASIC OPERATION	7
2.2.1. <i>Executing from command line</i>	7
2.3. ADVANCED FEATURES.....	10
2.4. KNOWN PROBLEMS.....	12
3. INTERFACE REFERENCE GUIDE.....	13
4. THE EDG LICENSE AGREEMENT	16

COPYRIGHT NOTICE

Copyright (c) 2005 by ICM. All rights reserved.

Use of this product is subject to the terms and licenses stated in the EDG license agreement. Please refer to Section 6 for details.

This research is partly funded by the European Commission IST-2001-32243 Project “CrossGrid”.

1. INTRODUCTION

GMDAT is a monitoring part of the CrossGrid scheduler. The main foundation of the tool was to create a lightweight monitoring system, working on 24h/day basis, being able to deliver data describing Grid status to the Grid scheduler. The tool consists of three main parts: sensors, installed on each computer in the Grid; central database gathering summary information from the clusters; and the data analysis module for preparation of the predictions of the future Grid status. Existing solutions were used and extended wherever it was possible. Sensors are built on top of Ganglia monitoring system, the central database uses Round Robin Database (RRD) format and SOAP interface to the scheduler. The data analysis module is written from scratch and utilizes Kalman filter to predict the behavior of the Grid.

Additionally to this document, there are available an installation guide and a developer manual.

1.1. ABBREVIATIONS AND ACRONYMS

API	Application Program Interface
CE	Computing Element
CMH	Central Monitoring Host
CrossGrid	The EU CrossGrid Project IST-2001-32243
CPU	Central Processing Unit
CVS	Concurrent Versioning System
DAO	Data Access Optimization
Globus	Grid middleware
Gmdat	Grid Monitoring Data Analysis Tool
JIMS	
NWS	Network Weather Service
PERL	http://www.perl.org
RB	Resource Broker
RPMS	Red Hat Package Management (packages)
RRDtool	Round Robin Database, http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/license.html
SE	Storage Element
SOAP	Single Object Access Protocol
UI	User Interface
XML	eXtensible Markup Language, http://www.w3.org/XML

1.2. REFERENCES AND SOURCE CODE

The source code of this application and how to install it is fully described in the Installation Guide of the application.

2. PRODUCT USAGE

The general structure of the GMDAT is presented in the Fig.1.

