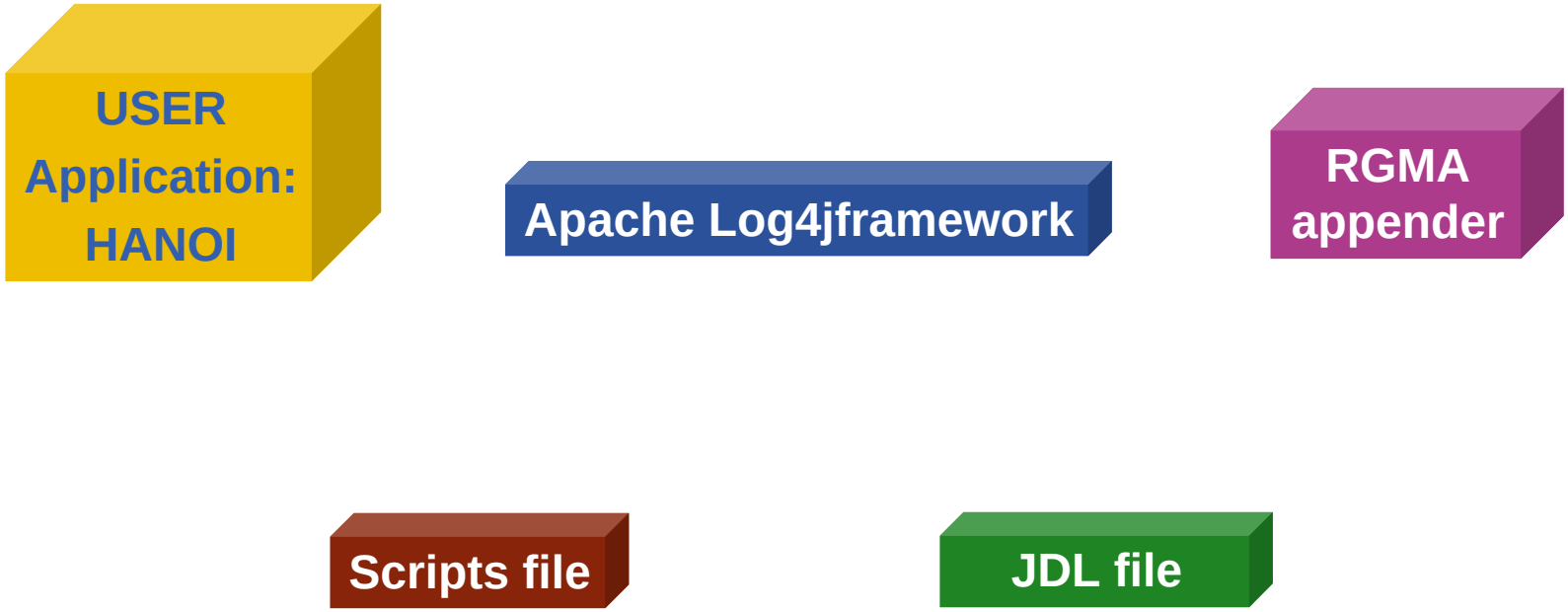
Practicals on RGMA

Valeria Ardizzone
INFN Catania

gLite Application Developers Course
CERN, 23.10.2006

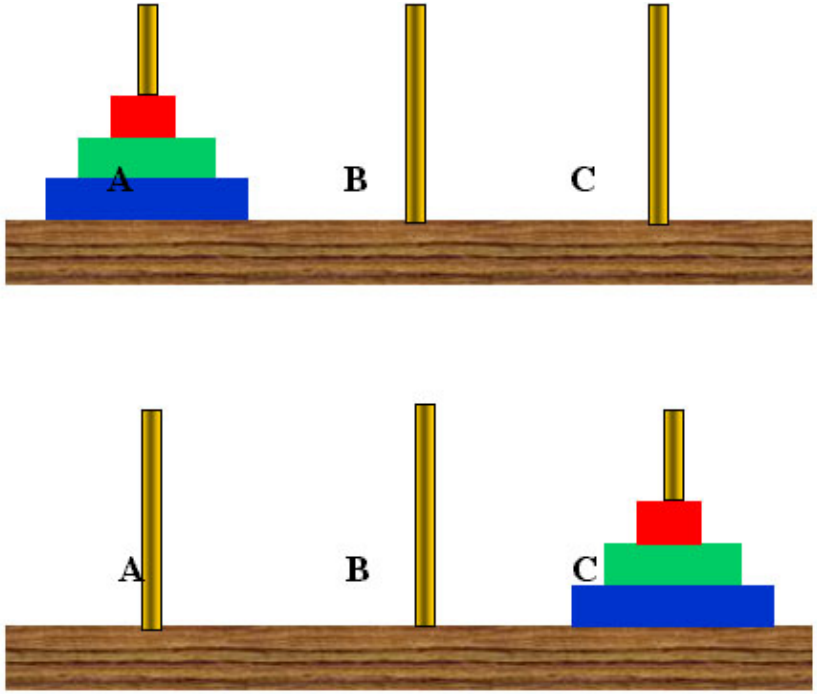
Exercise:

https://grid.ct.infn.it/twiki/bin/view/GILDA/ApplicationMonitoringRGMA



Application

- **Tower of Hanoi will be the application to use like example.**



The aim is to move the discs, one at a time, from one pole to another, such that a larger disc is never placed on top of a smaller disc, until all the discs are arranged on one of the initially empty poles.

Apache Log4j framework

- Log4j has three main components:

  1. Loggers
  2. Appenders
  3. Layouts

- These three types of components work together to enable developers to log messages according to message type and level, and to control at runtime how these messages are formatted and where they are reported.

Apache Log4j Logger

- **Loggers may be assigned levels. The set of possible levels is DEBUG, INFO, WARN, ERROR and FATAL.**
- **Basic Selection Rule:**

  A log request of level p in a logger with (either assigned or inherited, whichever is appropriate) level q, is enabled if p > q.

```
.....
static Logger logger = Logger.getLogger(HanoiLog.class.getName());
.....
logger.info("Move the disk " + n + " from pole " + p1 + " to pole " + p2 + ".");
.....
logger.debug("Entering application.");
.....
logger.warn("Disk: " + n + " Initial Pole: " + p1 + " Final Pole: " + p2 + " Total
    steps: "+count);
```

Apache Log4j Appender

- Log4j allows logging requests to print to multiple destinations. In log4j speak, an output destination is called an appender.
- Currently, appenders exist for the console, file, GUI components, remote socket servers, JMS, NTEvent Loggers and remote UIX Syslog daemon.
- The R-GMA log4j appender allows you to to publish the logging events into R-GMA instead of the standard log4j appender.
- This utility is built on top of the log4j framework by providing a customized appender that slots into R-GMA.

Configuration file: log4jRGMA.properties

Normally, the user defines a properties file that defines the verbosity level of logging and where logging information will be placed. The user may route logging output to stdout, to file or to a socket and so on.

```
log4j.rootLogger=DEBUG, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=[%t] %d %-5p - %m%n

log4j.logger.HanoiLog=INFO, rgma, file
log4j.appender.rgma=MyAppender
log4j.appender.rgma.JobName=HanoiLog
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=HanoiLog.log
log4j.appender.file.MaxFileSize=100KB
log4j.appender.file.MaxBackupIndex=1
……
```

MyAppender(1)

- This method called by the log4j system once all the properties have been initialised from the config file.
- The Producer is created here because the predicate depends upon the JobName thats parsed from the log4j properties file.

```
public void activateOptions() {
    if ((jobName == null) || jobName.equals(""))
    {       errorHandler.error(MyErrors.UNDEFINED_JOBNAME); }
    else {
            createProducer(); }
}
```

MyAppender(2)

- **The MyAppender Class traps any log4j calls and then publishes them using an R-GMA 'Database' Primary Producer.**

```
private void createProducer() {
try { ….
    ProducerFactory factory = (ProducerFactory) factoryClass.newInstance();
     ProducerProperties props = new
    ProducerProperties(Storage.DATABASE, ProducerProperties.LATEST);
    TimeInterval ti = new TimeInterval(20, Units.MINUTES);
    producer = factory.createPrimaryProducer(ti, props);
    TimeInterval retenPeriod = new TimeInterval(1, Units.HOURS);
    String predicate = "WHERE JobName = '" + jobName + "'";
    producer.declareTable(MyConstants.TABLE_NAME, predicate,
    retenPeriod, retenPeriod);
} catch (Exception e) { errorHandler.error(MyErrors.FAILED_CREATE_PROD,
    e, ErrorCode.GENERIC_FAILURE); } }
```

MyAppender(3)

- **Log4j calls the append method when a logging event is generated.**

```
public void append(LoggingEvent event)
{ ....
   try
{    StringBuffer msg = new StringBuffer();
     msg.append(event.getMessage());
     if (layout.ignoresThrowable())
    { msg.append(formatException(event)); }
    if (msg.length() > MyConstants.MAX_VARCHAR_SIZE)
    {        msg.setLength(MyConstants.MAX_VARCHAR_SIZE); }
    producer.insert("INSERT INTO " + MyConstants.TABLE_NAME + "
      VALUES ('" + jobName + "', '" + millis + "', '" + threadName + "', '" + level +
      "', '" + className + "', '" + msg + "')");
 } catch (Exception e)
```

Stdout Appender log messages

.....