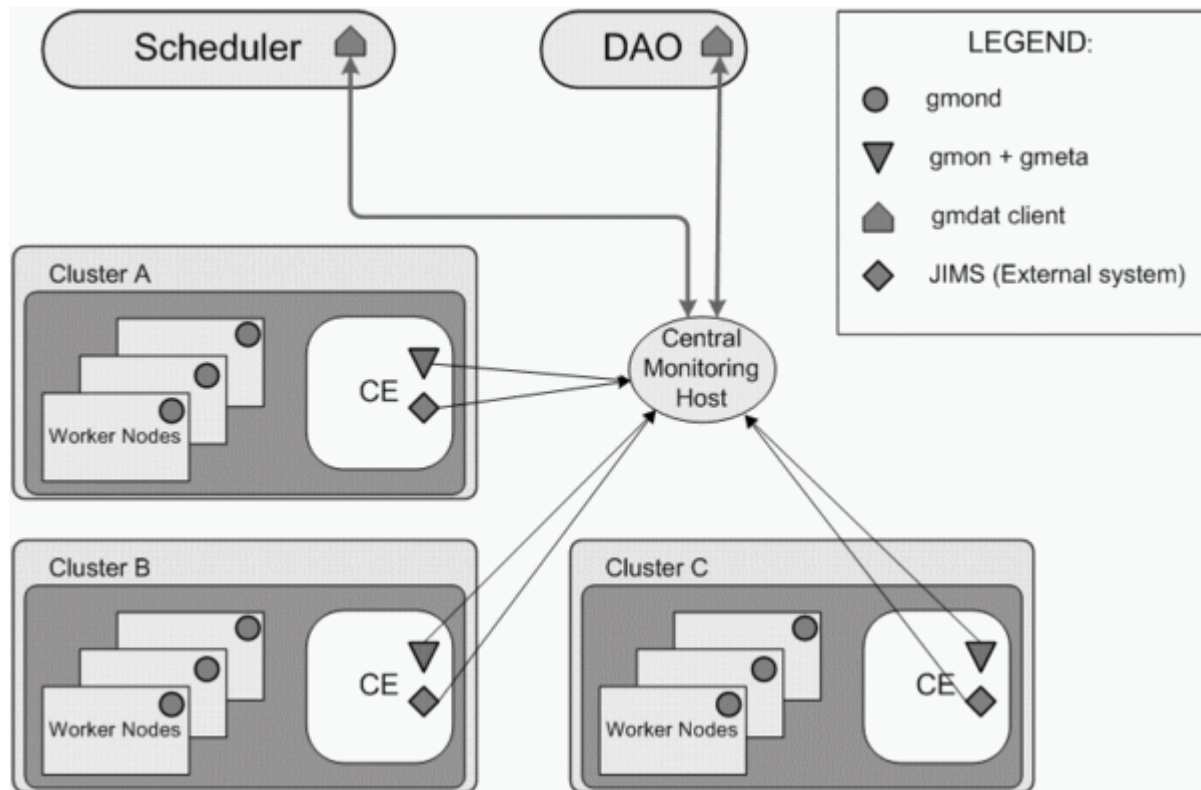


Fig. 1

The following ways can be used to access the data provided by GMDAT:

- using the web interface, available at the address: http://cmh_host/cgi-bin/gmdat/engine/grid2.pl, where cmh_name represents name of **Central Monitoring Host** – **CMH**, at the moment of writing this document cmh_name is grid.fuw.edu.pl)
- using the gmdat_sensor_client tools. gmdat_sensor_client tools are able to connect to the CMH and retrieve GMDAT data using the SOAP interface.

Both ways are described in details further in this document.

2.1. RUNNING THE PRODUCT

If you run the model into the CrossGrid Testbed there are no specific requirements.

2.1.1. Operating Requirements

2.1.1.1. Local hardware requirements

There are no specific requirements for executing the application.

2.1.1.2. Local software requirement

The program has been developed and tested on the RedHat 7.3. All software requirements are specified in the GMDAT's RPMS.

2.1.1.3. Grid infrastructure requirements

The program has been developed to work in the Globus testbed infrastructure. It assumes the Grid structure as presented in Fig. 1.

2.1.2. Step-by-Step User Setup

There is no specific user setup for running this application assuming that the software on CMH and gmdat_sensors has been installed on Grid clusters.

2.2. BASIC OPERATION

The GMDAT is available on the CrossGrid Testbed.

If you want to execute it locally, you can download it from the CVS repository of the CrossGrid project http://savannah.fzk.de/cgi-bin/viewcvs.cgi/crossgrid/crossgrid/wp3/wp3_3-moninfr/wp3_3_4-postproc and install all pre-requisites specified in RPM Spec files.

2.2.1. Executing from command line

User wanting to use data provided by GMDAT can use gmdat_sensor_client consisting of the following tools:

- gmdat_update_db script
- gmdat_read_db script
- the C++ API to give a programmer access to the GMDAT data from within a C++ program

All components are described below.

- /opt/cg/bin/gmdat_update_db program is used for retrieving data and predictions from the CMH server in the XML format and store it locally

Synopsis of gmdat_update_db program is the following:

```
$ gmdat_update_db -xml -db -v
```

-xml The data for clusters specified in the file /opt/cg/etc/clusters.cf and metrics specified in the file /opt/cg/etc/metrics.cf is written to the file /opt/cg/var/gmdat/state.xml

-db Available bandwidth data between CrossGrid testbed clusters specified in the file /opt/cg/etc/clusters.cf are written to the mysql database. This usage is reserved for the DAO.

-v The data for clusters specified in the file /opt/cg/etc/clusters.cf and metrics

specified in the file `/opt/cg/etc/metrics.cf` is written to the STDOUT

Execution of the `gmdat_update_db` can be added to the crontab to get data periodically.

- `/opt/cg/bin/gmdat_update_db` program can be used to query the locally stored database (in the form of the file `/opt/cg/var/gmdat/state.xml`) about a value of metric for a given cluster and for specified time.