[main] 2006-09-23 11:06:27,064 DEBUG - entering PropertyGetter::getProperty - PrimaryProducer

[main] 2006-09-23 11:06:27,066 DEBUG - using property file: /opt/glite/etc/rgma/rgma.conf

[main] 2006-09-23 11:06:27,067 DEBUG - entering PropertyGetter::parse

[main] 2006-09-23 11:06:27,068 DEBUG - looking for key value: PrimaryProducer

[main] 2006-09-23 11:06:27,085 DEBUG - exiting PropertyGetter::parse - [https://rgmasrv.ct.infn.it:8443/R-GMA/PrimaryProducerServlet]

[main] 2006-09-23 11:06:27,085 DEBUG - exiting PropertyGetter::getProperty - https://rgmasrv.ct.infn.it:8443/R-GMA/PrimaryProducerServlet

[main] 2006-09-23 11:06:27,102 DEBUG - Connect using GET to: https://rgmasrv.ct.infn.it:8443/R-GMA/PrimaryProducerServlet/createPrimaryProducer?terminationIntervalSec=1200&type=database&isLatest=true&isHistory=false

[main] 2006-09-23 11:06:27,103 DEBUG - Trust properties file: null

[main] 2006-09-23 11:06:27,103 DEBUG - Grid proxy file: /tmp/x509up_u515

[main] 2006-09-23 11:06:27,103 DEBUG - CA dir: null

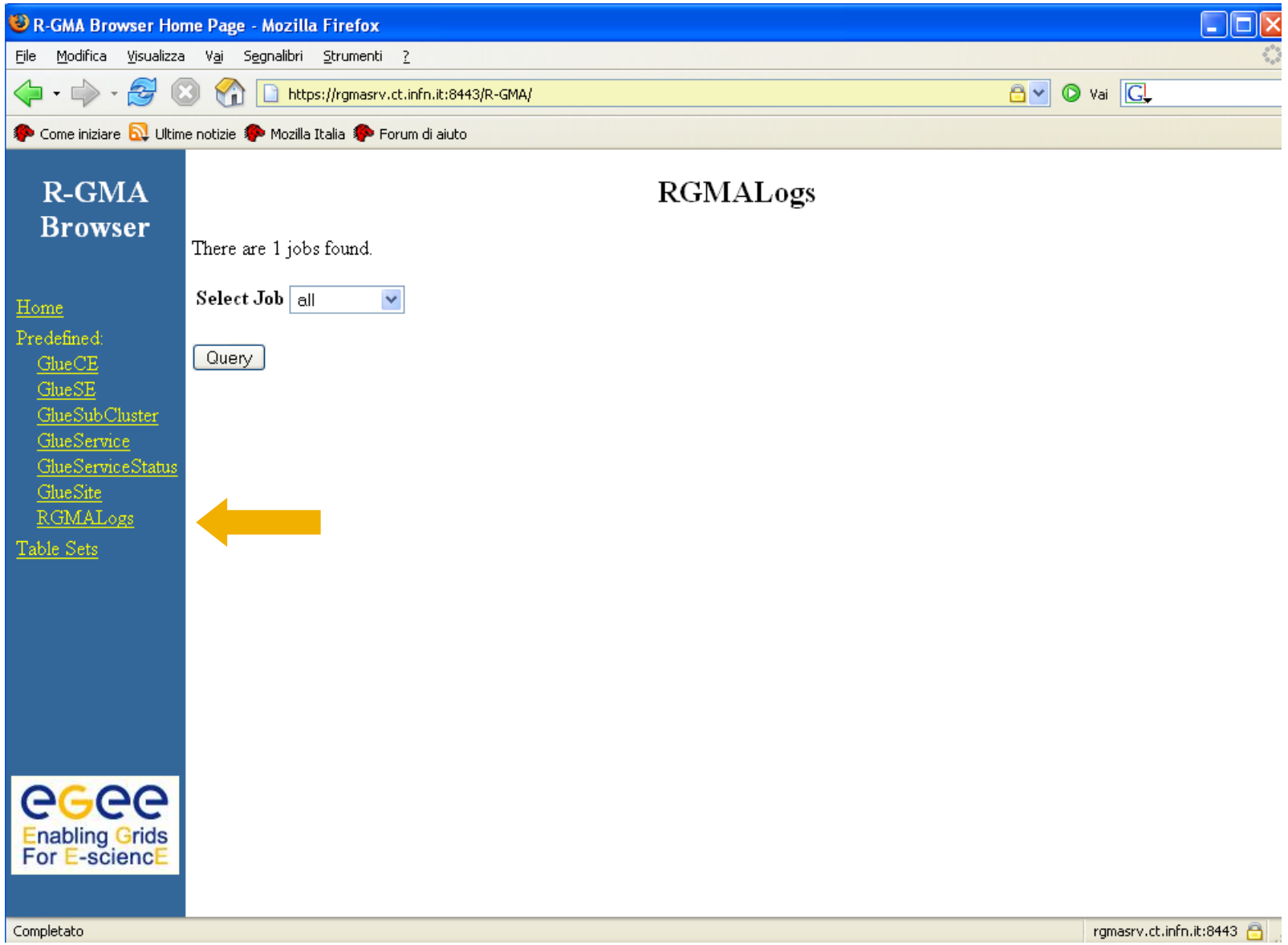
[main] 2006-09-23 11:06:27,103 DEBUG - X509_USER_PROXY: /tmp/x509up_u515