Synopsis of `gmdat_update_db` program is the following:

```
$ gmdat_read_db -c cluster -m metric -t unixtime
```

- the C++ API can be used to query the locally stored database about GMDAT data from within a C++ program . The library `libppt.so` is stored in the directory `/opt/cg/lib` and the include files `ppt.h` and `ppt_exceptions.h` in the directory `/opt/cg/include`. Below an example self-explaining program of the usage of the C++ API is presented:

```
#include "ppt.h"
#include "ppt_exceptions.h"

int main() {
try {
// declare the postprocessing object with the name of file retrieved
// using gmdat_update_db tool as parameter
    Postprocessing myXML = Postprocessing("/opt/cg/var/gmdat/state.xml ");

// define set of clusters for which parameters are to be calculated
    set<string> clusters;
    clusters.insert("xgrid.icm.edu.pl");
    clusters.insert("cms.fuw.edu.pl");
    clusters.insert("ce02.lip.pt");

// calculate minimum value of 'idle_bogomips' for defined set of clusters for time=0 (ie. real data)
    cout << "MIN " << myXML.getClusterParameter("min","idle_bogomips",clusters,0) << endl;

// calculate maximum value of 'idle_bogomips' for defined set of clusters for time=0.6 min
// (ie. data and predictions for next 6 minutes for each cluster from defined set are averaged and
// maximum value of these values over clusters is returned)
    cout << "MAX " << myXML.getClusterParameter("max","idle_bogomips",clusters,6) << endl;

// calculate maximum value of 'bandwidth' between all pairs of clusters for defined set
// for time=0 (ie. real data)
    cout << "MAX " << myXML.getNetParameter("max","bandwidth",clusters,0) << endl;
```

```
// calculate average value of 'bandwidth' between all pairs of clusters for defined set for time=0..15 min
// (ie. data and predictions for next 15 minutes for each cluster pair from defined set are averaged and
// the average value of these values over pairs of clusters is returned)
    cout << "AVG " << myXML.getNetParameter("avg","bandwidth",clusters,15) << endl;
}
catch (CannotLoadXmlFile &ex) {
    ex.what();
}
catch (VectorIsEmpty &ex) {
    ex.what();
}
catch (NoDataForMetric &ex) {
    ex.what();
}
return 0;
}
```

Assuming that above program is stored in the file testppt.cc the following makefile can be used to compile the program:

```
CXXFLAGS = -c -g -Wall -DDEBUG
LDLIBS = -lscew -lexpat

# object definitions -----
#
OBJ = ppt.o ppt_exceptions.o

# targets -----

all:
    make clean
    make libppt
    make testppt

libppt: ${OBJ}
    touch libppt.so
    rm libppt.so
    ${CXX} -shared -o libppt.so ${OBJ}

testppt: testppt.o libppt.so
    ${CXX} testppt.o ./libppt.so $(LDLIBS) -o $@

clean:
    @rm -f *.o *.so testppt
```

2.3. ADVANCED FEATURES

One of the **GMDAT (Grid Monitoring Data Analysis Toolkit)** components is the **predict** application which is responsible for building forecasts. This application is independent from the rest GMDAT components and can be treated as separated, alone standing application. This logical consistency of **predict** allows to use it with various kinds of data not only with monitoring parameters. If you decide to use **predict** directly, you have to manually provide time series to analysis.

Synopsis of predict is as follows:

```
predict [-lup] [STEPS] [-s SOURCE] [-d DEST] [-c CONF] [-k STATE] [-b UNIX_TIME]
[-e UNIX_TIME] [-i LOWER_MARGIN] [-j UPPER_MARGIN]
```

Two or three steps are necessary to construct a prediction by this software:

1. Learn the model

Construct and learn model according to the historical time series stored in *SOURCE* file. Model parameters are included in *CONF* file.

2. Update the model [optional]

Update model if new samples were gathered. The model does not have to be rebuilt when new data are taken into account. Application finds last sample in *SOURCE* file which was used during learning process and updates model according to new samples.

3. Make prediction

Prediction for the next n steps is calculated. This mode is only available if the model is after the learning process.