.....

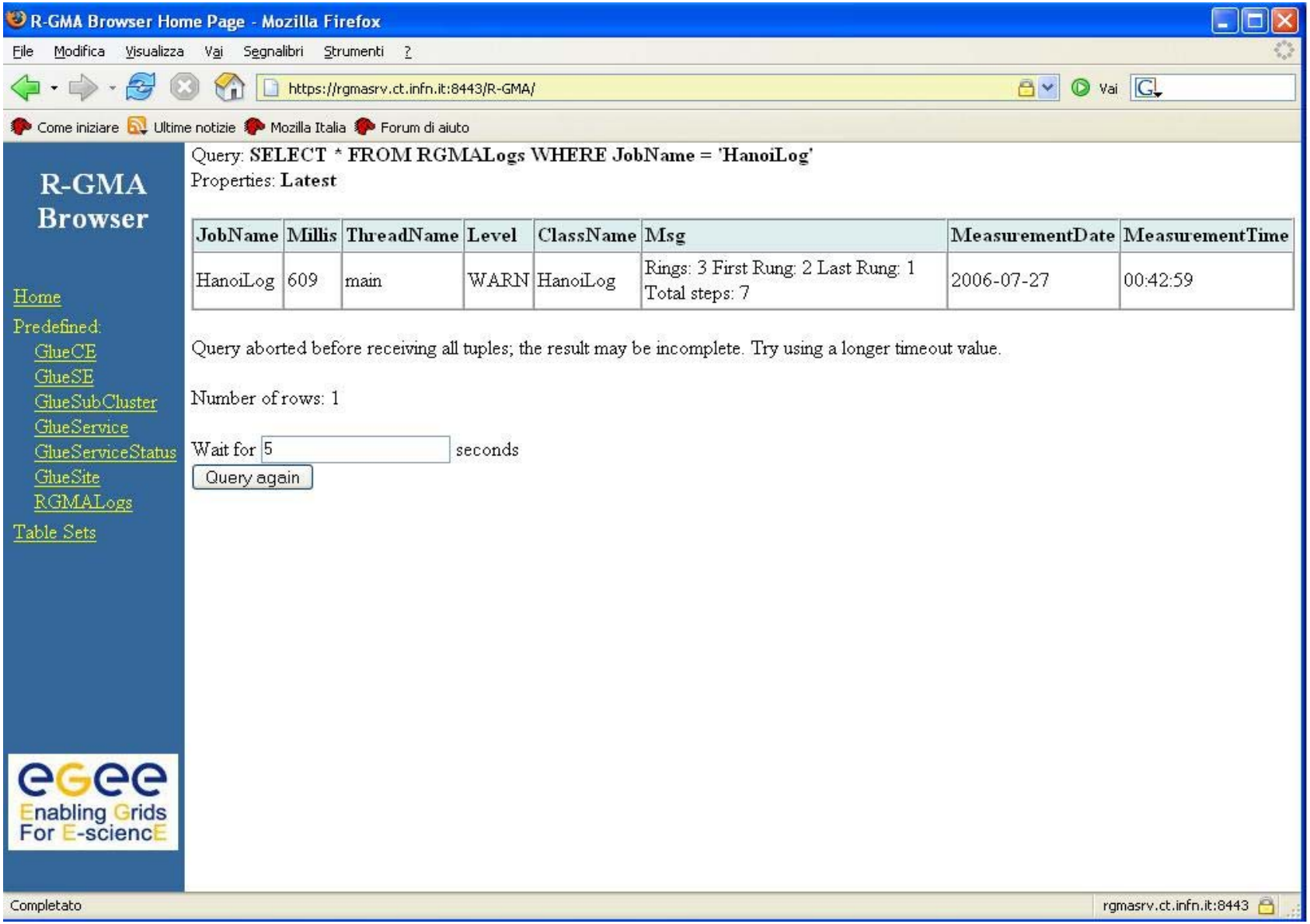
HanoiLog.log

INFO main HanoiLog - Move the ring 1 from rung 2 to rung 1.
INFO main HanoiLog - Move the ring 2 from rung 2 to rung 3.
INFO main HanoiLog - Move the ring 1 from rung 1 to rung 3.
INFO main HanoiLog - Move the ring 3 from rung 2 to rung 1.
INFO main HanoiLog - Move the ring 1 from rung 3 to rung 2.
INFO main HanoiLog - Move the ring 2 from rung 3 to rung 1.
INFO main HanoiLog - Move the ring 1 from rung 2 to rung 1.
WARN main HanoiLog - Rings: 3 First Rung: 2 Last Rung: 1 Total steps: 7

HanoiLog on RGMALogs table (1)



HanoiLog on RGMALogs table (2)



RGMA: Environment

```
#!/bin/sh
export RGMA_HOME=$GLITE_LOCATION
TEST_CLASSPATH="$RGMA_HOME/share/java/glite-rgma-log4j.jar:$RGMA_HOME/libexec/rgma-log4j/"

TRUSTMANAGER_CLASSPATH="$RGMA_HOME/share/java/glite-security-trustmanager.jar"
SECURITY_UTIL_CLASSPATH="$RGMA_HOME/share/java/glite-security-util-java.jar"
BOUNCYCASTLE_CLASSPATH="$RGMA_HOME/share/glite-security-trustmanager/bcprov-jdk14-122.jar"

SECURITY_CLASSPATH=$TRUSTMANAGER_CLASSPATH:$SECURITY_UTIL_CLASSPATH:$BOUNCYCASTLE_CLASSPATH