Taking these three steps is possible by calling predict with proper options. Three main options and set of auxiliary options are delivered.

MAIN OPTIONS

Main options are disjunctive - you can not use those options together.

- l** Start learning process. This option has to be, at least, connected with option -s. You can also use options: -d, -c, -b, -e, -k, -i, -j.
- u** Start updating process. Option -u occurs alone or with -d -e and -k, -i, -j.
- p STEPS** Start prediction process on the next **STEPS** samples (where **STEPS** is numerical value). It is obligatory to use also option -d. Additionally options: -k, -i, -j are also possible.

AUXILARY OPTIONS

-
- s SOURCE** Full path to the *SOURCE* file. File has to be in appropriate format: **UNIX_TIME x_value**. -s can occur only with option -l.
- d DEST** Full path to the *DEST* file. *DEST* file has following structure: **UNIX_TIME x_value d_value y_value**. Where **x_value** - input sample, **y_value** - output sample, **d_value** - desired sample at time **UNIX_TIME**. -d can occur with options: -l, -u, -p.
- c CONF** Full path to the *CONF* file. This file includes filter parameters. Default location for *CONF* file is /etc/param.txt. By using '-c' you can specify other location. -c can occur only with option -l.
- k STATE** Full path to the *STATE* file. This file includes state of model which was achieved on the end of learning process. Default location for *STATE* file is /etc/kalman.dat. By using '-k' you can specify other location. *STATE* file is created during learning process and is modified during update process. Existence of this file is also obligatory for prediction process.
- b UNIX_TIME** Begin learning process from sample **UNIX_TIME**.
-b can occur only with option -l.
- e UNIX_TIME** Finish learning process on sample **UNIX_TIME**.
-s can occur with options -l, -u.
- i LOWER_MARGIN** Limit minimal value of signal to **LOWER_MARGIN**.
- j UPPER_MARGIN** Limit maximal value of signal to **UPPER_MARGIN**.

EXAMPLES

Below some possible combinations:

Example 1

Execute learning process using samples from *source.txt* Take into consideration samples from 1051661700 to 1053041100. Save results in file *dest.txt*. *dest.txt* contains prediction for one step.

After that, upgrade model according to new samples in file *source.txt* but not newer than 1053041700. Add results of updating process to the file *dest.txt*. Forecast next 200 samples and save prediction in file *prediction.txt*.

```
predict -l -s source.txt -d dest.txt -b 1051661700 -e 1053041100
predict -u -d dest.txt -e 1053041700
predict -p 200 -d prediction.txt
```

Example 2

Execute learning process using all samples from *source.txt*. Forecast next 50 samples and save prediction in file *prediction.txt*

```
predict -l -s source.txt  
predict -p 50 -d prediction.txt
```

Example 3

Execute learning process. Read model parameters from *param.txt* file. The signal is expressed as a percentage, thus constraints for signal are feasible. When learning process is finished, dump info about state of model to *state.dat* file. Location of source data is *source.txt* file. Take into consideration samples from 1051661700 to 1053041100. Save results in file *dest.txt*. *dest.txt* contains prediction for one step.

After that, upgrade model according to new samples in file *source.txt* but not newer than 1053041700. Because during learning process you redefined default localization of *STATE* file now you have to specify where *STATE* file is located (option -k). Add results of updating process to the file *dest.txt*.

Forecast next 200 samples and save prediction in file *prediction.txt*

```
predict -l -c param.txt -k state.dat -s source.txt -d dest.txt -b 1051661700 -e 1053041100 -i 0 -j 100  
predict -u -d -k state.dat dest.txt -e 1053041700 -i 0 -j 100  
predict -p 200 -k state.dat -d prediction.txt -i 0 -j 100
```

Additional remarks to above example:

You can see, that option -c was used just one time, during learning process. Updating and prediction processes don't need publicly defined *CONF* file location, because this information is stored in *STATE* file.

2.4. KNOWN PROBLEMS

All the known bugs at the moment of write this manual are corrected in the available version.

3. INTERFACE REFERENCE GUIDE

GUI description:

GMDAT Grid Monitoring Data Analysis Toolkit includes web interface (**grid.pl** located at http://cmh_name/cgi-bin/gmdat/engine/grid.pl, where cmh_name represents name of **Central Monitoring Host - CMH**) designed to visualize history and prediction of clusters metrics. Metrics represent mean values for whole cluster. These mean values are computed for 5 minutes intervals.

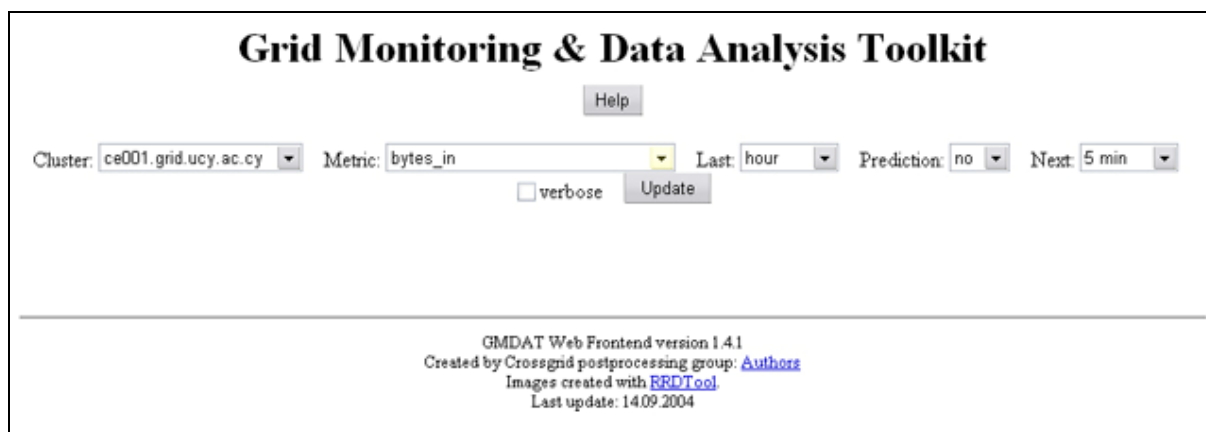


Fig. 2

The snapshot of the front-end is presented at figure 2. The basic function of web front-end is to present historical time series for chosen metric. Additional option allows also achieving prediction for given period (option **Prediction** has to be set on "yes"). Prediction is constructed by **predict** application. According to intuition, first samples of prediction should quite precisely imitate real data. When prediction period becomes longer, then time series go smoother and prediction becomes more and more general.

Another option is **Verbose** mode. This option allows visualizing learning or updating process and is especially useful for administrators to adjust Kalman filter parameters.

GUI description – admin's part:

If quality of prediction is not sufficient it is possible to adjust filter parameters by **KFAP - Kalman Filter Administration Panel** (http://cmh_name/cgi-bin/gmdat/admin/adm.pl – snapshot is presented at figure 3).

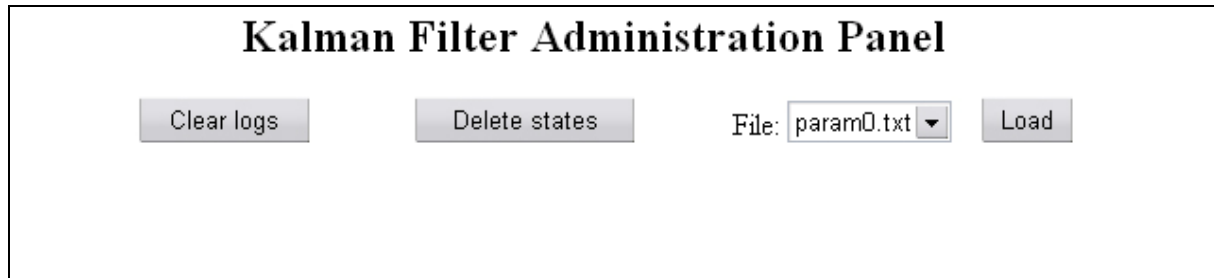


Fig. 3

To adjust parameters make as follows:

1) At first you have to delete old state of examined metric (dir: var/gmdat/predict/states). You can do it manually from command line (recommended) or by KFAP.