SECURITY_PROPERTIES="-DX509_USER_PROXY=$X509_USER_PROXY"

RGMA_CLASSPATH="$TEST_CLASSPATH:$RGMA_HOME/share/java/glite-rgma-api-java.jar:$RGMA_HOME/share/java/glite-rgma-stubs-servlet-java.jar"
export RGMA_PROPERTIES="-DRGMA_HOME=$RGMA_HOME"

LOG4J_CLASSPATH="$RGMA_HOME/externals/share/java/log4j-1.2.8.jar:$RGMA_HOME/externals/share/java/log4j.jar"
export LOG4J_PROPERTIES="-Dlog4j.configuration=file:./log4jRGMA.properties"

export CLASSPATH="$RGMA_CLASSPATH:$LOG4J_CLASSPATH:$SECURITY_CLASSPATH:./RGMALog.jar"
```

JDL File

If you want to run this application on grid element, you need to prepare the following jdl file to submit:

```
Type = "Job";
JobType = "Normal";
Executable="script.sh";
Arguments = "3 2 1";
StdOutput="stdout.log";
StdError="stderr.log";
InputSandbox={"script.sh","RGMALog.jar","log4jRGMA.properties"};
OutputSandbox={"stdout.log","stderr.log","HanoiLog.log"};
```

More information

- **R-GMA overview page.**
  - http://www.r-gma.org/
- **R-GMA documentation in EGEE**
  - http://hepunx.rl.ac.uk/egee/jra1-uk/
- **R-GMA in GILDA Testbed**
  - https://rgmasrv.ct.infn.it:8443/R-GMA