Manually - Log on **Central Monitoring Host (CMH)** and go to /etc/gmdat/predict/states. Find file with state responsible for interesting metric and remove it.

By KFAP - Open administration panel: http://cmh_name/cgi-bin/gmdat/admin/adm.pl and choose delete option (notice that if you decide to use KFAP all states will be deleted which in some cases would not be advised).

2) Load file with interesting parameters and modify them using KFAP. Save parameters.

Some groups of metrics have similar properties, thus they can be described by one set of parameters. For metrics we are created only seven sets of parameters. It seems to be enough to describe metrics' characteristics. Files including parameters match metrics according to relation presented below:

files with parameters	group of metrics
param1	metrics connected with cpu
param2	traffic metrics
param3	rrt metrics
param4	load metrics
param5	idle_bogomips metrics
param6	bogomips metrics
param0	other metrics

3) Using GMDAT (http://cmh_name/cgi-bin/gmdat/engine/grid.pl) choose historical period for metric. In initial phase, regardless of the length of historical data learning process of

4) Check quality of prediction using **Updating** option on web page. Remember to check if the **Verbose box** is set on, what allows you to trace learning process. Period used during learning process is standardized and is set to 30-day period. For interval equaling five minutes it gives us 8640 samples what seems to be much more than is necessary to teach model. If data stored in rrd are shorter than 30 days, then longest consistent data are used.

5) Analyze generated figures and Mean Square Error value. If quality of learning process is not sufficient try to modify parameters and repeat all steps.

4. THE EDG LICENSE AGREEMENT

Copyright (c) 2005 CrossGrid. All rights reserved.

This software includes voluntary contributions made to the CrossGrid Project. For more information on CrossGrid, please see <http://www.eu-crossgrid.org>.

Installation, use, reproduction, display, modification and redistribution of this software, with or without modification, in source and binary forms, are permitted. Any exercise of rights under this license by you or your sub-licensees is subject to the following conditions:

1. Redistributions of this software, with or without modification, must reproduce the above copyright notice and the above license statement as well as this list of conditions, in the software, the user documentation and any other materials provided with the software.

2. The user documentation, if any, included with a redistribution, must include the following notice: “This product includes software developed by the CrossGrid Project (<http://www.eu-crossgrid.org>).”

Alternatively, if that is where third-party acknowledgments normally appear, this acknowledgment must be reproduced in the software itself.

3. The names “CrossGrid” and “CG” may not be used to endorse or promote software, or products derived therefrom, except with prior written permission by cgooffice@cyfronet.krakow.pl.

4. You are under no obligation to provide anyone with any bug fixes, patches, upgrades or other modifications, enhancements or derivatives of the features, functionality or performance of this software that you may develop. However, if you publish or distribute your modifications, enhancements or derivative works without contemporaneously requiring users to enter into a separate written license agreement, then you are deemed to have granted participants in the CrossGrid Project a worldwide, non-exclusive, royalty-free, perpetual license to install, use, reproduce, display, modify, redistribute and sub-license your modifications, enhancements or derivative works, whether in binary or source code form, under the license conditions stated in this list of conditions.

5. DISCLAIMER

THIS SOFTWARE IS PROVIDED BY THE CROSSGRID PROJECT AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, OF SATISFACTORY QUALITY, AND FITNESS FOR A PARTICULAR PURPOSE OR USE ARE DISCLAIMED. THE CROSSGRID PROJECT AND CONTRIBUTORS MAKE NO REPRESENTATION THAT THE SOFTWARE, MODIFICATIONS, ENHANCEMENTS OR DERIVATIVE WORKS THEREOF, WILL NOT INFRINGE ANY PATENT, COPYRIGHT, TRADE SECRET OR OTHER PROPRIETARY RIGHT.

6. LIMITATION OF LIABILITY

THE CROSSGRID PROJECT AND CONTRIBUTORS SHALL HAVE NO LIABILITY TO LICENSEE OR OTHER PERSONS FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, OR PUNITIVE DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOSS OF USE, DATA OR PROFITS, OR BUSINESS INTERRUPTION, HOWEVER CAUSED AND ON ANY THEORY OF CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.